UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS



TESIS

"EVALUACIÓN DE TÉCNICAS DE APRENDIZAJE DE MÁQUINA PARA LA IDENTIFICACIÓN DE IMÁGENES DE EDIFICIOS HISTÓRICOS DE LA CIUDAD DEL CUSCO BASADO EN BAG-OF-WORDS Y REDES NEURONALES CONVOLUCIONALES"

Para optar al título profesional de:
INGENIERO INFORMÁTICO Y DE SISTEMAS

Presentado por:
Br. Jeanfranco David Farfan Escobedo

Asesor :
Dr. Lauro Enciso Rodas

Co-asesor :
M.Sc. John Edgar Vargas Muñoz

CUSCO - PERÚ
2017

Este trabajo está dedicado a toda mi familia; en especial a mis padres Senovia y David, por brindarme su apoyo incondicional; a mis hermanos Werner y Juan Pablo, por servirme de modelo durante mi vida académica; a mis tías Delfina y Doris, por acompañarme en los momentos de salud y enfermedad; a mis primos(as) y sobrinos(as), por brindarme fuerza en los momentos de flaqueza. También, esta dedicado a la memoria de mi tía Georgina y papá Bacilio por el cariño especial que tuvieron con mi persona.

Agradecimiento

Hay muchas personas a las que debo agradecerles por contribuir a los cinco maravillosos años de mi experiencia como estudiante de pregrado. Sin embargo, primero quiero agradecer a Dios por permitirme llegar hasta esta etapa; por darme salud y por darme la maravillosa familia que tengo.

Segundo, debo agradecer a mi amigo y co-asesor John Edgar Vargas Muños, quien, a través de más de 1000 correos electrónicos y muchas reuniones durante un año, moldeó un estudiante ansioso de investigación. Estoy muy agradecido con John Vargas por enseñarme no solo sobre algunos aspectos de la visión computacional o aprendizaje de máquina, sino también por enseñarme a redactar papers.

Tercero, agradezco al Dr. Lauro Enciso y a la M.Eng. Gladys Cutipa, quienes, a través de sus consejos y asesorías lograron que mi persona se interese por desarrollar un trabajo a largo plazo, haciendo que este sirva como un trabajo iniciación científica.

Finalmente, agradezco a todos mis docentes, quienes, me formaron como un académico competente, a través de los diferentes cursos y seminarios desarrollados durante mi vida universitaria.

Resumen

Actualmente existen muchas técnicas de aprendizaje de máquina efectivas durante la tarea de clasificación. Sin embargo, existe la necesidad de identificar que técnica destaca por encima del resto. Por consiguiente, es necesario evaluar un conjunto de técnicas de aprendizaje de máquina en un escenario desafiante. En particular, el escenario seleccionado en este trabajo corresponde al reconocimiento de edificios a partir de imágenes; el reconocimiento de edificios es una tarea difícil, ya que las imágenes pueden tomarse desde diferentes ángulos, bajo diferentes condiciones de iluminación y un desafío adicional es diferenciar edificios visualmente similares (por ejemplo, imágenes de iglesias). La mayoría de métodos de reconocimiento de edificios utilizan descriptores locales de la imagen para la extracción de características (es decir, luego de aplicar SIFT o SURF, se aplica un clasificador). No obstante, este método presenta una precisión limitada. Por lo tanto, es necesario evaluar técnicas de aprendizaje de máquina que resuelvan este tipo de problemas de una manera más precisa. Se propone evaluar técnicas de aprendizaje de máquina como Support Vector Machine (SVM), Random Forest (RF), Neuronal Network (NN) y K-Nearest Neighbod (KNN), a partir de métodos basados en Bag-of-Words y Redes Neuronales Convolucionales, para obtener vectores de características efectivos y realizar un reconocimiento de edificios preciso. Por último, se espera que los resultados permitan una mejor comprensión de las técnicas de aprendizaje de máquina aplicado al problema del reconocimiento de edificios de la ciudad del Cusco.

Palabras claves— Aprendizaje de Máquina, Redes Neuronales Convolucionales, Reconocimiento de Edificios, Bag-of-Words.

Abstract

Currently there are many effective machine learning techniques during the classification task. However, there is a need to identify which technique stands out above the rest. Therefore, it is necessary to evaluate a set of machine learning techniques in a challenging scenario. In particular, the scenario selected in this work corresponds to building recognition from images; building recognition from images is a challenging task since pictures can be taken from different angles and under different illumination condition and an additional challenge is to differentiate buildings with a similar architectural design(for example, Church images). Most building recognition methods use local image descriptors for feature extraction (that is, after applying SIFT or SURF, a classifier is applied). However, this technique have limited accuracy. Therefore, it is necessary to evaluate machine learning techniques to solve these types of problems in a more precise way. It is proposed to evaluate machine learning techniques such as Support Vector Machine (SVM), Random Forest (RF), Neuronal Network (NN) and K-Nearest Neighbors (KNN), based on Bag-of-Words and Neural Networks Convolutions, that obtain effective feature vectors to perform accurate classification of buildings. Finally, it is expected that the results allow a better understanding of the machine learning techniques applied to the problem of buildings recognition of city of Cusco.

Keywords— Machine Learning, Convolutional Neural Network, Building Recognition, Bag-of-Words.

Índice general

I Generalidades **1**

1. Aspectos Generales **2**
1.1. Problema de Investigación 2
1.1.1. Descripción del Problema 2
1.1.2. Formulación del Problema 4
1.2. Antecedentes 5
1.2.1. Estilo arquitectónico 5
1.2.2. Reconocimiento visual de lugares 7
1.2.3. ImageNet: Reconocimiento Visual a Gran Escala 8
1.3. Justificación 10
1.4. Objetivos 10
1.4.1. Objetivo General 10
1.4.2. Objetivos Específicos 10
1.5. Alcances y Limitaciones 11
1.6. Metodología 11
1.7. Cronograma de Actividades 12

II Marco Teórico **14**

2. Marco Conceptual **15**
2.1. Visión Computacional 15
2.1.1. Desafíos en Visión Computacional 16
2.1.1.1. Reconocimiento o Clasificación 16
2.1.1.2. Detección 16
2.1.1.3. Segmentación 17
2.2. Bag-of-Words 17
2.2.1. Extracción de Características 19
2.2.1.1. Scale-invariant feature transform (SIFT) 19
2.2.1.2. Speeded up robust features (SURF) 20
2.2.2. Codebook 21
2.3. Aprendizaje de Máquina 22
2.3.1. Support Vector Machine 23
2.3.2. Random Forest (RF) 26
2.3.3. K-Nearest Neighbors (kNN) 26
2.3.4. K-Means 28
2.4. Redes Neuronales Artificiales 29
2.4.1. Modelo computacional de una neurona 29

2.4.2. Arquitectura de una Red Neuronal . . . 30
2.4.3. Función de Activación . . . 32
2.4.4. Aprendizaje . . . 33
2.4.4.1. Backpropagation . . . 33
2.4.4.2. Descenso de Gradiente Estocástico . . . 35
2.4.4.3. Adam Optimization . . . 36
2.5. Deep Learning . . . 37
2.5.1. Red Neuronal Convolucional . . . 37
2.5.1.1. Capa de Convolución . . . 37
2.5.1.2. Pooling . . . 40
2.5.1.3. Capa Fully-Connected . . . 41
2.6. ImageNet Challenge . . . 42
2.6.1. Arquitecturas propuestas . . . 42
2.6.1.1. AlexNet . . . 42
2.6.1.2. Vgg Net . . . 43
2.6.1.3. GoogleNet . . . 45
2.6.1.4. Inception-V3 . . . 46
2.7. Transfer Learning . . . 47

III Desarrollo del Proyecto 49

3. Método Propuesto 50
3.1. Descripción de las fases . . . 50
3.1.1. Entrenamiento y construcción del modelo . . . 50
3.1.1.1. *Bag-of-Words* . . . 51
3.1.1.1.1. Extracción de Características . . . 51
3.1.1.1.2. Codebook . . . 52
3.1.1.1.3. Representación de la Imagen . . . 54
3.1.1.2. *Redes Neuronales Convolucionales (CNN)* . . . 55
3.1.1.2.1. Extracción de Características: VGG-16 . . . 58
3.1.1.2.2. Extracción de Características: VGG-19 . . . 59
3.1.1.2.3. Extracción de Características: Inception-V3 60
3.1.1.3. Clasificación . . . 61
3.1.1.3.1. Clasificación utilizando Support Vector Machine . . . 61
3.1.1.3.2. Clasificación utilizando Random Forest . . . 63
3.1.1.3.3. Clasificación utilizando K-Nearest Neighbors . . . 63
3.1.1.3.4. Clasificación utilizando Neural Network . . . 63
3.1.2. Predicción . . . 64

IV Proceso Experimental 66

4. CuscoBID y Experimentos 67
4.1. Cusco Building Image Dataset (CuscoBID) . . . 67
4.2. Proceso Experimental . . . 69
4.2.1. Distribución de la Base de Datos . . . 69

4.2.2. Entrenamiento y Construcción del modelo 71
4.2.2.1. *Bag-of-Words (BoW)* 71
4.2.2.2. *Redes Neuronales Convolucionales (CNN)* 128
4.2.3. Predicción 141

V Resultados **143**

5. Resultados e interpretaciones **144**
5.1. Detalles Técnicos 148
5.1.1. Hardware 148
5.1.2. Software 149

Conclusiones **150**

Trabajos Futuros **153**

Apéndice A. Publicación **154**

Bibliografía **159**

Publicaciones

- **Jeanfranco David Farfan-Escobedo**, Lauro Enciso-Rodas and John Edgar Vargas-Muñoz. Towards accurate building recognition using convolutional neural networks, *IEEE XXIV International Congress on Electronics, Electrical Engineering and Computing (INTERCON)*, Cusco, Perú. Agosto, 2017. IEEE.
- **Jeanfranco David Farfan-Escobedo** y Jose Luis Flores-Campana. Un sistema de aprendizaje profundo aplicado al reconocimiento de gestos estáticos de la mano, *IX Congreso Internacional de Computación y Telecomunicaciones (COMTEL)*, Lima, Perú. Octubre, 2017.

Índice de figuras

1. Capturas de diferentes puntos de vista de dos edificio histórico diferentes. . . . 3
2. Edificios visualmente similares. De izquierda a derecha, Iglesia de San Francisco, Templo de la Merced e Iglesia de Santo Domingo. . . 3
3. Ejemplos de oclusión de algunos edificios históricos de la ciudad del Cusco. . . . 4
4. De izquierda a derecha. Problemas de iluminación, ruido y resolución. 4
5. Diagrama de Gantt del cronograma de actividades. . . . 13
6. Áreas que contribuyen al desarrollo de la Visión Computacional. . . 15
7. Ejemplos de clasificación de imágenes. . . . 16
8. Detección de objetos en una imagen. . . . 16
9. Regiones segmentadas de una imagen. . . . 17
10. Ejemplos de texto. . . . 17
11. Palabras representativas. . . . 18
12. Generación de histogramas y clasificación. . . . 18
13. Detección y descripción de puntos de interés utilizando el algoritmo SIFT. . . . 19
14. Detección y descripción de puntos de interés utilizando el algoritmo SURF. . . . 20
15. Descriptores generados por SIFT, sirven como entrada del *Codebook*. 21
16. Construcción del *Codebook* y generación de las palabras visuales. . . 22
17. Separación de dos clases utilizando SVM. . . . 23
18. Detalles de los parámetros de sepación de dos clases utilizando SVM. 24
19. Resultados de aplicar *Random Forest* a diferentes clases. . . . 27
20. Diagrama de Voronoi. . . . 27
21. Aplicación de la técnica *k-means* sobre puntos en $2D$. . . . 28
22. Modelo de neurona porpuesta por McCulloch y Pitts. . . . 29
23. Diferentes tipos de función de activación. . . . 30
24. Arquitectura básica de una red neuronal. . . . 30
25. Arquitecturas de red neuronal propuestas hasta la actualidad. . . . 31
26. Función de activación Sign, Sigmoide y tanh. . . . 32
27. Función de activación ReLu. . . . 33
28. Secuencia de pasos de la fase Forward Propagation. . . . 34
29. Secuencia de pasos de la fase Backward. . . . 35
30. Mínimo local y mínimo global. . . . 36
31. Desafíos durante la asignación del índice de aprendizaje. . . . 36
32. Arquitectura de red neuronal convolucional desarrollada en 1998. . 37
33. Valores de salida de una operación de convolución. . . . 38

34. Ejemplos de Stride y Padding. 39
35. Aplicación de la operación convolución sobre una imagen. 39
36. Aplicación de la función de activación Relu. 39
37. Ejemplos de max y average pooling. 40
38. Aplicación de *Max Pooling* y *Sum Pooling*. 41
39. Aplicación de la capa Fully-Conneted. 42
40. Arquitectura de red neuronal convolucional *AlexNet*. 43
41. Concatenación de las convoluciones del módulo *inception*. 45
42. Arquitectura GoogleNet y módulos *inception*. 46
43. Nuevo módulo *Inception-V3*. 47
44. Visualización de los datos de las capas convolucionales. 48

45. Entrenamiento y construcción del modelo utilizando *Bag-of-Words*. 51
46. Descriptores de características de una imagen. 52
47. Vectores de características de una imagen. 52
48. Construcción del codebook utilizando K-means. 53
49. Ejemplos de la aplicación de la técnica *kNN*. 54
50. Modelo Bag-of-Words desarrollado paso a paso. 55
51. Top-1 de redes neuronales convolucionales. 56
52. Transfer Learning utilizando los modelos VGG-16, VGG-19 e Inception-V3. 57
53. Aplicación de la técnica data augmentation. 57
54. Vector de características extraído a partir del modelo VGG-16. . . . 59
55. Vector de características extraído a partir del modelo VGG-19. . . . 59
56. Vector de características extraído a partir del modelo Inception-V3. 61
57. Variación del parámetro C de la técnica SVM. 62
58. Variación de los parámetros C y γ de la técnica SVM. 62
59. Arquitectura propuesta de red neuronal. 64
60. Arquitectura propuesta para identificar la etiqueta de una imagen externa a la base de datos. 65

61. Algunas imágenes de la segunda versión de *Cusco Building Image Dataset*. 68
62. Distribución de la base de datos. 69
63. Ejemplo de Matriz de Confusión. 70
64. Promedio de los resultados de aplicar SVM con $kernel_{\mathbf{linear}}$, C equivalente a 100 y el extractor de características SIFT. 72
65. Promedio de los resultados de aplicar SVM con $kernel_{\mathbf{linear}}$, C equivalente a 100 y el extractor de características SURF. 74
66. Promedio de los resultados de aplicar SVM con $kernel_{\mathbf{rbf}}$, C y γ equivalentes a 100 y 10 respectivamente, y el extractor de características SIFT. 76
67. Promedio de los resultados de aplicar SVM con $kernel_{\mathbf{rbf}}$, C y γ equivalentes a 100 y 10 respectivamente, y extractor de características SURF. 78
68. Promedio de los resultados de aplicar Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente, y un extractor de características SIFT. 80

69. Promedio de los resultados de aplicar Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente, y un extractor de características SURF. 82
70. Promedio de los resultados de aplicar k-Nearest Neighbor con $k = 1$ y un extractor de características SIFT. 84
71. Promedio de los resultados de aplicar k-Nearest Neighbor (kNN) con $k = 1$ y un extractor de características SURF. 86
72. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF, sobre un tamaño de *codebook* (Cod_{size}) equivalente a 300. 91
73. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF con un tamaño de *codebook* (Cod_{size}) equivalente a 500. 95
74. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF con un tamaño de *codebook* (Cod_{size}) equivalente a 700. 99
75. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando los extractor de características SIFT y SURF, con un tamaño de *codebook* (Cod_{size}) equivalente a 900. 103
76. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF, con un tamaño de *codebook* (Cod_{size}) equivalente a 1100. 107
77. Resultados de aplicar *Neuronal Network* durante 15000 iteraciones, los extractores de características SIFT y SURF con un tamaño de *codebook* (Cod_{size}) equivalente a 1300. 111
78. Resultados más elevados de SVM, RF y kNN basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT. 112
79. Resultados más elevados de SVM, RF y kNN basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF. 113
80. Resultados más elevados de Neural Network basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones. 114
81. Resultados más elevados de Neural Network basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones. 115
82. Resultados más elevados de SVM, RF y kNN basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT. 116
83. Resultados más elevados de SVM, RF y kNN basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF. 117

84. Resultados más elevados de Neural Network basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones. 118
85. Resultados más elevados de Neural Network basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones. 119
86. Resultados más elevados de SVM, RF y kNN basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT. 120
87. Resultados más elevados de SVM, RF y kNN basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF. 121
88. Resultados más elevados de Neural Network basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones. 122
89. Resultados más elevados de Neural Network basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones. 123
90. Resultados más elevados de SVM, RF y kNN basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT. 124
91. Resultados más elevados de SVM, RF y kNN basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF. 125
92. Resultados más elevados de Neural Network basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones. 126
93. Resultados más elevados de Neural Network basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones. 127
94. Visualización de las meétricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo VGG-16. Utilizando SVM, RF y kNN. 130
95. Visualización de las métricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo VGG-16, basado en Neural Network. 131
96. Visualización de las métricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo VGG-19. 132
97. Visualización de las métricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo VGG-19, basado en Neural Network. 135

98. Visualización de las métricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo Inception-V3. . . 136
99. Visualización de las métricas de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score*, obtenidas utilizando el modelo Inception-V3, basado en la técnica de aprendizaje de máquina Neural Network. . . . 139
100. Data augmentation aplicado sobre una imagen de entrenamiento. . 140
101. Éxito de la predicción de un edificio histórico de la ciudad del Cusco.141
102. Error de la predicción de un edificio histórico de la ciudad del Cusco.142

103. Comparación de los mejores resultados de la métrica Accuracy, Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words. 145
104. Comparación de mejores resultados de métrica Recall. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words. 145
105. Comparación de mejores resultados de métrica Precisión. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words. 146
106. Comparación de mejores resultados de métrica F1_Score. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words. 146

Índice de tablas

1. Resumen de trabajos enfocados a la clasificación basada en estilos arquitectónicos. 6
2. Resumen de trabajos enfocados al reconocimiento visual de lugares. 8
3. Trabajos con mejores resultados desarrollados durante *Imagenet Large-Scale Visual Recognition Challenge (ILSVRC)* 9

4. Arquitectura detallada *AlexNet*. 43
5. Configuración de las Arquitecturas VGG. 44

6. Arquitectura VGG-16 utilizada para el Transfer Learning, la función de activación ReLU no se muestra para mayor brevedad. Donde FM, FCONV y FPOL representa los *feature maps*, el tamaño de filtros de convolución y el tamaño de los filtros de *pooling* respectivamente. 58
7. Arquitectura VGG-19 utilizada para el Transfer Learning, la función de activación ReLU no se muestra para mayor brevedad. Donde FM, FCONV y FPOL representa los *feature maps*, el tamaño de filtros de convolución y el tamaño de los filtros de *pooling* respectivamente. 60
8. Optimización de los parámetros de SVM. 63
9. Asignación de los parámetros para Random Forest. 63

10. Detalles de la primera versión de *Cusco Building Image Dataset*. . . 68
11. Detalles de la segunda versión de *Cusco Building Image Dataset*. . . 68
12. Resumen de los resultados a partir de la aplicación del SVM con un $kernel_{\mathbf{linear}}$ con C equivalente a 100 y el extractor de características SIFT. 71
13. Resumen del promedio y desviación estándar a partir de la aplicación del SVM con $kernel_{\mathbf{linear}}$, C equivalente a 100 y el extractor de características SIFT. 72
14. Resumen de los resultados a partir de la aplicación del SVM con $kernel_{lineal}$ con C equivalente a 100 y extractor de características SURF. 73
15. Resumen del promedio y desviación estándar a partir de la aplicación del SVM con $kernel_{\mathbf{linear}}$, C equivalente a 100 y el extractor de características SURF. 74
16. Resumen de los resultados a partir de la aplicación del SVM con $kernel_{rbf}$ con C y γ equivalentes a 100 y 10 respectivamente y extractor de características SIFT. 75

17. Resumen del promedio y desviación estándar a partir de la aplicación del SVM con $kernel_{\mathbf{rbf}}$, C y γ equivalentes a 100 y 10 respectivamente y extractor de características SIFT. 76
18. Resumen de los resultados a partir de la aplicación del SVM con $kernel_{rbf}$ con C y γ equivalentes a 100 y 10 respectivamente y extractor de características SURF. 77
19. Resumen del promedio y desviación estándar a partir de la aplicación del SVM con $kernel_{\mathbf{rbf}}$, C y γ equivalentes a 100 y 10 respectivamente y extractor de características SURF. 78
20. Resumen de los resultados a partir de la aplicación del Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SIFT. 79
21. Resumen del promedio y desviación estándar a partir de la aplicación de Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SIFT. 80
22. Resumen de los resultados a partir de la aplicación del Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SURF. 81
23. Resumen del promedio y desviación estándar a partir de la aplicación de Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SURF. 82
24. Resumen de los resultados a partir de la aplicación de k-Nearest Neighbor con $k = 1$ y un extractor de características SIFT. 83
25. Resumen del promedio y desviación estándar a partir de la aplicación de k-Nearest Neighbor con $k = 1$ y un extractor de características SIFT. 84
26. Resumen de los resultados a partir de la aplicación de k-Nearest Neighbor con $k = 1$ y un extractor de características SURF. 85
27. Resumen del promedio y desviación estándar a partir de la aplicación de k-Nearest Neighbor con $k = 1$ y un extractor de características SURF. 86
28. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 300. 88
29. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 300. 89
30. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 300. 90

31. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network (NN)* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 300. 90
32. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 500. 92
33. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 500. 93
34. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 500. 94
35. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 500. 95
36. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 700. 96
37. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 700. 97
38. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 700. 98
39. Promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 700. 99
40. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 900. 100
41. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 900. 101
42. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 900. 102

43. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 900. 103
44. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 1100. 104
45. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 1100. 105
46. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 1100. 106
47. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 1100. 107
48. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 1300. 108
49. Resumen de los resultados a partir de la aplicación de *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 1300. 109
50. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de *codebook* (Cod_{size}) equivalente a 1300. 110
51. Resumen del promedio y desviación estandar a partir de aplicar *Neuronal Network* durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de *codebook* (Cod_{size}) equivalente a 1300. 111
52. Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre los extractores de características SIFT y SURF. 112
53. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT, durante 15000 iteraciones. 113
54. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Accuracy*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF, durante 15000 iteraciones. 114

55. Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre los extractores de características SIFT y SURF. 116
56. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT, durante 15000 iteraciones. 117
57. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Recall*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF, durante 15000 iteraciones. 118
58. Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre los extractores de características SIFT y SURF. 120
59. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT, durante 15000 iteraciones. 121
60. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *Precisión*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF, durante 15000 iteraciones. 122
61. Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* sobre los extractores de características SIFT y SURF. 124
62. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SIFT, durante 15000 iteraciones. 125
63. Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica *F1_Score*, al utilizar *Bag-of-Words* con diferentes tamaños del *codebook* y SURF, durante 15000 iteraciones. 126
64. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN. 128
65. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN. 128
66. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando *Neural Network* como técnica de aprendizaje de máquina. 129

67. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina *Neural Network.* 130
68. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN 132
69. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN. 132
70. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando *Neural Network* como técnica de aprendizaje de máquina 133
71. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina *Neural Network.* 134
72. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN 136
73. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN 136
74. Resumen de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando *Neural Network* como técnica de aprendizaje de máquina. 137
75. Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina *Neural Network.* 138
76. Promedio y desviación estandar de los resultados de aplicar data augmentation sobre las carpetas Split. Utilizando el modelo preentrenado Inception-V3. 140
77. Comparación de los mejores resultados generados por las técnicas de aprendizaje de máquina Support Vector Machine, Random Forest, Neural Network y k Nearest Neighbor. Utilizando Inception-V3 (con y sin data augmentation) y Bag-of-Words. 144

Parte I

Generalidades

Capítulo 1

Aspectos Generales

1.1. Problema de Investigación

1.1.1. Descripción del Problema

El reciente éxito de aplicaciones que utilizan técnicas de aprendizaje de máquina genera una necesidad, identificar que técnica destaca por encima del resto. Sin embargo, la verdadera dificultad de estas técnicas radica en el escenario de prueba. En particular, en este trabajo el escenario seleccionado es el reconocimiento de imágenes de edificios históricos de la ciudad del Cusco.

Por otro lado, el Perú es un país con una gran cantidad de sitios arqueológicos[1], según voceros oficiales del Ministerio de Comercio Exterior y Turismo[2], el Perú recibió la visita de 3.4 millones de turístas internacionales en el año 2016. Por lo tanto, existe la necesidad de manejar la información generada[3], a partir de nuevas herramientas tecnológicas.

En particular, para identificar imágenes de edificios históricos, es necesario realizar un proceso de clasificación[4]. Así mismo, las particularidades a la hora de capturar edificios históricos de la ciudad del Cusco representan un desafío complicado. Por ejemplo, un edificio puede ser capturado desde diferentes posiciones y ángulos, dependiendo de la ubicación del observador como se muestra en la Figura 1, donde las imágenes corresponden a capturas de la parte frontal, lateral y posterior de los edificios históricos Cristo Blanco e Inca Pachacutec.

De igual modo, uno de los desafios más importantes es diferencias edificios históricos visualmente similares (por ejemplo, imágenes de iglesias). Un estudio realizado por [Viñuales, 2004] demuestra que los estilos arquitectónicos de las

[1]Según declaraciones del Ministro de Cultura, existen alrededor de 13 mil sitios arqueológicos registrados. Sin embargo, existen muchos otros sin reconocer. Ver 29'39" *https://goo.gl/xxGGLn*.

[2]Esta cifra fue alcanzada durante los 11 primeros meses del 2016. Ver en *https://goo.gl/xxDFLn*.

[3]En los últimos 20 años, los datos han aumentado a gran escala en diversos campos. Según un informe de la Corporación Internacional de Datos (IDC), en 2011, el volumen total de datos creados y copiados en el mundo fue de 1.8ZB (10^{21} B), que aumentó casi nueve veces en cinco años [Chen et al., 2014], mientras que en [McAfee et al., 2012] se menciona que a partir del 2012, alrededor de 2,5 exabytes de datos se crean cada día, y ese número se duplica cada 40 meses más o menos.

[4]La clasificación, es una de las tareas de toma de decisiones más frecuentes de la actividad humana. Un problema de clasificación ocurre cuando un objeto necesita ser asignado a un grupo o clase predefinido [Zhang, 2000].

Figura 1: *Capturas de diferentes puntos de vista de dos edificio histórico diferentes.*



Fuente: *Elaboración propia*

iglesias de la ciudad del Cusco son muy similares. Por lo tanto, algunas imágenes de iglesias de diferentes clases[5] son semejantes como se muestra en la Figura 2.

Figura 2: *Edificios visualmente similares. De izquierda a derecha, Iglesia de San Francisco, Templo de la Merced e Iglesia de Santo Domingo.*



Fuente: *Elaboración propia*

Así mismo, muchos eventos son realizados en la ciudad del Cusco tales como el Inti Raymi[6], Corpus Christi[7] y otros; estos eventos generan oclusiones. Es decir, objetos transitorios(personas, vehículos y otros) ocultan la imagen de análisis. Por lo general, los edificios históricos sirven de *background* de diversos eventos como se muestra en la Figura 3.

También, para incrementar el nivel de dificultad del escenario de prueba, se considera los típicos problemas de Visión Computacional. Por ejemplo, el cambio de iluminación, ruido y resolución (Figura 4).

[5]Basado en el concepto de Programación Orientada a Objetos, en Visión Computacional, en específico en este problema una clase hace referencia a un conjunto de imágenes que comparten características de un determinado edificio histórico.

[6]Según [Vega and Palomino, 2003] el Inti Raymi es la principal festividad del pueblo cusqueño, tiene como objetivo el reconocimiento y gratitud a la deidad máxima, el Sol.

[7]Festividad que consiste de un recorrido sacro de imágenes religiosas [Viñuales, 2004].

Figura 3: *Ejemplos de oclusión de algunos edificios históricos de la ciudad del Cusco.*



Fuente: *Elaboración propia*

Figura 4: *De izquierda a derecha. Problemas de iluminación, ruido y resolución.*



Fuente: *Elaboración propia*

Finalmente, existen diferentes técnicas y métodos para resolver este tipo de problemas como se muestra en la Tabla 1, Tabla 2 y Tabla 3. Siendo los métodos *Bag-of-words* y *Redes Neuronales Convolucionales* los más utilizados para representar una imagen. Así mismo, para verificar que técnica presenta un mejor rendimiento durante el proceso de clasificación de los datos, se utiliza un conjunto de técnicas de aprendizaje de máquina; siendo la mejor técnica, aquella que presente los mejores resultados durante la evaluación de las métricas de aprendizaje.

1.1.2. Formulación del Problema

¿Cuál es la técnica de aprendizaje de máquina más óptima en un escenario de identificación de edificios históricos de la ciudad del Cusco, basado en los métodos *Bag-of-words* y *Redes Neuronales Convolucionales*?.

1.2. Antecedentes

La identificación de edificios a partir imágenes se puede aplicar a diversas áreas como el estilo arquitectónico, reconocimiento visual de lugares, visión de robots, localización y otros. A continuación se presenta un conjunto de trabajos relacionados que describen algunos de estos casos.

1.2.1. Estilo arquitectónico

El estilo arquitectónico es un problema muy complejo. Debido a que cada estilo define ciertas formas, reglas de diseño y técnicas para la construcción de una determinada edificación. Muchos trabajos abordan este problema atacando estilos arquitectónicos específicos. Es necesario indicar que estilo arquitectónico se considera una tarea de clasificación de elementos arquitectónicos separados [Shalunts et al., 2012b]. Es decir, se realiza la clasificación de las ventanas, torres, cúpulas y otros elementos de forma independiente.

En [Shalunts et al., 2011] se aborda este problema a través de la clasificación de ventanas con diferentes estilos arquitectónicos como el románico, gótico y barroco. Obtuvo una tasa de precisión igual a 95.16 %.

- Se creó una base de datos de 400 imágenes, donde 90 imágenes representan al conjunto de entrenamiento y el resto al conjunto de prueba. Así mismo, el rango de resolución de las imágenes es de 138×93 a 4320×3240 píxeles.
- Para extraer los puntos de interés y descriptores se utilizó *SIFT*. Además, se utilizó el método *Bag-of-Words*, para luego generar el *codebook* a través del algoritmo *k-means*. La clasificación se determina a partir del máximo pico de cada histograma generado por imagen.
- Es necesario indicar que las imágenes analizadas solo contenían ventanas con diferentes estilos arquitectónicos.

En [Shalunts et al., 2012c] se realizó una segmentación de diferentes edificios eclesiásticos como iglesias, mezquitas y otros, para posteriormente realizar una detección de cúpulas con una precisión media de 95.45 %. Estas imágenes sirven de entrada para [Shalunts et al., 2012a], en este trabajo se aborda el problema de la clasificación de cúpulas identificando estilos arquitectónicos como el renacimiento, ruso e islámico. Obtuvo una tasa de precisión igual a 90.24 %.

- Se analizó la altura y el ancho de la cúpula para identificarla como islámica. Por otro lado, el color dorado representaba con mayor probabilidad a las cúpulas rusas. Por último, se analizó la forma de la cúpula representando con mayor probabilidad al renacimiento, a través de la agrupación y aprendizaje de las características locales.
- La técnica de extracción de características y representación de la imagen es igual a [Shalunts et al., 2011]. Es decir, utiliza de descriptor y detector a *SIFT* y para representar la imagen utiliza *Bag-of-Words*.

De forma similar, en [Shalunts, 2015b] se realizó una segmentación y posterior detección de torres de diferentes edificios históricos con una precisión media igual a 96 %. Estos elementos detectados sirven de entrada para [Shalunts, 2015a], en este trabajo se realiza una clasificación de torres a partir de los estilos arquitectónicos románico, gótico y barroco. Obtuvo una tasa de precisión igual a 80.54 %.

- Se realizó una extracción de características a partir del algoritmo *SIFT*. Por otro lado para construir el *codebook* se varió el número de centroides del algoritmo *k-means* a 40 por cada clase.

- La diferencia de este trabajo es el uso del clasificador *KNN*. Sin embargo, cuando k aumenta su valor, la tasa de precisión disminuye. Por último, este trabajo desarrolla una base de datos de imágenes disponible en *Flicker*[8] que incluye 325 imágenes de 35 diferentes edificios.

Finalmente, el resumen de todos estos trabajos se muestra en la Tabla 1.

Tabla 1: *Resumen de trabajos enfocados a la clasificación basada en estilos arquitectónicos.*

Nombre	Architectural style classification of building facade windows.
Autor / Año	Shalunts Gayane, Haxhimusa Yll y Sablatnig Robert / 2011
Objetivo	Clasificar estilos arquitectónicos de imágenes de fachadas de ventanas.
Precisión	95.16 %
Conclusión	Se propone un nuevo enfoque de clasificación basado en *Bag-of-words*.
Nombre	Segmentation of building facade domes.
Autor / Año	Shalunts Gayane, Haxhimusa Yll y Sablatnig Robert / 2012
Objetivo	Segmentar e identificar fachadas de cúpulas de edificios históricos.
Precisión	95.45 %
Conclusión	Un nuevo enfoque de segmentación basados en gráficos y análisis de imágenes.
Nombre	Architectural style classification of domes.
Autor / Año	Shalunts Gayane, Haxhimusa Yll y Sablatnig Robert / 2012
Objetivo	Clasificar estilos arquitectónicos de imágenes de fachadas de cúpulas.
Precisión	90.24 %
Conclusión	Se clasifica sumando los histogramas generados por *Bag-of-words*
Nombre	Segmentation of Building Facade Towers.
Autor / Año	Shalunts Gayane / 2015
Objetivo	Segmentar e identificar fachadas de torres de edificios históricos.
Precisión	96 %
Conclusión	Además de incluir el enfoque de [Shalunts et al., 2012c], se propone un procesamiento de imágenes
Nombre	Architectural Style Classification of Building Facade Towers
Autor / Año	Shalunts Gayane / 2015
Objetivo	Clasificar estilos arquitectónicos de imágenes de fachadas de torres.
Precisión	80.54 %
Conclusión	El uso del clasificar *KNN* mejora los resultados del modelo *Bag-of-words*.

Fuente: *Elaboración propia*

[8]La base de datos de imágenes puede ser descargada de: `https://www.flickr.com/photos/lady_photographer/sets/72157636149550844/`

1.2.2. Reconocimiento visual de lugares

El reconocimiento visual de lugares se centra en identificar edificios, calles, avenidas y carreteras. En [Lowry et al., 2016] se realiza una descripción de las técnicas locales y globales más utilizadas para abordar esta tarea, siendo los detectores y descriptores *SITF* [Lowe, 2004] y *SURF* [Bay et al., 2006] las técnicas más utilizadas a la hora de extraer características de una imagen. Mientras que el modelo *Bag-of-Word* se utiliza para aumentar la eficacia al cuantificar las características locales, Finalmente, para evitar falsos positivos y aumentar la tasa de acierto a la hora de clasificar las imágenes se utiliza un *module mapping* [Lowry et al., 2016]. Es decir, un sistema *geoetiquetado* que identifique las coordenadas de geoposicionamiento de una imagen.

En [Groeneweg et al., 2006] se desarrollo un algoritmo basado en el extractor de características SIFT que permite el reconocimiento de edificios, a partir del uso de un telefono móvil, a pesar de la limitada capacidad de procesamiento y almacenamiento esta técnica obtuvo un 80 % de precisión en la base de datos Zúrich(ZuBuD).

- Por la capacidad de los teléfonos móviles de esa época, se bajo la resolución de las imágenes de ZuBuD a 160×120.
- La clasificación de una imagen se realiza mediante un sistema de votación por mayoría, en la que cada región dentro de la imagen de consulta vota por el edificio perteneciente a su vecino más cercano en el espacio de componente principal. El peso del voto es igual a $1/(d + c)$, donde d es la distancia euclidiana al vecino más cercano y c es una pequeña cantidad constante.
- El algoritmo se ejecuta en varios segundos, mientras que requiere alrededor de 10 KB de memoria para representar un solo edificio dentro de la base de datos local.
- Finalmente, estos resultados son elevados por la poca variabilidad que existe dentro de la base de datos ZuBuD.

Por otro lado, en [Torii et al., 2013] se realiza una clasificación de estructuras repetidas como construcciones de cercas, fachadas o marcas viales de carreteras. Obtuvo una precisión de 73.83 % sobre *Pittsburgh Dataset*.

- Este trabajo se propone una comparación del modelo *Fisher vector* y las modificaciones hechas al modelo *Bag-of-Words*, en específico compran su modelo propuesto denominado *adaptive soft-assignment approach (AA thr-idf)* frente a otros.
- En ese trabajo se demuestra que las estructuras repetidas no son una molestia, pero deben estar representadas apropiadamente.
- Esto se logra gracias a que la recuperación se basa en la detección robusta de estructuras de imagen repetidas y una modificación adecuada de pesos en el modelo *Bag-of-Words*.
- Se utilizan múltiples ocurrencias de elementos repetidos para proporcionar una asignación suave natural de entidades a *Bag-of-Words*.

El método propuesto por [Bezak, 2016] utiliza el modelo de *red neural convolucional* LeNet-5 [LeCun et al., 1998] para identificar edificios. El autor utiliza este modelo ya que entrenar esta red neuronal desde cero requiere relativamente bajos recursos computacionales.

- Ese modelo se basó en un conjunto de 460 imágenes de entrenamiento y 140 de validación. Entrenando su modelo desde cero. Sin embargo, existen otras alternativas, como el *Transfer Learning*, que puede usar un modelo de CNN más complejo.
- El modelo se aplicó a edificios históricos de la ciudad de Trava, Consiguiendo una tasa de precisión alta. Sin embargo, se menciona que las imágenes no presentan mucha variabilidad.
- Los errores durante el proceso de clasificación detectados fueron ocasionados por la falta de nitidez e iluminación.
- Finalmente, el modelo se ejecuto sobre una GPU y obtuvo una precisión equivalente a 95 %.

Finalmente, el resumen de estos trabajos se muestran en la Tabla 2.

Tabla 2: *Resumen de trabajos enfocados al reconocimiento visual de lugares.*

Nombre	A fast offline building recognition application on a mobile telephone.
Autor	Groeneweg, N. and de Groot, B. and Halma, A. and Quiroga, B.
Año	2006
Objetivo	Clasificar imágenes de edificios de la base de datos de Zúrich(ZuDuB).
Precisión	80 %
Conclusión	Se propone una clasificación utilizando los recursos de los dispositivos móviles.
Nombre	Visual Place Recognition with Repetitive Structures.
Autor	Torii, Akihiko and Sivic, Josef and Pajdla, Tomas and Okutomi, Masatoshi
Año	2013
Objetivo	Clasificar imágenes de estructuras repetidas como cercas, fachadas y otros.
Precisión	73.83 %
Conclusión	Modificaron el modelo *Bag-of-Words* para demostrar que trabajar estructuras repetitivas no es un problema, solo si se extraen características correctamente.
Nombre	Building recognition system based on deep learning.
Autor	Bezak, Pavol
Año	2016
Objetivo	Clasificar imágenes de edificios de la ciudad de Trava.
Precisión	95 %
Conclusión	A partir de un modelo LeNet-5 entrenado desde cero, se consiguen buenos resultados con un costo computacional reducido.

Fuente: *Elaboración propia*

1.2.3. ImageNet: Reconocimiento Visual a Gran Escala

ImageNet[9], es una base de datos que contiene 14 millones de imágenes de alta resolución de 1000 categorías diferentes, sobre esta base de datos se desarrolla el

[9]La base de datos Imagenet se puede descargar de: `http://www.image-net.org`

concurso *Imagenet Large-Scale Visual Recognition Challenge (ILSVRC)*. Es necesario mencionar que esta base de datos obtiene los mejores resultados al utilizar arquitecturas basadas en redes neuronales convolucionales. Desde que la arquitectura de red neuronal convolucional AlexNet [Krizhevsky et al., 2012] obtuvo la tasa de error más baja equivalente a 16.4 % en la competencia ILSVRC 2012, modelos más complejos de redes neuronales convolucionales se han propuesto. Así mismo, en el ILSVRC 2013 el equipo de Clarify de la Universidad de New York desarrollo una mejora de la arquitectura anterior llamada ZF Net [Zeiler and Fergus, 2014] y obtuvo un error equivalente a 11.7 %. En la competencia ILSVRC 2014 la arquitectura CNN de Google [Szegedy et al., 2015] obtuvo el mejor rendimiento con una tasa de error equivalente a 6.7 %, superando al equipo de Oxford y sus CNNs VGG16 y VGG19 [Simonyan and Zisserman, 2014] con una tasa de error igual a 7.4 %. El éxito de GoogleNet muestra que los componentes de la CNN no necesariamente tienen que ser organizados secuencialmente. Por otro lado, en el ILSVRC 2015, el modelo ResNet [He et al., 2016] propuesto por Microsoft alcanzó los mejores resultados con una tasa de error equivalente a 3.57 % utilizando una CNN con 152 capas. Sin embargo, una versión mejorada de GoogleNet, llamada Inception-V3 [Szegedy et al., 2016] obtuvo un 3.5 % de error. Este modelo mejora los resultados obtenidos por GoogleNet maximizando el flujo de información en la red. Esto se hace construyendo redes que equilibren la profundidad y el ancho. El resumen de estos trabajos se muestra en la Tabla 3.

Tabla 3: *Trabajos con mejores resultados desarrollados durante Imagenet Large-Scale Visual Recognition Challenge (ILSVRC)*

Arquitectura / Año	Alexnet / 2012
Autor	Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E.
Nro de Capas / Error	8 *layers* / 16.4 %
Arquitectura / Año	ZF Net / 2013
Autor	Zeiler, Matthew D and Fergus, Rob
Nro de Capas / Error	8 *layers* / 11.7 %
Arquitectura / Año	VGG Net / 2014
Autor	Simonyan, Karen and Zisserman, Andrew
Nro de Capas / Error	16 *layers* / 7.4 %
Arquitectura / Año	GoogleNet / 2014
Autor	Szegedy, C. and Liu, W. and Jia, Y. and Sermanet, P.
Nro de Capas / Error	22 *layers* / 6.7 %
Arquitectura / Año	ResNet / 2015
Autor	He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing
Nro de Capas / Error	152 *layers* / 3.57 %
Arquitectura / Año	Inception-V3 / 2015
Autor	Szegedy, C. and Vanhoucke, V. and Ioffe, S. and Shlens, J.
Nro de Capas / Error	48 *layers* / 3.5 %

Fuente: *Elaboración propia*

1.3. Justificación

De los antecedentes revisados (ver Tabla 1, Tabla 2 y Tabla 3) se muestra que los métodos más utilizados al abordar problemas como el estilo arquitectónico, reconocimiento visual de lugares y el reconocimiento de objetos a partir de la base de datos *ImageNet* son *Bag-of-words* y *Redes Neuronales Convolucionales*. Así mismo, para identificar de forma precisa edificios históricos de la ciudad del Cusco existe la necesidad de comparar estos métodos. De forma similar, se utiliza un conjunto de métricas de aprendizaje *(Accuracy, Recall, Precisión y F1_Score)* para evaluar un grupo de técnicas de aprendizaje de máquina *(Support Vector Machine, Random Forest, k Nearest Neighbor y Neural Network)*, para identificar la técnica más adecuado durante esta tarea. Además, con la finalidad de apoyar a la comunidad científica de visión computacional, se recolecto una base de datos de imágenes de edificios históricos de la ciudad del Cusco; estas imágenes presentan desafíos como: rotación, condiciones de iluminación variada, capturas de diferentes ángulos, oclusiones y otros. De esta forma, esta es la primer base de datos de edificios históricos de la ciudad del Cusco que será utilizada con fines de investigación sin costo alguno como Zúrich(ZuBuD)[10] , Sheffield(SBID)[11] y Oxford(OBD)[12].

Por otro lado, aplicar técnicas de aprendizaje de máquina para resolver este tipo de problemas es muy importante, debido a que es una de las áreas de mayor crecimiento dentro de las ciencias de la computación. Por lo tanto, el documento servirá de base bibliográfica para futuras investigaciones en este campo.

Por último, se espera que la identificación precisa de edificios históricos de la ciudad del Cusco sirva de base para aplicaciones más complejas como la navegación de robots, vehículos autónomos, realidad aumentada y otros.

1.4. Objetivos

1.4.1. Objetivo General

Determinar la técnica de aprendizaje de máquina más óptima. Por medio de, la evaluación de técnicas de aprendizaje de máquina para la identificación de imágenes de edificios históricos de la ciudad del Cusco basado en *Bag-of-words* y *redes neuronales convolucionales*.

1.4.2. Objetivos Específicos

1. Recolectar fotografías que contengan imágenes de edificios históricos de la ciudad del Cusco.

2. Evaluar las imágenes recolectadas a partir de técnicas de aprendizaje de máquina basado en métodos de Bag-of-Words.

[10]La base de datos de edificios de Zúrich puede ser descargada de `http://www.vision.ee.ethz.ch/showroom/zubud/`

[11]La base de datos de edificios de Sheffield puede ser descargada de `https://www.sheffield.ac.uk/eee/research/iel/research/buildings_data`

[12]La base de datos de edificios de Oxford puede ser descargada de `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`

3. Evaluar las imágenes recolectadas a partir de técnicas de aprendizaje de máquina basado en métodos de Redes Neuronales Convolucionales.

4. Comparar los resultados obtenidos de los métodos basados en Bag-of-Words y Redes Neuronales Convolucionales.

5. Determinar la técnica de aprendizaje de máquina más óptima.

1.5. Alcances y Limitaciones

La construcción del conjunto de datos para la fase de entrenamiento y prueba exigía recolectar imágenes de edificios históricos de la ciudad del Cusco. Sin embargo, por el arduo trabajo que esta tarea representa, se limita el número de edificios históricos de interés a 14, estos son: La Casa del Inca Garcilaso de la Vega, La Catedral del Cusco, La Compañia de Jesús, El Coricancha, El Cristo Blanco, El Templo de la Merced, El Mural de Historial Inca, La Paccha de Pumaqchupan, La Pileta de San Blas, El Monumento del Inca Pachacutec, Sacsayhuaman, La Iglesia de San Francisco, La Iglesia de San Pedro y La Iglesia de Santo Domingo. La distribución de las imágenes por clase se muestra en la Tabla 11.

Así mismo, durante el proceso de clasificación de imágenes de edificios históricos de la ciudad del Cusco, se limita el número de técnicas de aprendizaje de máquina, estos son: *Support Vector Machine, Neural Network, K-Nearest Neighbors* y *Random Forest*. De forma similar, el número de métricas a utilizar se limita a 4, estas son: *Accuracy, Recall, Precisión y F1_Score*.

El eje principal de este trabajo se centra en utilizar diferentes métricas para evaluar técnicas de aprendizaje de máquina, a partir de los vectores de características generados por los métodos *Bag-of-Words* y *Redes Neuronales Convolucionales*. La visualización de los datos o una aplicación final de usuario que tenga en consideración aspectos relacionas a una interfaz, no están considerados en este trabajo.

Por último, la mayoría de las imágenes fueron recolectadas de internet. Sin embargo, fue necesario capturar imágenes manualmente con ruido, rotación, cambios de iluminación y otros problemas. De esta forma, se espera que imágenes externas a la base de datos sean reconocidas de forma eficiente.

1.6. Metodología

Para solucionar este problema se utiliza el método de investigación descriptivo[13] para recopilar y comparar información existente. Además, a partir de la identificación de la técnica de aprendizaje de máquina más óptima el problema adquiere una naturaleza aplicativa[14].

La secuencia de pasos a efectuar para desarrollar el presente trabajo es:

1. **Revisión de la literatura:** En esta fase se realiza un estudio de trabajos previos. Además, de una investigación de la teoría de visión computacional, aprendizaje de máquina, aprendizaje profundo y aprendizaje por transferencia. Esto permite una mejor comprensión de las siguientes etapas.

[13]Según el estudio realizado por Elías Mejía [Mejía, 2005]
[14]Según el estudio realizado por Roberto Hernández [Sampieri, 2015]

2. **Recolección de imágenes:** En esta fase es necesario recolectar imágenes de internet y capturas personalizadas (imágenes con problemas de rotación, ruido, iluminación y otros). Al finalizar esta fase se debe construir un *dataset* de imágenes de edificios históricos de la ciudad del Cusco.

3. **Investigación del método *Bag-of-Words*:** En esta etapa se debe buscar posibles métodos de extracción de características para el método *Bag-of-Words*. Así mismo, se debe buscar posibles técnicas de aprendizaje de máquina para solucionar el problema de la clasificación de edificios históricos de la ciudad del Cusco. Al finalizar con esta etapa se debe contar con al menos un método de extracción de características y un conjunto de técnicas de aprendizaje de máquina.

4. **Investigación de arquitecturas CNN:** En esta etapa se debe buscar posibles arquitecturas CNN que sirvan durante el proceso de extracción de características. Así mismo, se debe buscar posibles técnicas de aprendizaje de máquina para solucionar el problema de la clasificación de edificios históricos de la ciudad del Cusco. Al finalizar con esta etapa se debe contar con al menos una arquitectura CNN y un conjunto de técnicas de aprendizaje de máquina.

5. **Implementación de los métodos:** Esta fase se desarrolla a partir de la implementación y prueba de las técnicas de aprendizaje de máquina y los métodos basados en *Bag-of-Words* y *redes neuronales convolucionales*.

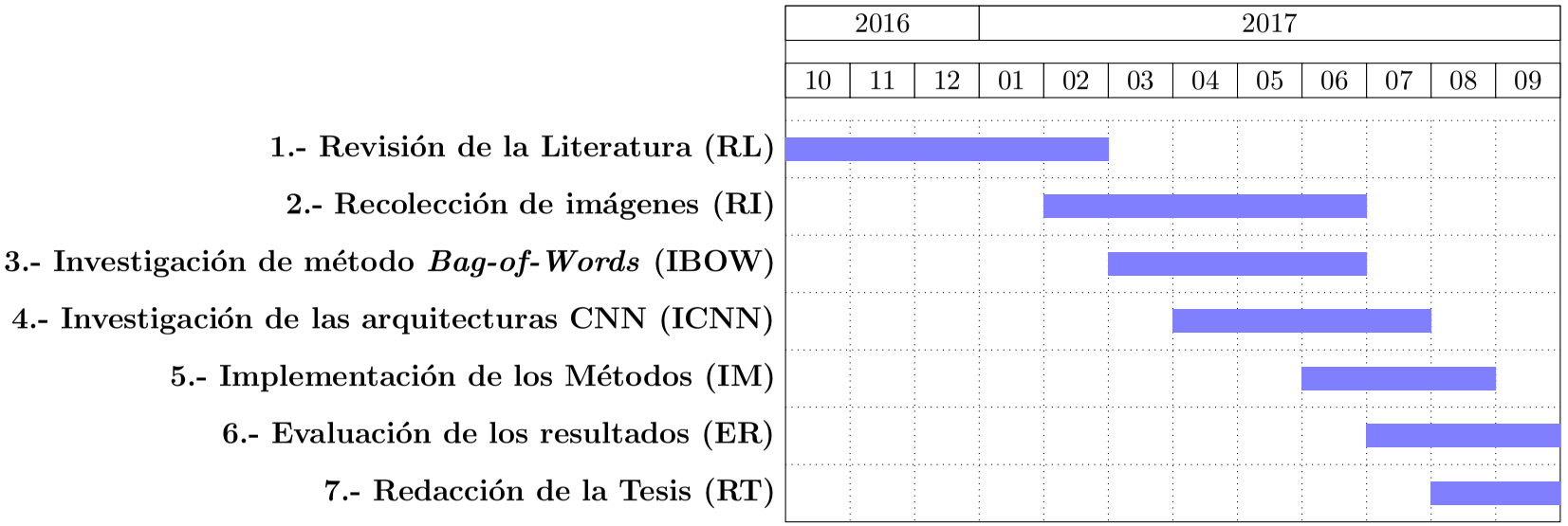
6. **Comparación de los métodos:** A partir de los experimentos realizados durante la etapa anterior, se procede a comparar los resultados de las técnicas de aprendizaje de máquina a través de métricas de aprendizaje. Al finalizar con esta etapa se procederá a identificar el rendimiento de cada técnica.

7. **Identificación de la técnica más óptima:** Finalmente, a partir de la etapa previa se procede a identificar la técnica de aprendizaje de máquina más óptima. Siendo esta reconocida a partir de obtener los resultados más elevados durante la evaluación de las métricas de aprendizaje.

1.7. Cronograma de Actividades

El cronograma de actividades se muestra en la Figura 5. Donde RL, RI, IBOW, ICNN, I, ER y RT representan la Revisión de la Literatura, Recolección de imágenes, Investigación de método *Bag-of-Words*, Investigación de las arquitecturas de *redes neuronales convolucionales*, Implementación de los Métodos, Evaluación de los resultados y Redacción de la Tesis respectivamente.

Figura 5: *Diagrama de Gantt del cronograma de actividades.*



Fuente: *Elaboración propia*

Parte II

Marco Teórico

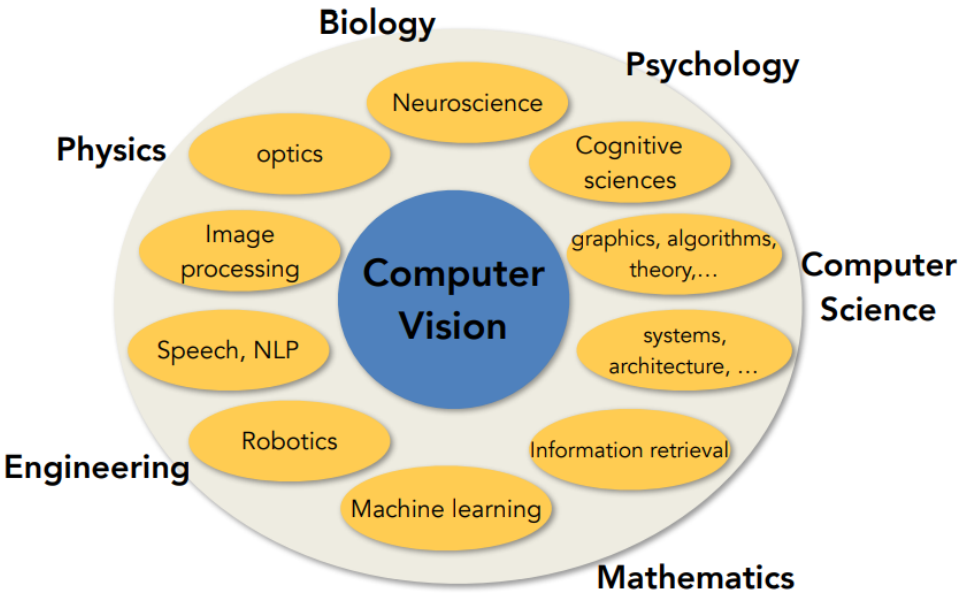
Capítulo 2

Marco Conceptual

2.1. Visión Computacional

La visión computacional es una de las disciplinas de la Inteligencia Artificial con mayor crecimiento en el mundo académico y la industria actualmente. Su objetivo es simple, emular el funcionamiento de la visión humana[15] para permitir que las computadoras vean, identifiquen y procesen imágenes de la misma manera que el sentido de la vista. Este objetivo se consigue gracias al apoyo de diferentes áreas de investigación como la Neurociencia, Procesamiento de imágenes, Machine Learning y otros (Figura 9).

Figura 6: *Áreas que contribuyen al desarrollo de la Visión Computacional.*



Fuente: [Karpathy, 2016]

El campo de la Visión Computacional ha sido testigo de continuos avances en los últimos años. Actualmente, existen aplicaciones de Visión Computacional muy

[15]Los seres humanos tenemos la capacidad de dar sentido a lo que nuestros ojos nos muestran. Pero casi todo ese trabajo se hace inconscientemente.

sofisticada, como la Autonavegación, Reconocimiento de Rostros, Diagnóstico en imágenes médicas y otros. Estas aplicaciones complejas resultan de resolver los desafios más comunes que presenta el campo de la Visión Computacional.

2.1.1. Desafíos en Visión Computacional

2.1.1.1. Reconocimiento o Clasificación

El reconocimiento o clasificación de imágenes busca etiquetar una imagen de acuerdo a su contenido, normalmente en un conjunto de categorías previamente definidas, como se muestra en la Figura 7.

Figura 7: *Ejemplos de clasificación de imágenes.*



Fuente: [Fridman et al., 2017]

2.1.1.2. Detección

La detección de objetos, busca localizar todos los objetos presentes en una imagen. Por lo tanto, tendrá varios cuadros delimitadores, como se muestra en la Figura 8.

Figura 8: *Detección de objetos en una imagen.*



Fuente: [Fridman et al., 2017]

2.1.1.3. Segmentación

La segmentación consiste en dividir una imagen digital en múltiples regiones para luego analizarlas. También se utiliza para distinguir objetos diferentes en una imagen [Khan, 2013], como se muestra en la Figura 9.

Figura 9: *Regiones segmentadas de una imagen.*



Fuente: [Fridman et al., 2017]

2.2. Bag-of-Words

Bag-of-words es una técnica que toma su origen en el ámbito de la clasificación de textos. Para entender de mejor forma su funcionamiento se desarrolla un ejemplo. Suponga que se tiene dos textos que hablan sobre la visión humana, el texto *a)* describe a la visión humana desde el punto de la vista biología, mientras que el texto *b)* lo hace desde un punto de vista computacional, como se muestra en la Figura 10.

Figura 10: *Ejemplos de texto.*

El ojo es la puerta de entrada por la que ingresan los estímulos luminosos que se transforman en impulsos eléctricos gracias a unas células especializadas de la retina que son los conos y los bastones. El nervio óptico transmite los impulsos eléctricos generados en la retina al cerebro, donde son procesados en la corteza visual.

a)

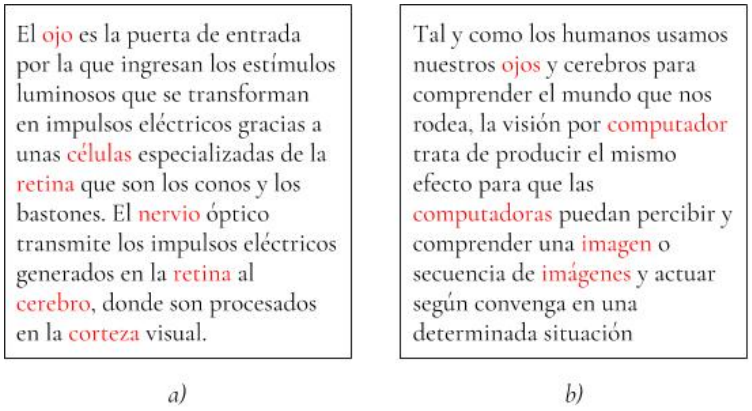
Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de producir el mismo efecto para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación

b)

Fuente: *Elaboración propia*

Luego, se procede a identificar las palabras más representativas de cada texto, como se muestra en la Figura 11.

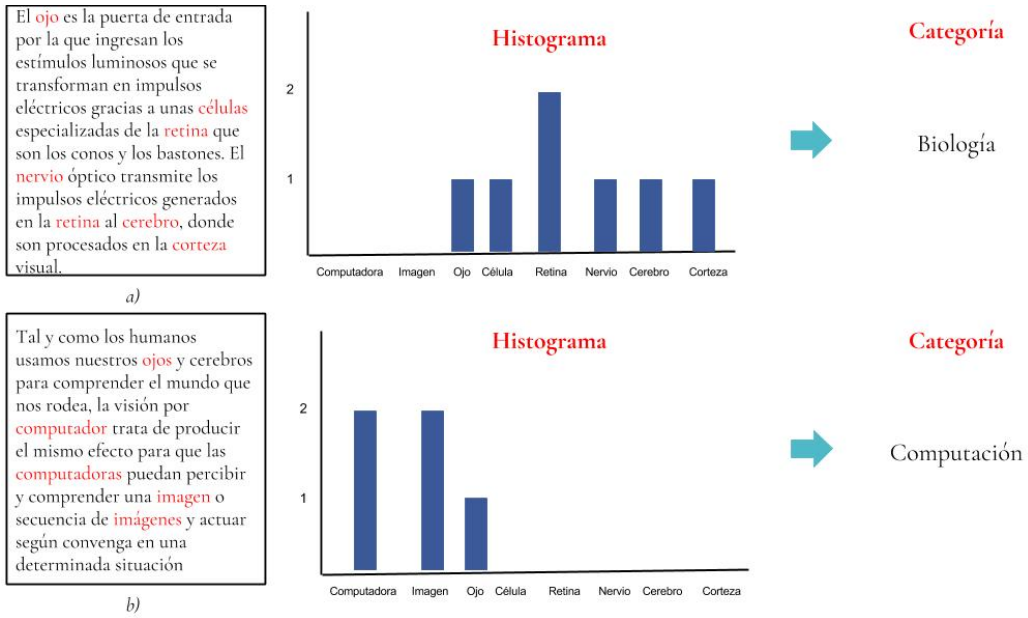
Figura 11: *Palabras representativas.*



Fuente: *Elaboración propia*

Finalmente, a partir de las palabras representativas, se genera un histograma de ocurrencias. A pesar de que existan palabras repetidas en ambos documentos, los histogramas son lo suficientemente diferentes para poder distinguir que los documentos pertenecen a categorías diferentes, como se muestra en la Figura 12.

Figura 12: *Generación de histogramas y clasificación.*



Fuente: *Elaboración propia*

Esta técnica se replica en el ámbito de la clasificación de imágenes, a través de un proceso de extracción de características y construcción de un *Codebook* (vocabulario visual). Estos conceptos son revisados a detalle en la Subsección 2.2.1 y Subsección 2.2.2 respectivamente.

2.2.1. Extracción de Características

Este proceso consiste en detectar y describir puntos de interés, el objetivo es obtener vectores de características que representen a una imagen. Los métodos más usados son SIFT y SURF, descritos a continuación.

2.2.1.1. Scale-invariant feature transform (SIFT)

Desarrollado por [Lowe, 2004], este método incluye un detector de puntos de interés y un descriptor. A pesar de los años se mantiene muy utilizado, debido a que su descriptor es muy robusto. SIFT es invariante a los cambios de escala, rotación e intensidad. Esto le permite obtener buenos resultados al hacer comparaciones entre imágenes como lo describe [Panchal et al., 2013]. Un ejemplo de esta técnica se muestra en la Figura 13. Donde la figura *a)* representa a la imagen de entrada, la figura *b)* representa a la imagen en escala de grises, la figura *c)* corresponde a los puntos de interés detectados y la figura *d)* simboliza a los puntos de interés descritos.

Figura 13: *Detección y descripción de puntos de interés utilizando el algoritmo SIFT.*



Fuente: *Elaboración propia*

El método consta de las siguientes etapas:

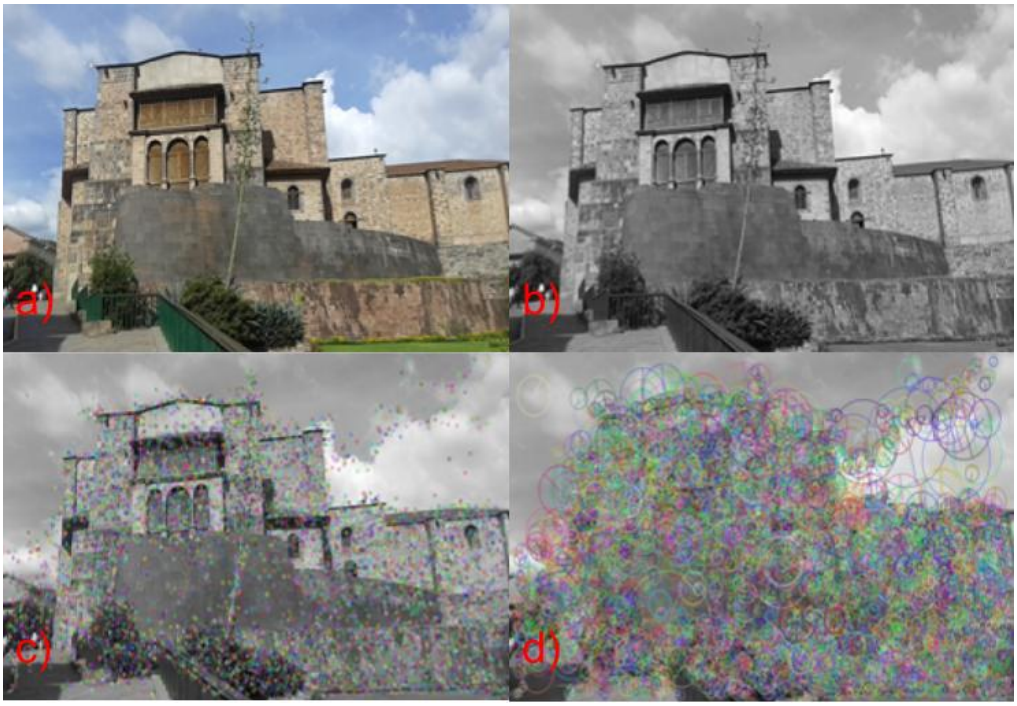
- Primero, para identificar los puntos de interés utilizando SIFT, se desarrollan dos pasos. Primero, se aplica sobre la imagen una secuencia de filtros Gaussianos a diferentes escalas y resoluciones. Segundo, se calcula la diferencia de Gaussianas de la imagen sucesivamente.
- Segundo, una vez que se encuentren los puntos de interés potenciales, estos necesitan ser refinados para obtener resultados más precisos. A partir de la serie de Taylor se obtiene una localización más exacta.

- Tercero, identificados los puntos de interés, se les asigna una orientación para lograr la invariación a la rotación de la imagen. Luego se crea un histograma de orientación, el histograma tendra tantos valores como orientaciones.
- Finalmente, el descriptor consiste de la concatenación de todos los histogramas generados para formar un vector de 128 características.

2.2.1.2. Speeded up robust features (SURF)

Desarrollado por [Bay et al., 2006], esta es una técnica que pretende ser más rápida y robusta que SIFT. También incluye un detector y descriptor. SURF es invariante a cambios de iluminación, debido a que detecta una mayor cantidad de puntos de interés como lo describe [Panchal et al., 2013], En la Figura 14 se muestra un ejemplo del algoritmo. Donde la imagen *a)* corresponde a la imagen de entrada, la imagen `b)` corresponde a la imagen en escala de grises, la imagen `c)` representa los puntos de interés detectados y la imagen *d)* simboliza los puntos de interés descritos

Figura 14: *Detección y descripción de puntos de interés utilizando el algoritmo SURF.*



Fuente: *Elaboración propia*

El método consta de las siguientes etapas:

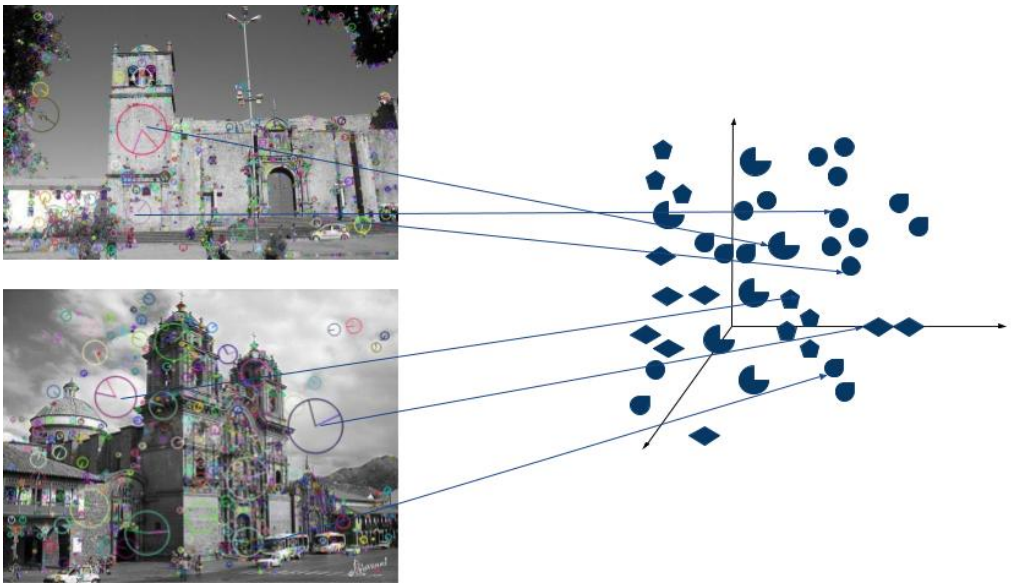
- Primero, a diferencia de SIFT, SURF utiliza imágenes integras para acelerar los cálculos, crea octavas. Pero no reduce la escala de la imagen, en cambio varia el tamaño del filtro.
- Segundo, para la búsqueda de puntos de interés, SURF utiliza el determinante Hessiano, lo que ayuda durante el proceso de localización y escalado de los puntos.

- Tercero, esta etapa consiste de la creación del descriptor. Es decir, la asignación de la orientación de cada uno de los puntos de interés obtenidos en la etapa anterior, para esto se realiza un cálculo de la respuesta de Haar en ambas direcciones x e y.
- Finalmente, SURF genera un vector de 64 características.

2.2.2. Codebook

El objetivo del *Codebook* es encontrar las palabras visuales más representativas. Este se construye utilizando una técnica de agrupación, la más utilizado es la técnica *K-Means*. Esta técnica es aplicada sobre los descriptores de características obtenidos durante la fase de extracción de características. *K-Means* es una técnica de Aprendizaje de Máquina, en específico pertenece a la categoría de aprendizaje no supervisado; esta técnica es detallada en la Subsección 2.3.4. Para empezar, la técnica toma como entrada los descriptores obtenidos previamente, como se observa en la Figura 15.

Figura 15: *Descriptores generados por SIFT, sirven como entrada del Codebook.*

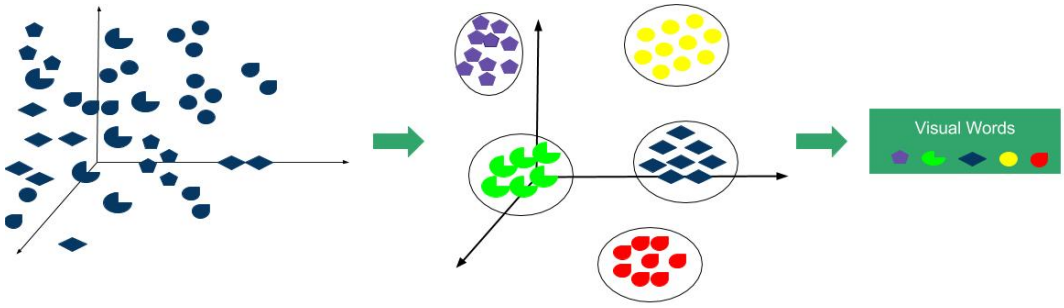


Fuente: *Elaboración propia*

Identificados los descritores se procede a ejecutar la técnica *K-Means* indicando el número de k *clusters*. Como resultado de la técnica devuelve un representante para cada uno de los k *clusters* que se quieren representar. Los diferentes *clusters* de muestras quedaran determinados por la distancia de cada *clusters* al representante del *clusters*, cada muestra se asignara al grupo que tenga un representante más cercano. En la Figura 16 se muestra la aplicación de la técnica con un $k = 5$. Como resultado se obtiene un conjunto de palabras visuales *(Visual Words)*.

Identificadas las palabras visuales más representativas, se procede a construir el histograma de palabras visuales, para ello se asigna cada característica local de la imagen a la palabra visual más parecida. Se necesita comparar todas las

Figura 16: *Construcción del Codebook y generación de las palabras visuales.*



Fuente: *Elaboración propia*

características locales de la imagen con todas las palabras visuales del *Codebook*, para reducir el costo computacional se utiliza la técnica *K-Nearest Neightbors*. Esta técnica es descrita en la Subsección 2.3.3.

2.3. Aprendizaje de Máquina

El Aprendizaje de Máquina es una rama de la Inteligencia Artificial, este emplea métodos computacionales que utilizan la experiencia para mejorar el rendimiento o para hacer predicciones acertadas [Esposito and Tse, 2017]. Sin embargo, los datos desempeñan un papel primordial en este proceso. A partir de estos se pueden considerar 3 tipos de Aprendizaje de Máquina, como son:

- **Aprendizaje Supervisado.-** En este tipo de aprendizaje, la máquina está abastecida de un conjunto de entradas con sus salidas deseadas. La máquina necesita estudiar esos conjuntos de entradas y salidas para encontrar una función general que asigne entradas a las salidas deseadas [Dasgupta and Nath, 2014]. Este tipo de modelos son utilizados para resolver problemas de clasificación[16] y regresión[17].

- **Aprendizaje No Supervisado.-** El objetivo es encontrar una buena representación interna de la entrada. Los ejemplos etiquetados no están disponibles en el aprendizaje no supervisión [Dasgupta and Nath, 2014]. Este tipo de modelos son utilizados para resolver problemas de clustering[18].

- **Aprendizaje por Refuerzo.-** Su objetivo es aprender que acciones tomar dada una cierta situación, a fin de maximizar una señal de recompensa.

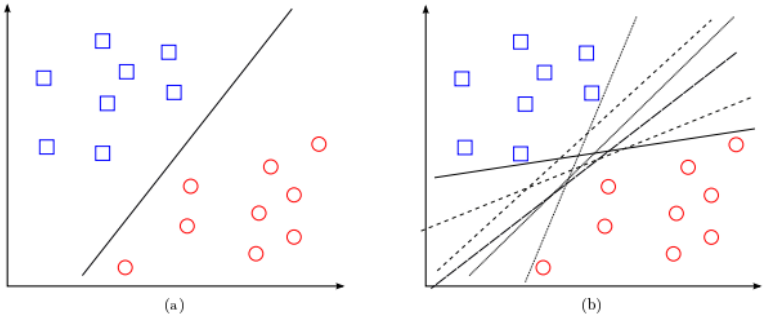
A continuación se describen algunas técnicas.

[16]Clasificación, la salida deseada es la etiqueta de una clase.
[17]Regresión, la salida esta asociada a predecir un valor y a partir de una muestra x.
[18]Clustering, esta asociado a la agrupación de datos.

2.3.1. Support Vector Machine

Es un modelo dedicado a separar clases distintas, a partir del uso de hiperplanos. A diferencia de otros modelos, *SVM* busca optimizar la estructura de separación entre clases [Betancourt, 2005], como se muestra en la Figura 17. Donde los cuadrados azules y círculos rojos son las dos clases, (a) La mejor línea que separa las clases y (b) El conjunto de las posibles líneas que separan las dos clases.

Figura 17: *Separación de dos clases utilizando SVM.*



Fuente: *[Suárez, 2014]*

Se tiene L puntos de entrenamiento, donde cada entrada x_m tiene D atributos (es decir, es de dimensionalidad D) y está en una de las dos clases y_i = -1 o +1, es decir, los datos de entrenamiento son de la forma:

$$\{x_i, y_i\} \text{ donde } i = 1 \ldots L, y_i \in \{-1, 1\}, x \in \Re^D \tag{1}$$

Se asume que los datos son linealmente separables, lo que significa que se puede dibujar una linea en la gráfica de x_1 vs x_2 separando las dos clases cuando $D = 2$ y un hiperplano de x_1, x_2. . . x_D cuando $D > 2$.

El hiperplano puede ser descrito por $w \cdot x + b = 0$ donde:

- w es la normal del hiperplano.
- $\frac{b}{\|w\|}$ es la distancia perpendicular entre el hiperplano y el origen.

Los *Support Vectors* son los puntos más próximos al hiperplano de separación y el objetivo del *Support Vector Machine* es orientar este hiperplano de tal manera que esté lo más lejos posible de los miembros más cercanos de ambas clases.

Observando la Figura 18, la implementación del *Support Vector Machine* se reduce a seleccionar las variables w y b para que los datos de entrenamiento se puedan describir mediante:

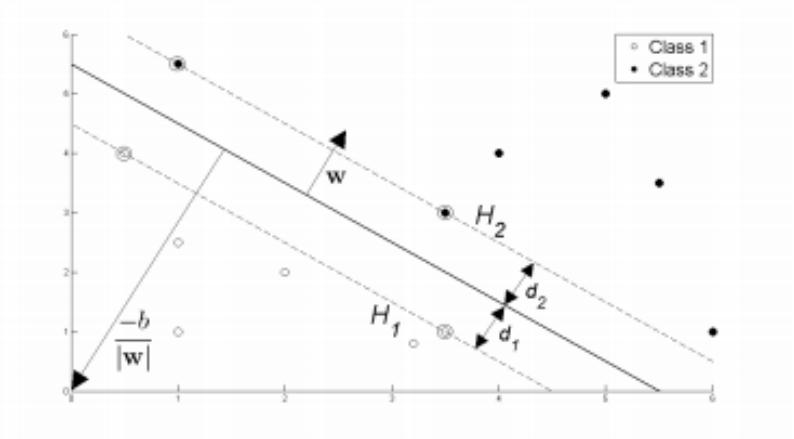
$$x_i \cdot w + b \geq +1 \quad \text{for } y_i = +1 \tag{2}$$

$$x_i \cdot w + b \leq -1 \quad \text{for } y_i = -1 \tag{3}$$

Estas ecuaciones se pueden combinar en:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall_i \tag{4}$$

Figura 18: *Detalles de los parámetros de sepación de dos clases utilizando SVM.*



Fuente: *[Fletcher, 2009]*

Si se consideran los puntos que están más próximos al hiperplano de separación. Es decir, los *Support Vectors* (mostrados en círculos en la Figura 18), entonces los hiperplanos H_1 y H_2 a los cuales pertenecen estos puntos se pueden describir como:

$$H_1 : x_i \cdot w + b = +1 \tag{5}$$

$$H_2 : x_i \cdot w + b = -1 \tag{6}$$

En la Figura 18, se define a d_1 y d_2 como las distancias a los hiperplanos H_1 y H_2 respectivamente. En un plano equidistante $d_1 = d_2$, esta cantidad es conocida como margen del *SVM* y es ésto lo que se busca maximizar.

La geometría simple del vector muestra que el margen es igual a $\frac{1}{\|w\|}$ y maximizarlo sujeto a la restricción en la equación (4) equivale a encontrar:

$$\min\|w\| \quad \text{tal que} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall_i \tag{7}$$

Minimizar $\|w\|$ es equivale a minimizar $\frac{1}{2}\|w\|^2$ y el uso de este término hace posible realizar la optimización de la Programación Cuadrática más adelante. Por lo tanto, se necesita encontrar:

$$\min\frac{1}{2}\|w\|^2 \quad \text{tal que} \quad y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall_i \tag{8}$$

Con el fin de cumplir con las restricciones en esta minimización, se tiene que asignar los multiplicadores de Lagrange α, donde $\alpha_i \geq 0\ \forall_i$:

$$L_P \equiv \frac{1}{2}\|w\|^2 - \alpha\left[y_i(x_i \cdot w + b) - 1\forall_i\right] \tag{9}$$

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L}\alpha_i\left[y_i(x_i \cdot w + b) - 1\right] \tag{10}$$

$$L_P \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L}\alpha_i y_i(x_i \cdot w + b) + \sum_{i=1}^{L}\alpha_i \tag{11}$$

Se desea encontrar w y b que minimicen, y α que maximice 11 (mientras que mantenga $\alpha_i \geq 0 \; \forall_i$). Se puede hacer ésto mediante la diferencial de L_P con respecto a w y b fijando la diferencial a:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \tag{12}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{13}$$

Sustituyendo 12 y 13 en 11 da una nueva fórmula que depende de α, se debe maximizar:

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \text{ Tal que } \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{14}$$

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i H_{ij} \alpha_i \text{ Donde } H_{ij} \equiv y_i y_j x_i \cdot x_j \tag{15}$$

$$L_D \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H_\alpha \text{ Tal que } \alpha_i \geq 0 \; \forall_i, \; \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{16}$$

Esta nueva fórmula para L_D se conoce como la forma *Dual* de la forma primaria para L_P. La forma *Dual* requiere que sólo se calcule de producto punto de cada vector de entrada x_i.

Cualquier punto que satisface la equación 13 sera un *Support Vector* x_s que tendrá la forma:

$$y_s(x_s \cdot w + b) = 1 \tag{17}$$

Reemplazando en la ecuación 12:

$$y_s^2(\sum_{m \in \mathrm{S}} \alpha_m y_m x_m \cdot x_s + b) = 1 \tag{18}$$

Donde S indica el conjunto de índices de los *Support Vectors*. S se determina mediante la búsqueda de los índices i donde $\alpha_i > 0$. Multiplicando por y_s y luego usando $y_s^2 = 1$ de 2 y 3:

$$y_s^2(\sum_{m \in \mathrm{S}} \alpha_m y_m x_m \cdot x_s + b) = y_s \tag{19}$$

$$b = y_s - \sum_{m \in \mathrm{S}} \alpha_m y_m x_m \cdot x_s \tag{20}$$

En lugar de utilizar *Support Vector* x_s arbitrario, es mejor tomar un promedio sobre todos los *Support Vectors* en S:

$$b = \frac{1}{N_s} \sum_{s \in \mathrm{S}} (y_s - \sum_{m \in \mathrm{S}} \alpha_m y_m x_m \cdot x_s) \tag{21}$$

Ahora se tiene las variables w y b que definen la orientación óptima del hiperplano de separación. Por lo tanto, el *Support Vector Machine* [Fletcher, 2009].

2.3.2. Random Forest (RF)

Random Forest se basa en generar un número importante de árboles, para entrenarlos y calcular el promedio de su salida. Para la construcción de cada árbol se le asigna [Gouron, 2016]:

- Una parte aleatoria de los datos *(tree bagging)*
- Una parte aleatoria de las características *(feature sampling)*

Dado un conjunto de clasificadores $h_1(x), h_2(x), \ ... \ , h_K(x)$, y un conjunto de entrenamiento generado al azar a partir de la distribución de los vector aleatorio Y y X, se define la función margen:

$$\mathrm{mg}(X,Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j) \tag{22}$$

Donde $I(\cdot)$ es la función de indicador. El margen mide la medida en que el número promedio de votos en X, Y para la clase correcta excede el voto promedio para cualquier otra clase. Cuanto mayor sea el margen, mayor será la confianza en la clasificación. El error de generalización está dado por:

$$\mathrm{PE}^{\star} = P_{X,Y}(\mathrm{mg}(X,Y) < 0) \tag{23}$$

Donde los subíndices X, Y indican que la probabilidad está sobre el espacio X, Y. En *Random Forest*, $h_k(X) = h(X, \Theta_k)$. Donde $h(X, \Theta_k)$ es el clasificador entrenado con el vector aleatorio Θ_k. Para un gran número de árboles, se deriva de la Ley Fuerte de Grandes Números y de la estructura de árbol: A medida que el número de árboles aumenta, casi seguramente todas las secuencias Θ_1 ... $\mathrm{PE}^{\star}$ converge a:

$$\mathrm{P}_{X,Y}(P_{\Theta}(h(X,\Theta) = Y) - \max_{j \neq Y} P_{\Theta}(h(X,\Theta) = j) < 0) \tag{24}$$

Todo este trabajo fue desarrollado por [Breiman, 2001]. En la Figura 19 se muestran los diferentes resultados que se obtiene al variar el número de clases, En *a)* se presentan resultados con dos clases, en *b)* se presentan resultados con tres clases y en *c)* con cuatro clases.

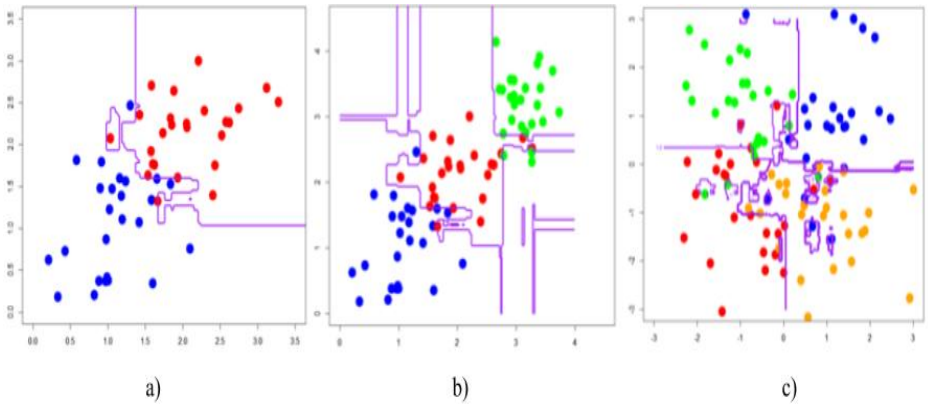
2.3.3. K-Nearest Neighbors (kNN)

La idea básica de *kNN* es que un caso de prueba se va a clasificar en la clase más frecuente a la que pertenecen sus k vecinos más cercanos del conjunto de entrenamiento.

El clasificador *kNN* se basa comúnmente en la distancia euclídea entre una muestra de prueba y las muestras de entrenamiento especificadas. Sea x_i una muestra de entrada con p características $(x_{i1}, x_{i2}, ..., x_{ip})$, n sea el número total de muestras de entrada $(i = 1, 2, ..., n)$ y p el número total de características $(j = 1, 2, ..., p)$. La distancia euclídea entre la muestra x_i y $x_l (l = 1, 2, ..., n)$ se define como:

$$\mathrm{d}(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + (x_{ip} - x_{lp})^2} \tag{25}$$

Figura 19: *Resultados de aplicar Random Forest a diferentes clases.*

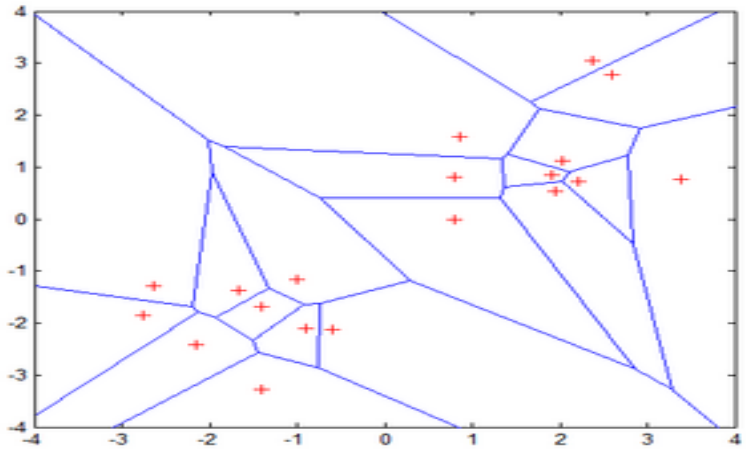


Fuente: *[Ozaki, 2014]*

Una representación gráfica del concepto de *kNN* se ilustra en el diagrama de Voronoi (Figura 20). La gráfica muestra 19 elementos representados con el signo " + ", y la celda de Voronoi, R, que rodea cada muestra. Una celda de Voronoi encapsula todos los puntos vecinos que están más próximos a cada muestra y se define como:

$$\mathrm{R}(i) = \{x \in \mathbb{R}^p : d(x, x_i) \leq d(x, x_m),\ \forall i \neq m\} \tag{26}$$

Figura 20: *Diagrama de Voronoi.*



Fuente: *[Peterson, 2009]*

Donde R_i es la celda de Voronoi para la muestra x_i, y x representa todos los puntos posibles dentro de la celda de Voronoi R_i. La gráfica de Voronoi refleja principalmente dos características de un sistema de coordenadas: *i)* todos los puntos posibles dentro de la celda de Voronoi de una muestra son los puntos vecinos más cercanos para esa muestra, y *ii)* para cualquier muestra, la muestra más cercana está determinada por el borde de Voronoi más cercano. Usando esta

última característica, la regla de clasificación de *knn* es asignar a una muestra de prueba la etiqueta de categoría mayoritaria de sus k muestras de entrenamiento más cercanas. En la práctica, k se elige generalmente para ser impar, para evitar lazos. El valor $k = 1$ generalmente se utiliza como un estándar durante el proceso clasificación de *kNN* [Peterson, 2009].

2.3.4. K-Means

La técnica toma como entrada un conjunto de muestras y un número de k *clusters*. Como resultado la técnica devolverá un representante de cada uno de los k *clusters* [Solem, 2012]. Sea $X = x_1, x_2, x_3, ..., x_n$ el conjunto de puntos de datos y $V = v_1, v_2, ..., V_c$ el conjunto de centros. A continuación se presenta la secuencia de pasos a seguir [Ortega et al., 2010]:

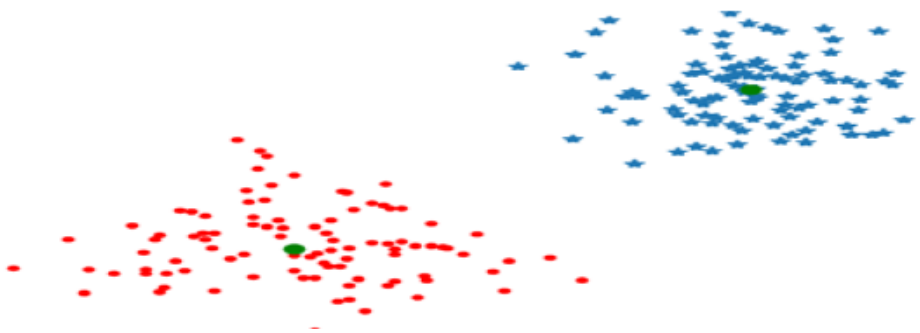
1. Selección aleatoria de los centroides 'c'.
2. Calculo de la distancia entre cada punto de datos y los centros de agrupación.
3. Asignación del punto de datos al centro de *cluster* cuya distancia del centro del *cluster* sea mínima de todos los centros de *clusters*.
4. Se Vuelve a calcular el nuevo centro del *cluster* utilizando:

$$v_i = \frac{1}{c_i} \sum_{j=1}^{c_i} x_i \tag{27}$$

5. Volver a calcular la distancia entre cada punto de datos y los nuevos centros de agrupación obtenidos.
6. Si no se reasignó ningún punto de datos, entonces detener, caso contrario, repetir desde el paso 3).

En la Figura 21 se muestra un ejemplo de la técnica *k-means* con un $k = 2$, donde los puntos verdes representan a los centroides.

Figura 21: *Aplicación de la técnica k-means sobre puntos en 2D.*



Fuente: *[Solem, 2012]*

2.4. Redes Neuronales Artificiales

Las Redes Neurales Artificiales son un tipo de técnica de aprendizaje de máquina basada en metodologías computacionales inspiradas en el estudio de las redes neuronales biológicas[19]. Una red neuronal artificial es básicamente una función matemática, esta constituida por capas de nodos que funcionan como un dispositivo sumador no lineal. Estos nodos están interconectados por líneas de conexión ponderadas, de forma similar a las conexiones del cerebro.

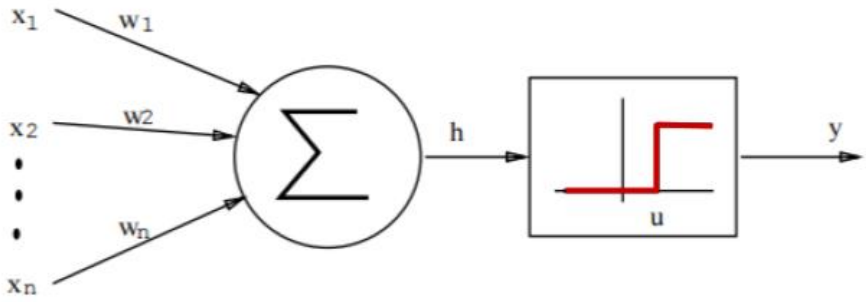
2.4.1. Modelo computacional de una neurona

El modelo de neurona propuesta por [McCulloch and Pitts, 1943] (Ver Figura 22) sirvió de base para futuras investigaciones. Esa neurona matemática calcula una suma ponderada de sus n señales de entrada, x_j; donde $j = (1, 2, ..., n)$, y genera una salida de 1 si esta suma está por encima de un cierto *threshold* u, y una salida de 0 en caso contrario. Esto se demuestra matemáticamente como:

$$y = \theta(\sum_{j=1}^{n} w_j x_j - u) \tag{28}$$

Donde θ es una función de activación, y w_j es el peso de sinapsis asociado con la entrada j-ésima. Para simplificar la notación, a menudo se considera el umbral u como otro peso $w_0 = u$ que está unido a la neurona con una entrada constante, $x_0 = 1$.

Figura 22: *Modelo de neurona porpuesta por McCulloch y Pitts.*

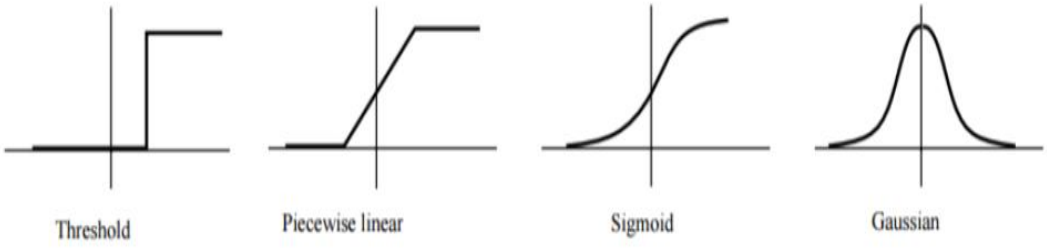


Fuente: *[Jain et al., 1996]*

Este modelo de neurona se ha generalizado de muchas maneras. Una generalización obvia es utilizar funciones de activación distintas de la función de *threshold*, por ejemplo, *linear active function, sigmoide y gaussian function*, representados en la Figura 23. La función sigmoide es la más utilizada en las redes neuronales artificiales. Es una función estrictamente creciente que muestra suavidad y tiene las propiedades asintóticas deseadas [Jain et al., 1996]. En la Subsección 2.4.3 se detalla la metodología de estas y otras funciones de activación.

[19]En [Shiffman, 2012] se describe al cerebro humano como una red neuronal biológica, una red interconectada de neuronas que transmite patrones elaborados de señales eléctricas. Las dendritas reciben señales de entrada y, sobre la base de esas entradas, disparan una señal de salida a través de un axón.

Figura 23: *Diferentes tipos de función de activación.*

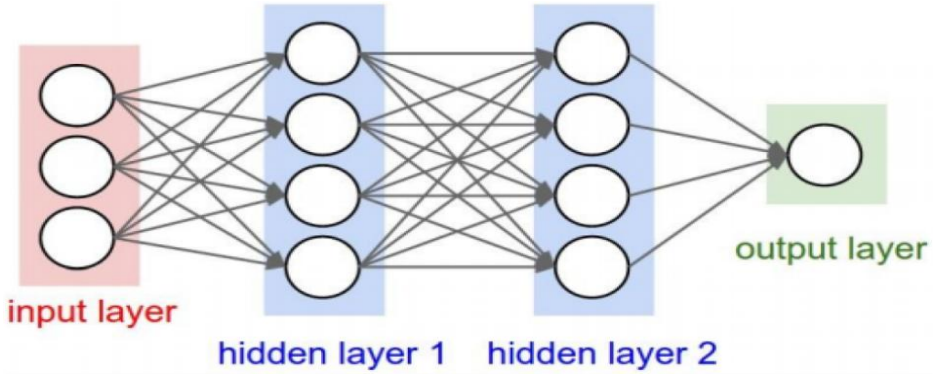


Fuente: *[Jain et al., 1996]*

2.4.2. Arquitectura de una Red Neuronal

La arquitectura de una Red Neuronal básica se detalla en la Figura 24. Donde la capa más a la izquierda se llama capa de entrada, las neuronas dentro de esta capa se llaman neuronas de entrada; la capa que se encuentra en el extremo derecho se denomina capa de salida, contiene la(s) neurona(s) de salida. Finalmente, las capas intermedias se denominan capas ocultas, estas contienen las neuronas ocultas.

Figura 24: *Arquitectura básica de una red neuronal.*



Fuente: *[Li and Karpathy, 2015]*

En la Figura 25 se muestra todos los modelos de redes neuronales diseñadas hasta la actualidad. Sin embargo, en este trabajo sólo se analizan algunos, como son:

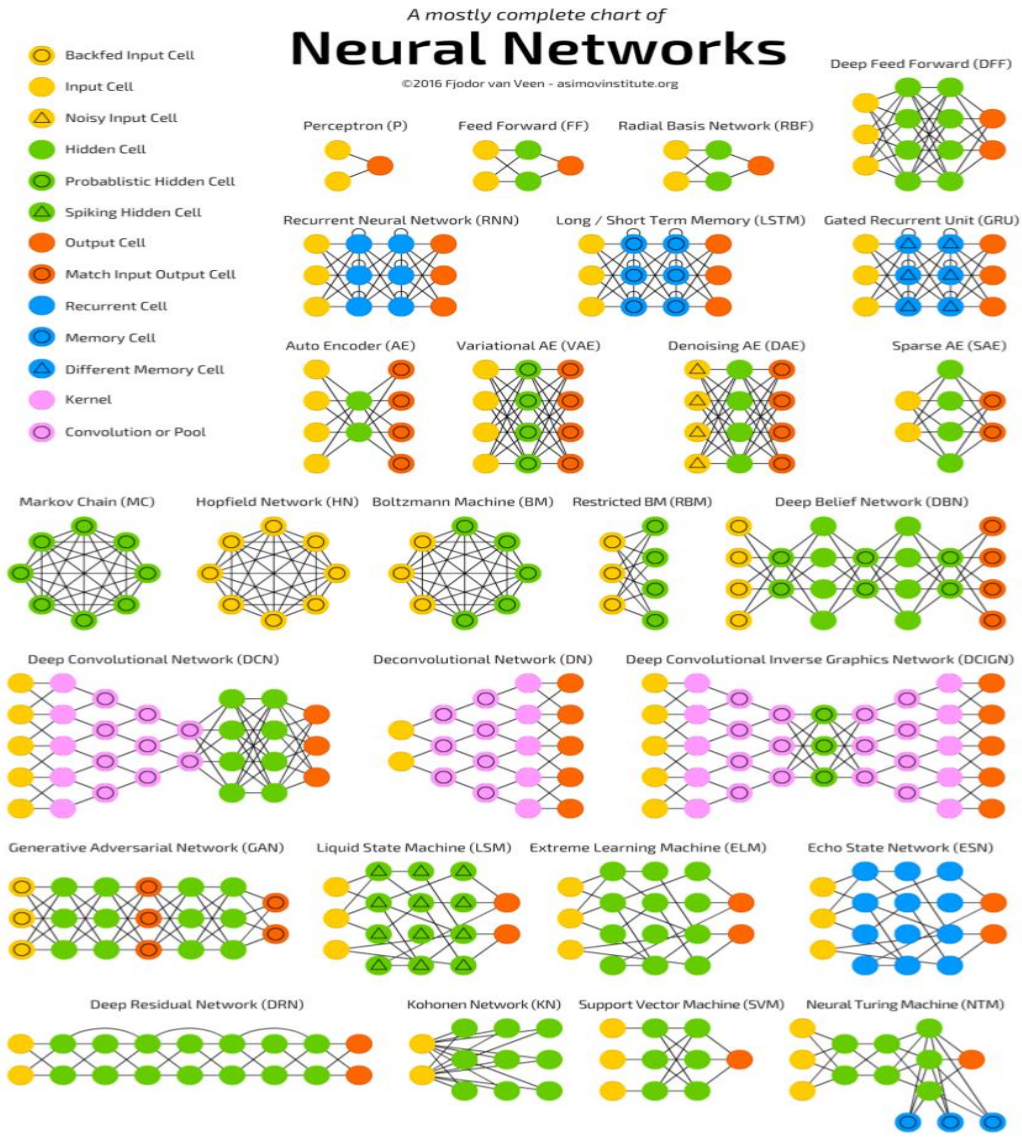
- *Feed forward neural networks.*- Cuando la salida de una capa se convierte en la entrada de la siguiente capa, eso significa que no hay bucles en la red.
- *Neurona Sigmoide.*- La neurona Perceptron tiene un problema enorme, un pequeño cambio en uno de los pesos cambiaría completamente el comportamiento de la red. Sin embargo, la neurona Sigmoide supera ese problema, puede hacer pequeños cambios a los pesos y bias[20], y hacer una pequeña

[20]La unidad bias es una neurona extra añadida a cada capa de pre-salida que almacena el valor de 1.

modificación en la salida. Este es el hecho que permite que una red neuronal aprenda. La unión de múltiples neuronas Sigmoide es conocida como perceptrones multicapa o MLPs, a pesar de estar formadas por neuronas sigmoides, no perceptrones [Nielsen, 2015].

- *Red Neuronal Convolucional.-* Una *red neuronal convolucional* o *deep convolutional network* consiste en una serie de capas convolucionales y submuestreos opcionalmente seguidas por capas completamente conectadas. Este modelo de red neuronal es profundizada en la Subsección 2.5.1.

Figura 25: *Arquitecturas de red neuronal propuestas hasta la actualidad.*



Fuente: *[Van Veen, 2014]*

2.4.3. Función de Activación

Cada función de activación o normalización toma un solo número y realiza una cierta operación matemática fija en él. Su objetivo es eliminar la linealidad. Existen varias funciones de activación que se puede encontrar en la práctica, En la Figura 26 se muestra el gráfico de algunas funciones:

- ***Sign.-*** Se genera una salida entre $[-1, 1]$.

$$\text{Sign}(x) = \begin{cases} +1 & \text{, if } x \geq 0 \\ -1 & \text{, if } x < 0 \end{cases} \tag{29}$$

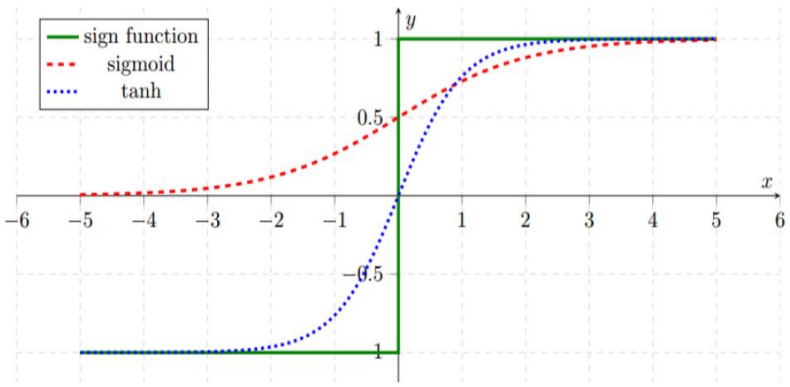
- ***Sigmoide.-*** Se genera una salida entre $[0, 1]$.

$$\text{Sigmoid}(x) = \frac{1}{1 + \text{e}^{-x}} \tag{30}$$

- ***Tanh.-*** Se genera una salida entre $[-1, 1]$.

$$\tanh(x) = \frac{2}{1 + \text{e}^{-2x}} - 1 \tag{31}$$

Figura 26: *Función de activación Sign, Sigmoide y tanh.*

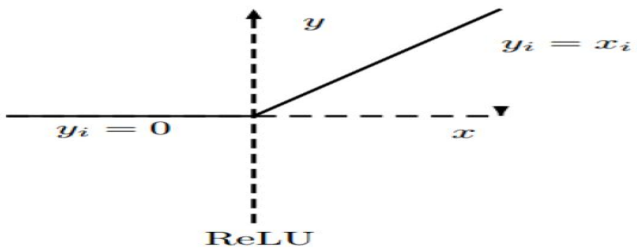


Fuente: *[Thoma, 2015]*

- ***Rectified Linear Unit (ReLu).-*** Se genera una salida entre $[0, x]$ [Xu et al., 2015]. En la Figura 27 se muestra la gráfica de la función.

$$\text{Relu}(x) = \begin{cases} 0 & \text{, if } x < 0 \\ x & \text{, if } x \geq 0 \end{cases} \tag{32}$$

Figura 27: *Función de activación ReLu.*



Fuente: *[Xu et al., 2015]*

2.4.4. Aprendizaje

El objetivo del entrenamiento de una red neuronal es encontrar pesos y bias que minimicen la función de costo[21] para todas las entradas de entrenamiento x. Siendo y(x) la salida deseada [Nielsen, 2015]. A continuación, se muestra algunas funciones de costo:

- ***Función de Coste Cuadrático.-***

$$\mathrm{C} = \frac{1}{2n}\sum_{x}\|y(x) - \mathrm{a}^{L}(x)\|^2 \tag{33}$$

  Donde: n es el número total de ejemplos de entrenamiento; la suma es sobre los ejemplos de entrenamiento individual, x; $y = y(x)$ es la salida deseada correspondiente; L denota el número de capas en la red; y $a^L = a^L(x)$ es el vector de activaciones de salida de la red cuando se introduce x.

- ***Función de coste de Entropía Cruzada.-***

$$\mathrm{C} = -\frac{1}{n}\sum_{x}\left[y\ln(a) + (1-y)\ln(1-a)\right] \tag{34}$$

  Donde n es el número total de elementos de entrenamiento, la suma es sobre todas las entradas de entrenamiento, x; y es la salida deseada correspondiente [Nielsen, 2015].

2.4.4.1. Backpropagation

Backpropagation, es una técnica de aprendizaje supervisado utilizado durante el descenso de gradiente. Dada una red neuronal artificial y una función de costo, el método calcula el gradiente de la función de costo con respecto a los pesos y bias de la red neuronal [McGonagle and Williams, 2017]. El descenso de gradiente sirve para encontrar los pesos w_k y bias b_l que minimicen la función de costo de las ecuaciones 33 o 34. Para obtener eso, se reemplazan los pesos y bias con $\partial C/\partial w_k$

[21] La función de costo, función objetivo o de perdida permite encontrar pesos y bias para que la salida de la red se aproxime a y(x)

y $\partial C/\partial d_l$ respectivamente. Entonces, el descenso de gradiente queda en términos de:

$$w_k = w_k^{'} = w_k - n\frac{\partial C}{\partial w_k} \tag{35}$$

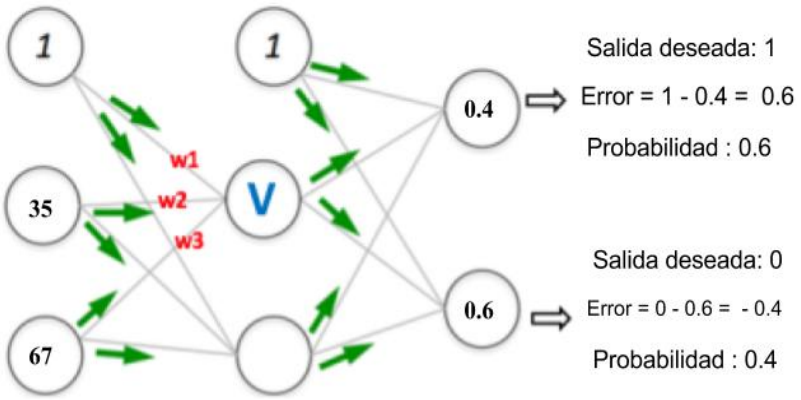
$$b_l = b_l^{'} = b_l - n\frac{\partial C}{\partial b_l} \tag{36}$$

Donde $w_k^{'}$ y $b_l^{'}$ representa la primera derivada de w_k y b_l respectivamente, C representa la función de costo y n el número total de entradas [Nielsen, 2015].

Sin embargo, Backpropagation desarrolla una secuencia de pasos antes de conseguir su objetivo [LeCun et al., 2015], como son:

- ***Forward Propagation*** Este proceso se realiza con las neuronas de cada capa. Primero, todos los pesos de la red se asignan al azar. Segundo, se calcula una entrada z, que representan la suma ponderada de las neuronas, pesos y bias de entrada a una neurona $\sum_i w_i x_i + bw_0$, donde w_i, x_i y b representan los pesos, entradas y bias de cada neurona respectivamente; w_0 es un peso aleatorio. Tercero, se aplica una función de activación $f(\cdot)$ a z. Este proceso se desarrolla en todos los nodos de la red neuronal(excepto en las neuronas de entrada). Finalmente, se aplica una función de coste, la salida indica la tasa de error generada. En la Figura 28 se ilustra de mejor forma estos conceptos, los pesos se asignan al azar, se consideran a los pesos w_1, w_2 y w_3 como las entradas a un nodo V. Las neuronas de entrada para este ejemplo son 35 y 67, mientras que las salidas deseadas son 0 y 1. Entonces, aplicada una función de activación cualquiera, el valor de V es igual a $f(1 \cdot w_1 + 35 \cdot w_2 + 67 \cdot w_3)$, este proceso se replica con todos los nodos de la red. Finalmente, los nodos de salida tienen valores equivalentes a 0.4 y 0.6. Estos resultados están muy lejos de las salidas deseadas, por lo tanto, se aplica una función de costo cualquiera, dando como resultado que la primera neurona de la capa de salida posee un error equivalente a $1 - 0.4 = 0.6$ y la segunda neurona posee un valor equivalente a $0 - 0.6 = -0.4$. Por lo tanto la probabilidad generada es equivalente a 0.6 y 0.4 respectivamente.

Figura 28: *Secuencia de pasos de la fase Forward Propagation.*



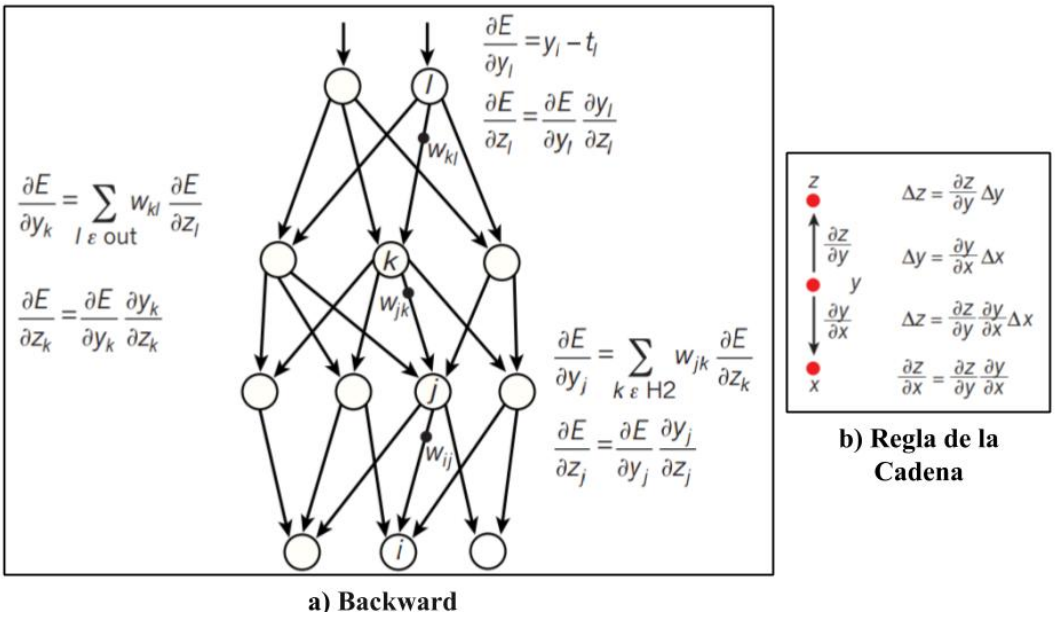
Fuente: *[Ujjwalkarn, 2016b]*

- ***Backward*** Este proceso tiene por objetivo minimizar el error. Por lo tanto, se calcula la derivada de la función de coste con respecto a la salida de cada neurona $\partial E/\partial y_l$, donde ∂E y ∂y_l representan la derivada de la función de coste y la derivada de cada neurona respectivamente. Ese cálculo se resuelve como la suma ponderada de las derivadas de la función de coste con respecto a las entradas totales de las unidades en la capa anterior, ese cálculo se realiza en todas las capas como se muestra en la Figura 29.*a*. Es decir:

$$\frac{\partial E}{\partial y_k} = \sum_{l \in salida} w_{kl} \frac{\partial E}{\partial z_l} \tag{37}$$

Donde $\partial E/\partial z_l$ se resuelve utilizando la regla de la cadena (Ver Figura 29.*b*).

Figura 29: *Secuencia de pasos de la fase Backward.*



Fuente: *[LeCun et al., 2015]*

2.4.4.2. Descenso de Gradiente Estocástico

Es una técnica que acelera el aprendizaje. La idea es calcular el gradiente de la función de costo para una pequeña muestra de entrenamiento. Haciendo un promedio sobre esa muestra*(mini-batch)* se puede obtener rápidamente una buena estimación del gradiente verdadero, esto ayuda a acelerar el gradiente y el aprendizaje [Nielsen, 2015].

$$w_k = w_k^{'} = w_k - \frac{n}{m} \sum_j \frac{\partial C_{x_j}}{\partial w_k} \tag{38}$$

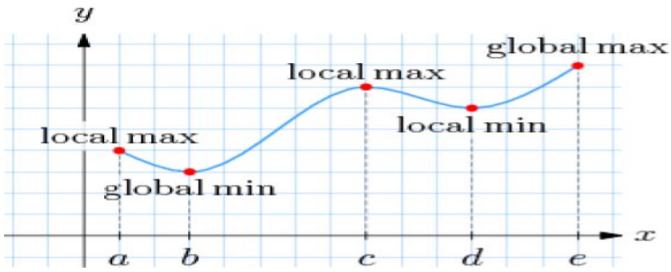
$$b_l = b_l^{'} = b_l - \frac{n}{m} \sum_j \frac{\partial C_{x_j}}{\partial b_l} \tag{39}$$

Donde w_k y b_l representan los pesos y bias respectivamente, n el número de neuronas y m la muestra [Nielsen, 2015].

2.4.4.3. Adam Optimization

Es una técnica basada en *mini-batch*, que adapta el índice de aprendizaje[22] de los parámetros del modelo. Se mantiene una tasa de aprendizaje para cada peso de la red (parámetro) y se adapta por separado a medida que se desarrolla el aprendizaje. El método calcula las tasas individuales del aprendizaje para diferentes parámetros a partir de estimaciones del primer y segundo momento de los gradientes [Kingma and Ba, 2014]. El descenso de gradiente es equivalente al proceso de "descender de una colina" y encontrar un mínimo local o mínimo global (Ver Figura 30).

Figura 30: *Mínimo local y mínimo global.*

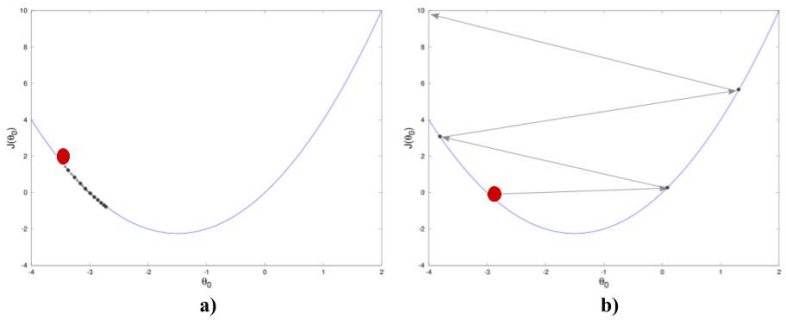


Fuente: *https://goo.gl/9zWZLn*

El valor que se asigne al índice de aprendizaje es importante, debido a que puede presentar los siguientes problemas:

- Si el índice de aprendizaje es demasiado pequeño, la técnica requerirá demasiadas épocas[23] (Ver Figura 31.*a*).
- Si el índice de aprendizaje es demasiado grande, el descenso gradiente superará los mínimos locales (Ver Figura 31.*b*).

Figura 31: *Desafíos durante la asignación del índice de aprendizaje.*



Fuente: *[Triangles, 2017]*

Sin embargo, para asignar de mejor forma el índice de aprendizaje se utiliza la técnica Adam optimization [Kingma and Ba, 2014].

[22] El índice de aprendizaje η es un parámetro > 0, indica que tan rápido aprenderá el modelo
[23] Una época es un paso completo a través de todos los datos de entrenamiento.

2.5. Deep Learning

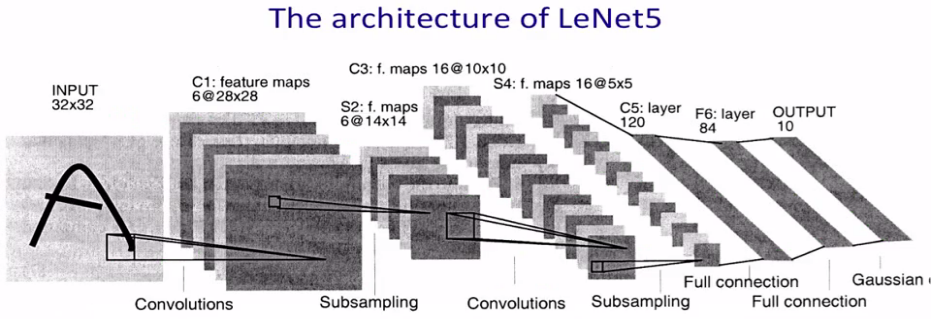
Deep Learning es un área del Aprendizaje de Máquina [Rere et al., 2015]. Permite crear modelos computacionales que están compuestos de múltiples capas[24] de procesamiento para aprender representaciones de datos con múltiples niveles de abstracción. Las Redes Neuronales Convolucionales son un tipo de arquitectura Deep Learning, estas son descritas a continuación.

2.5.1. Red Neuronal Convolucional

Las Redes Neuronales Convolucionales (CNNs) fueron diseñadas para procesar datos en forma de matrices, por ejemplo matrices 1D para señales y secuencias, incluyendo el lenguaje; 2D para imágenes o espectrogramas de audio; y 3D para video o imágenes volumétricas [LeCun et al., 2015].

El termino red neuronal convolucional indica que la red emplea una operación matemática llamada convolución. La convolución es un tipo de operación lineal. Las CNNs son simplemente redes neuronales que utilizan convolución en al menos una de sus capas [Goodfellow et al., 2016]. Una CNN típica se muestra en la Figura 32. Este modelo es denominado *Lenet-5* y sus operaciones son detalladas en la siguiente Subsección.

Figura 32: *Arquitectura de red neuronal convolucional desarrollada en 1998.*



Fuente: *[LeCun et al., 1998]*

2.5.1.1. Capa de Convolución

La operación de convolución esta denotada por un asterisco:

$$S(i,j) = (I * K)(i,j) \tag{40}$$

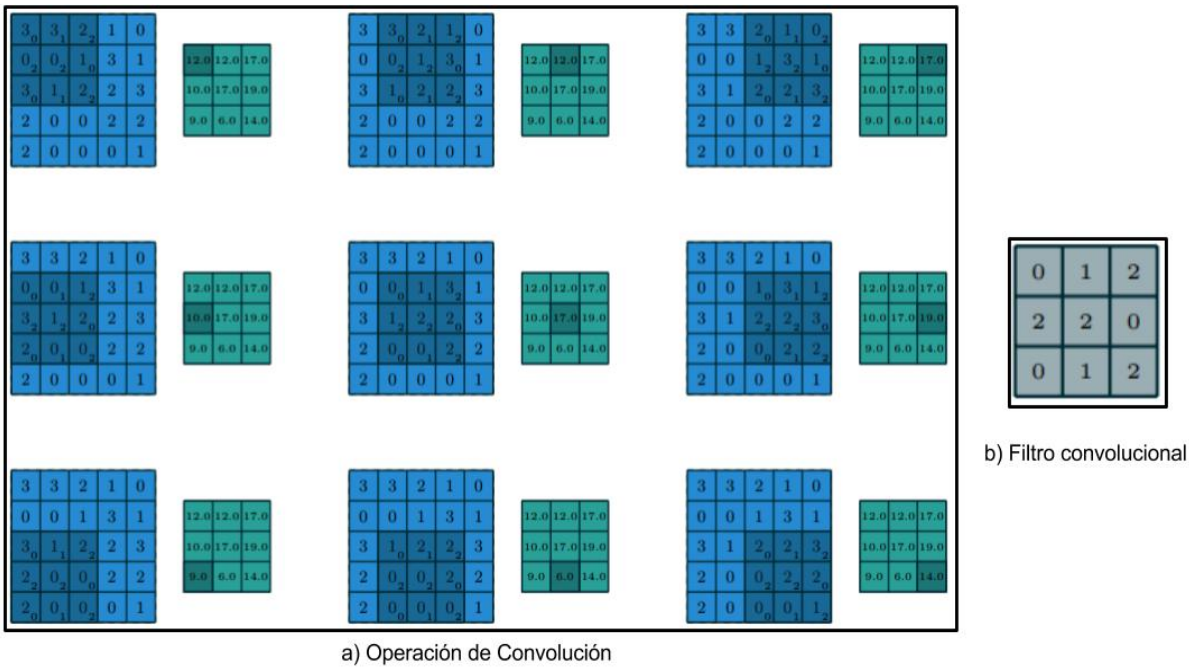
Donde I representa a una imagen y K un filtro convolucional, ambos en un escenario de dos dimensiones. Por ultimo, el resultado S se denomina *feature map* [Goodfellow et al., 2016].

Para entender como funciona esto, en la Figura 33.*a* se proporciona un ejemplo. La matriz azul clara representa a una imagen de entrada (para simplificar la

[24]Según [Nielsen, 2015] una arquitectura de red neuronal es de tipo Deep Learning si presenta dos o más capas ocultas.

operación solo se considera un canal de color) y el filtro convolucional se representa en la Figura 33.*b*; el filtro convolucional se desliza sobre la imagen, y en cada región, el producto entre cada filtro y la región delimitada de la imagen se calcula y los resultados se suman para obtener la salida en la región actual.

Figura 33: *Valores de salida de una operación de convolución.*



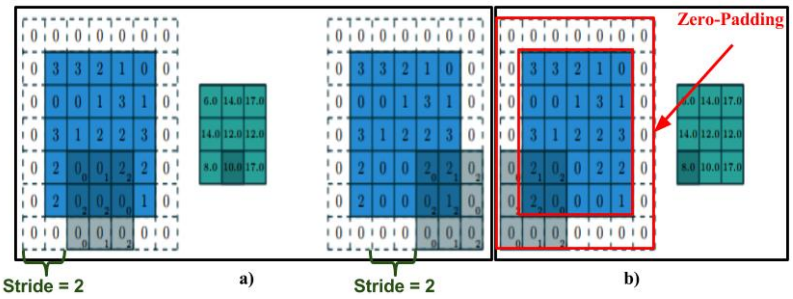
Fuente: *[Dumoulin and Visin, 2016]*

Los *feature maps* generados estan controlado por 3 parámetros:

- **Profundidad.-** La profundidad corresponde al número de filtros convolucionales que se utilizan durante la operación de convolución.
- ***Stride.-*** Es el número de píxeles por los que desliza un filtro convolucional sobre la matriz de entrada. Se muestra un ejemplo en la Figura 34.*a*.
- ***Padding.-*** Consiste en rellenar la matriz de entrada con ceros alrededor del borde, de modo que se pueda aplicar el filtro a los elementos de frontera de la matriz de entrada. Se muestra un ejemplo en la Figura 34.*b*.

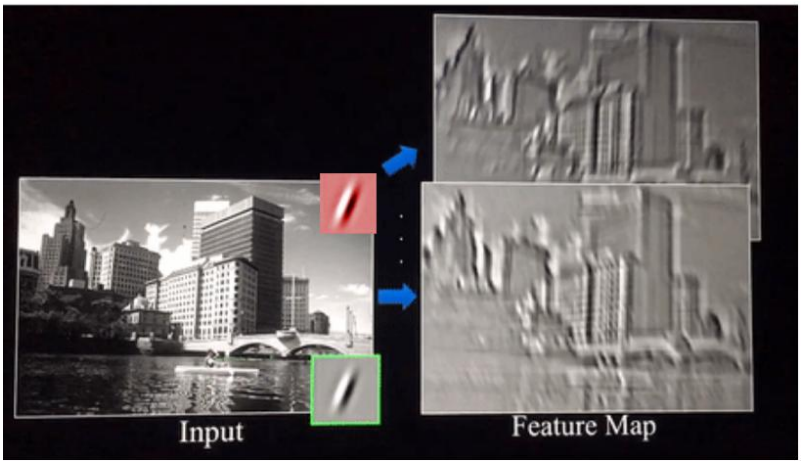
Así mismo, la salida de la convolución es no linealizada a través de una función de activación, se puede utilizar una de las funciones descritas en la Subsección 2.4.3. Por otro lado, la aplicación de una operación de convolución sobre una imagen se muestra en la Figura 35 y la aplicación de una función de activación se muestra en la Figura 36, donde la región oscura representa a los píxeles negativos, por el contrario la región más clara representa a los píxeles positivos.

Figura 34: *Ejemplos de Stride y Padding.*



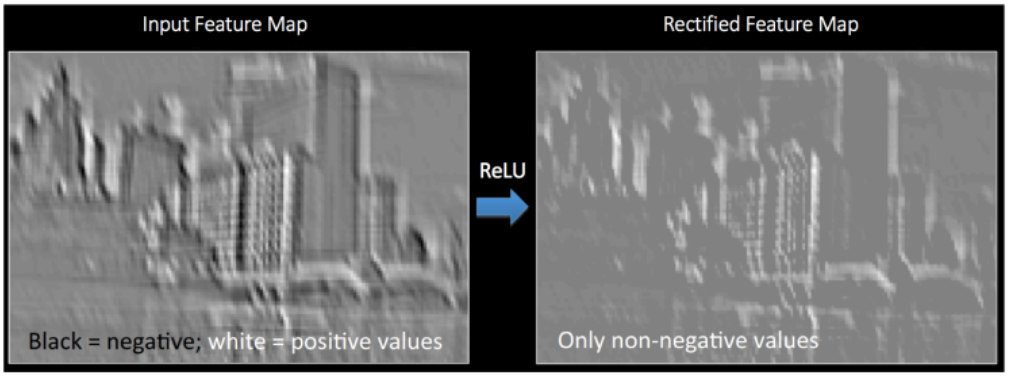
Fuente: *[Dumoulin and Visin, 2016]*

Figura 35: *Aplicación de la operación convolución sobre una imagen.*



Fuente: *[Ujjwalkarn, 2016a]*

Figura 36: *Aplicación de la función de activación Relu.*

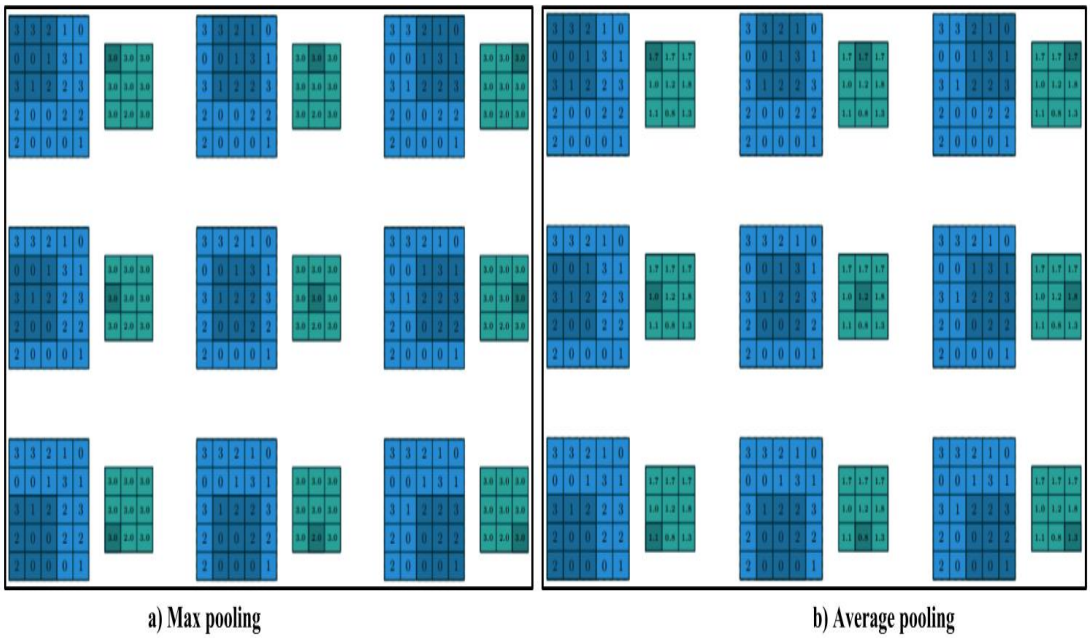


Fuente: *[Ujjwalkarn, 2016a]*

2.5.1.2. Pooling

La función de pooling, Submuestreo o Subsampling se utiliza para modificar los *feature maps*. Las operaciones de pooling reducen el tamaño de los *feature maps* utilizando alguna función para reducir subregiones. Pooling funciona deslizando una ventana a través de los *feature maps* y aplica una función de agrupación sobre la vecindad seleccionada. Las funciones más utilizadas son *max pooling, average pooling y min pooling* [Goodfellow et al., 2016]. En la figura 37.*a* y 37.*b* se muestra el funcionamiento de la función *max pooling* y *average pooling* respectivamente. Mientras tanto, se definen algunas funciones de agrupación.

Figura 37: *Ejemplos de max y average pooling.*

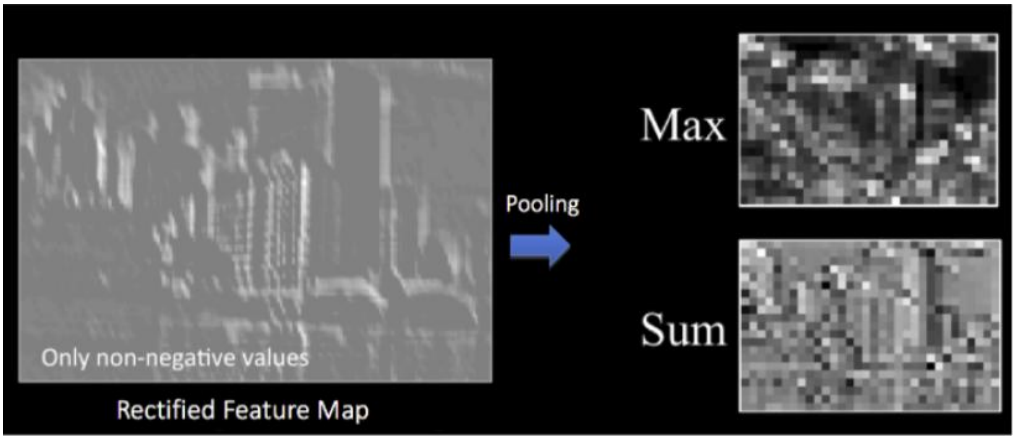


Fuente: *[Dumoulin and Visin, 2016]*

- ***Max Pooling*** Obtiene el píxel de mayor valor en una vecindad.
- ***Min Pooling*** Obtiene el píxel de menor valor en una vecindad.
- ***Average Pooling*** Obtiene el píxel de valor promedio den una vecindad.
- ***Sum Pooling*** Calcula la suma de todos los píxeles de una vecindad.

Finalmente, la aplicación de los filtros *Max Pooling* y *Sum Pooling* sobre una imagen se visualizan en la Figura 38.

Figura 38: *Aplicación de Max Pooling y Sum Pooling.*



Fuente: *[Ujjwalkarn, 2016a]*

2.5.1.3. Capa Fully-Connected

La capa Fully-Connected es un Perceptron Multicapa (MLP) estudiado en la Subsección 2.4.2 que utiliza una función de activación *Softmax* en la capa de salida. Sin embargo, también se pueden utilizar otros clasificadores. La capa Fully-Connected implica que cada neurona en la capa anterior está conectada a cada neurona en la siguiente capa. A continuación, se muestra la representación de *Softmax* [LeCun et al., 1998].

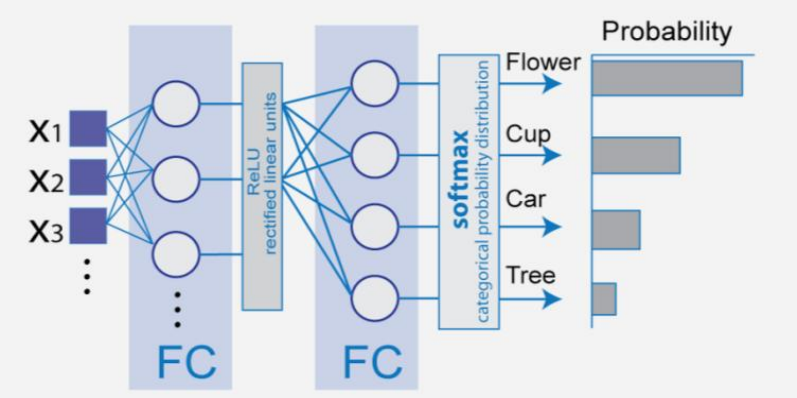
$$P(y = j|z^{(i)}) = \phi_{softmax}(z^{(i)}) = \frac{\mathrm{e}^{z^{(i)}}}{\sum_{j=0}^{k} \mathrm{e}^{z_k^{(i)}}} \tag{41}$$

Donde z es equivalente a:

$$z = w_0 x_0 + w_1 x_1 + ... + w_m x_m = \sum_{l=0}^{m} w_l x_l = w^T x \tag{42}$$

Donde w es el vector de pesos, x es el vector de características de 1 muestra de entrenamiento y w_0 es la unidad de bias). Ahora, la función *Softmax* calcula la probabilidad de pertenencia de la muestra de entrenamiento $x(i)$ a la clase j dado el peso y la entrada neta $z(i)$. Por lo tanto, calculando la probabilidad $p(y = j|x(i), w_j)$ para cada etiqueta de clase en $j = 1, ..., k$. Por otro lado, en la Figura 39 se muestra la aplicación de la capa Fully-Connected, donde $x_1, x_2, x_3, ...$ son *feature maps* generados por la capa de pooling. Además, se visualizan dos capas fully-connected donde la ultima capa genera la probabilidad de pertenencia de una imagen a todas las clases, mientras que la otra capa utiliza una función de activación para evitar la linealidad de las neuronas.

Figura 39: *Aplicación de la capa Fully-Conneted.*



Fuente: *https://goo.gl/hgWZLn*

2.6. ImageNet Challenge

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) o *ImageNet Challenge* es una competencia de visión computacional, donde equipos de todo el mundo compiten para ver quién tiene el mejor modelo para resolver tareas de clasificación, detección, segmentación y otros. El objetivo es permitir que los investigadores comparen sus resultados durante los desafíos mencionados. En este reto se utiliza la base de datos *ImageNet*; esta base de datos esta compuesta de 14 millones de imágenes de alta resolución de 1000 categorías diferentes [Deng et al., 2009].

2.6.1. Arquitecturas propuestas

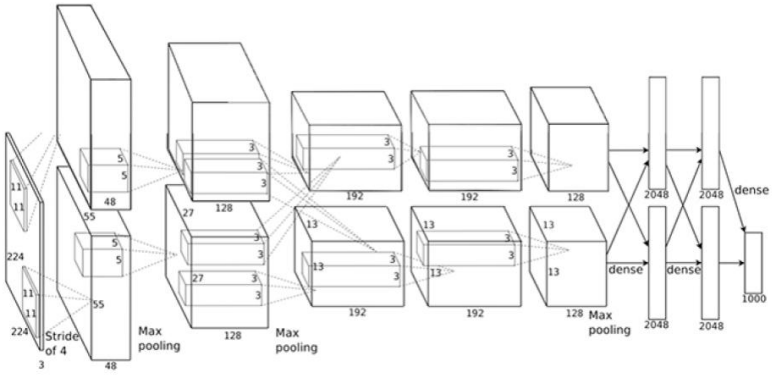
Durante el *ImageNet Challenge*, en específico durante la clasifición de imágenes se desarrollaron diversas técnicas, siendo los mas importantes aquellos basados en Redes Neuronales Convolucionales. A continuación se describen aquellas arquitecturas que obtuvieron una tasa de error mínima durante cada edición del desafío.

2.6.1.1. AlexNet

AlexNet [Krizhevsky et al., 2012] es una arquitectura de red neuronal convolucional basada en la arquitectura *Lenet-5* [LeCun et al., 1998]. Este trabajo es considerado como uno de los más influyentes en el campo de *Deep Learning*, por marcar un antes y un después en el reto de clasificación de imágenes de ILSVRC 2012, donde obtuvo el primer puesto con una tasa de error equivalente a 26.2 %. En la Figura 40 se muestra la arquitectura de la red. Donde Raw Image corresponde a imágenes sin preprocesamiento[25], FM es el número de *feature maps*, FCONV es el tamaño del filtro convolucional, FPOL es el tamaño del filtro de pooling. Por otro lado, es necesario informar que existen dos redes convolucionales porque utilizaron

[25]El preprocesamiento consiste en mejorar la imagen original, resaltando determinadas características de la imagen y eliminando otras [Sonka et al., 1993].

Figura 40: *Arquitectura de red neuronal convolucional AlexNet.*



Fuente: *https://goo.gl/frWZLn*

2 GPUs[26] en paralelo. Además, [Krizhevsky et al., 2012] entreno su modelo en seis días utilizando GPUs GTX 580 de 3GB. Finalmente, en la Tabla 4 se muestra a detalla la arquitectura de la red.

Tabla 4: *Arquitectura detallada AlexNet.*

Dimensión	Operación	Detalles
[227x227x3]	Imagen de Entrada	Raw Image
[55x55x96]	Convolución 1	96 FM, 11x11 FCONV, Stride 4, Padding 0
[27x27x96]	Max Pooling 1	3x3 FPOL, Stride 2
[27x27x96]	Normalización 1	Relu
[27x27x256]	Convolución 2	256 FM, 5x5 FCONV, Stride 1, Padding 2
[13x13x256]	Max Pooling 2	3x3 FPOL, Stride 2
[13x13x256]	Normalización 2	Relu
[13x13x384]	Convolución 3	384 FM, 3x3 FCONV, Stride 1, Padding 1
[13x13x384]	Convolución 4	384 FM, 3x3 FCONV, Stride 1, Padding 1
[13x13x256]	Convolución 5	256 FM, 3x3 FCONV, Stride 1, Padding 1
[6x6x256]	Max Pooling 3	3x3 FPOL, Stride 2
[4096]	Fully-Connected 1	4096 Neuronas
[4096]	Fully-Connected 2	4096 Neuronas
[1000]	Fully-Connected 3	1000 Neuronas(clasificación)

Fuente: *Elaboración propia*

2.6.1.2. Vgg Net

La red neuronal convolucional VGG [Simonyan and Zisserman, 2014] desarrollo modelos basados en la profundidad, siendo su aporte principal una evaluación de

[26]Una GPU*(Graphics Processor Uni)* es un coprocesador, dedicado al procesamiento gráfico. Las GPU están construidas de modo que sean mucho más eficiente para el cálculo de información gráfica.

redes cada vez más profundas que utilizan filtros convolucionales de 3 × 3, Stride 1 y Padding 1. Siendo la máxima profundidad 19 capas. Este trabajo obtuvo el primer y segundo lugar en el desafío de detección y clasificación de imágenes del ILSVRC 2014 respectivamente; en el proceso de clasificación obtuvo una tasa de error equivalente a 7.4 %. En la Tabla 5 se muestra las diferentes configuraciones de VGG Net (mostradas por columnas). La profundidad de las configuraciones aumenta desde la izquierda (A) hacia la derecha (E), a medida que se añaden más capas (las capas añadidas se muestran en negrita). Donde conv representa a la capa de convolución y FC representa a la capa *fully-connected*, mientras que la función de activación ReLU no se muestra por brevedad. Siendo la mayor precisión la red neuronal de 16 capas (columna D). Por otro lado, cada arquitectura fue entrenada en 4 GPUs Nvidia Titan Black durante dos o tres semanas dependiendo del modelo.

Tabla 5: *Configuración de las Arquitecturas VGG.*

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architecures of VGG Net. Configuration D produced the best results

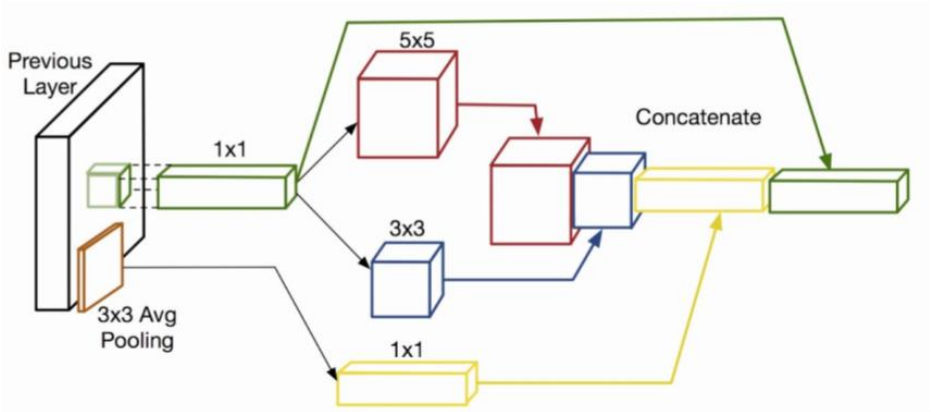
Fuente: *[Simonyan and Zisserman, 2014]*

2.6.1.3. GoogleNet

A diferencia de sus predecesoras GoogleNet o *Inception-V1* [Szegedy et al., 2015] introduce el concepto de paralelismo entre capas convolucionales, en específico introduce el modulo *inception*. Esta fue una de las primeras arquitecturas CNN que se desvió del enfoque general de simplemente apilar capas de convolución y pooling, en específico estructuras secuenciales. GoogleNet tiene 22 capas convolucionales y fue la arquitectura ganadora del ILSVRC 2014 con una tasa de error de 6,7 %. Esta arquitectura tiene la característica de apilar módulos *inception*.

En GoogleNet se analizaron formas de reducir la carga computacional de las redes neurales convolucionales, mientras se obtenía un desempeño elevado. Un módulo *inception* calcula múltiples transformaciones diferentes sobre el mismo *feature map* en paralelo, concatenando sus resultados en una sola salida (Ver Figura 41).

Figura 41: *Concatenación de las convoluciones del módulo inception.*



Fuente: *[Tripathy, 2016]*

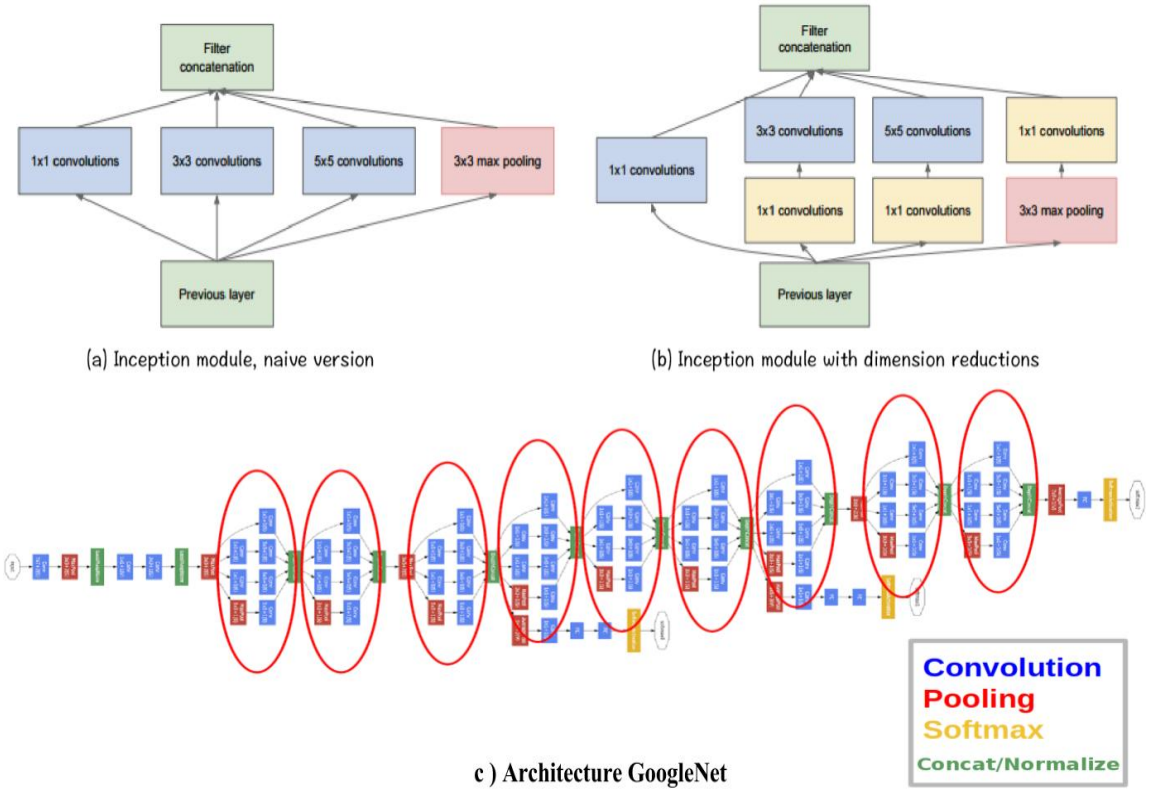
Sin embargo, suponiendo que hay M *feature maps* y N filtros convolucionales, aplicando la operación de convolución $N * M$, se obtiene un aumento cuadrático de la computación (Ver Figura 42.*a*). Para solucionar este problema se aplica filtros convolucionales de 1×1 para reducir de la dimensionalidad, y sobre estos *featu maps* generados se aplican filtros convolucionales de 3×3 y 5×5, además se aplica un filtro pooling de 3×3 seguido de un filtro de covolución de 1×1 como se muestra en la Figura 42.*b* [Szegedy et al., 2015]. Por otro lado, en la Figura 42.*c* se muestra la arquitectura GoogleNet completa y los módulos *inception* encerrados en un círculo rojo. Este modelo se entreno durante una semana en los servidores de Google.

Finalmente, la distribución de GoogleNet es utilizar 9 módulos *Inception* seguidos de un proceso de *Dropout*[27] del 40 % de los datos, seguida de una capa lineal[28] y una función de activación *Softmax*.

[27]Esta técnica eliminan algunas neuronas al azar para reducir el *overfitting*. Es decir, el sobre entrenamiento de los datos.

[28]Esta capa permite adaptar fÃ¡cilmente la red neuronal a otros conjuntos de etiquetas

Figura 42: *Arquitectura GoogleNet y módulos inception.*



Fuente: *[Szegedy et al., 2015]*

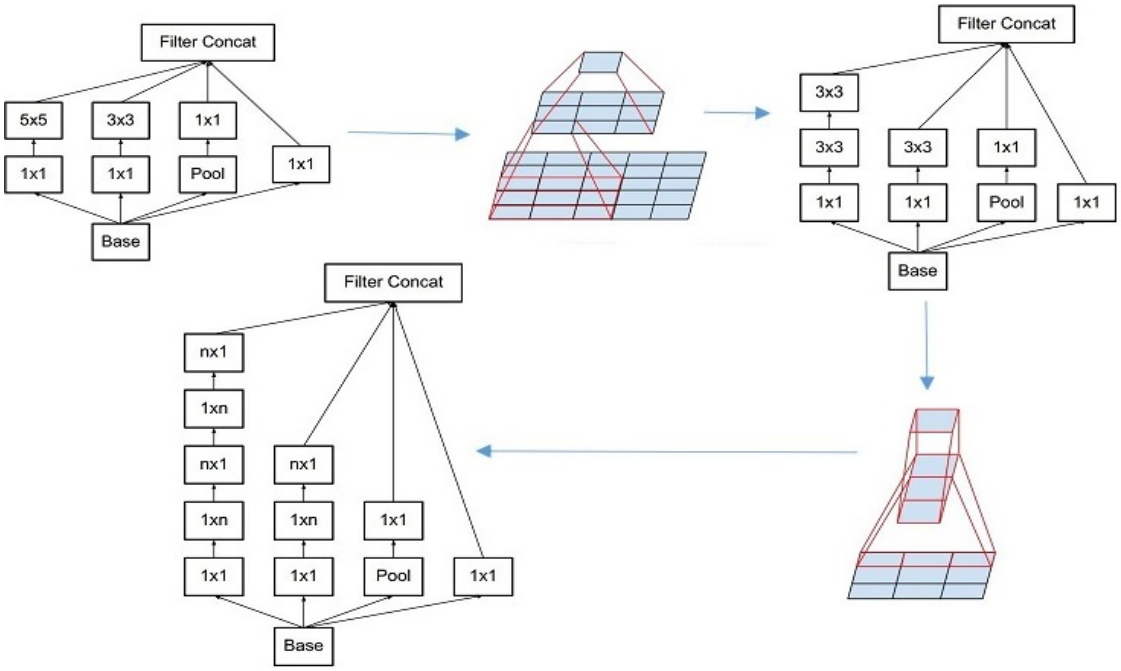
2.6.1.4. Inception-V3

La arquitectura *Inception-V3* [Szegedy et al., 2016] es una modificación de la arquitectura *Inception-V1*. Esta CNN tiene la característica de maximizar el flujo de información en la red, construyendo cuidadosamente una red que equilibre la profundidad y el ancho. Esta CNN tiene 48 capas y obtuvo un error de 3,5 % en la base de datos de *ImageNet*. En la Figura 43 se muestra el funcionamiento de la arquitectura, donde la primera imagen (extremo izquierdo) es el módulo *Inception-V1*, este presenta filtros convoluciones de 5×5, pero estos filtros tienden a ser desproporcionadamente caros en términos de cálculo. Por lo tanto, en la segunda imagen se muestra un filtro convolucional de 3×3 sobre el filtro de 5×5, luego se aplica un Fully-Connected sobre el filtro convolucional 3×3 y se obtiene una salida de 1×1, deslizando el filtro convolucional 3×3 sobre todo el filtro 5×5, se consigue como resultado dos filtros convolucionales de 3×3, en lugar de uno de 5×5. Sin embargo, existe la pregunta, "Un filtro puede ser más pequeño, por ejemplo 2×2". Según [Szegedy et al., 2016] se pueden generar filtros de $n \times 1$. Por ejemplo, usando una convolución de 3×1 seguida por una convolución de 1×3 es equivalente a deslizar una red de dos capas con el mismo campo receptivo[29] que

[29]El campo receptivo en una red neuronal convolucional se refiere a la parte de la imagen que es visible para un filtro convolucional

en una convolución de 3 × 3 como se muestra en la siguiente parte de la imagen, esta solución proporciona un 33 % de reducción computacional. En teoría se puede definir que un filtro de $n \times n$ puede ser reemplazado por un filtro de 1 $\times n$ seguido por uno de $n\times$ 1 como se muestra en la última imagen.

Figura 43: *Nuevo módulo Inception-V3.*



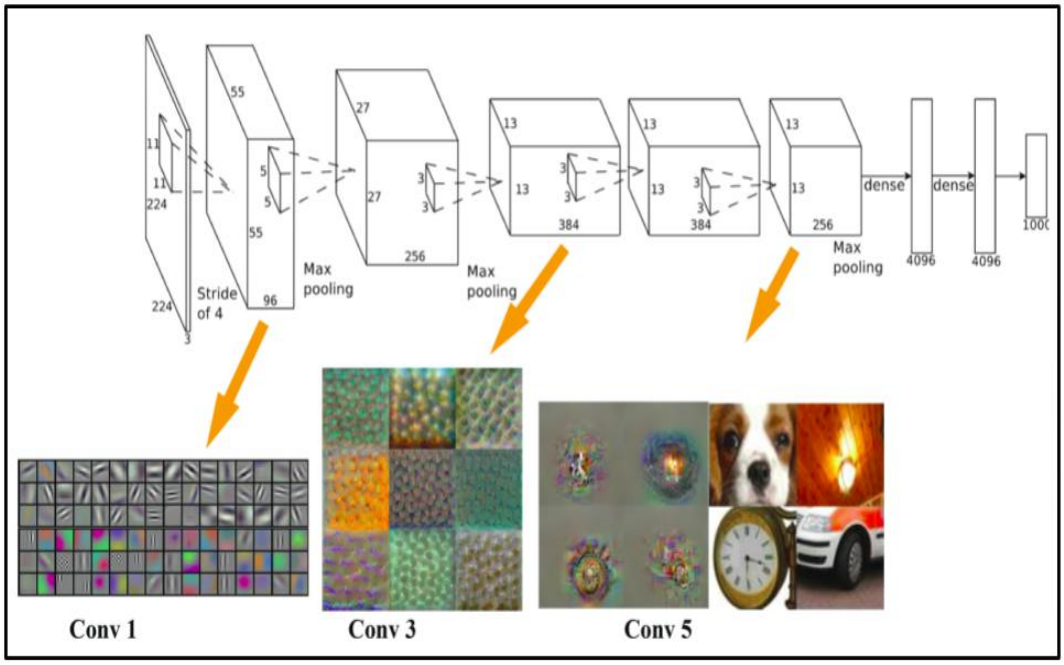
Fuente: *[Szegedy et al., 2016]*

2.7. Transfer Learning

Transfer Learning, aprendizaje de transferencia, meta learning o transferencia de parámetros, es una estrategia que ha sido muy apreciada en los últimos años por la comunidad de visión computacional para diseñar redes neuronales profundas. *Transfer Learning* es el proceso de tomar un modelo previamente entrenado (pesos y parámetros de una red que ha sido entrenada en un gran conjunto de datos por otra persona; este conjunto original de datos debe ser similar al nuevo conjunto de datos) y se ajusta el modelo(*Fine-Tuning*) con una base de datos personalizada, este modelo pre-entrenado actuará como un extractor de características de la nueva base de datos [Akilan et al., 2017]. Las CNNs modernas presentan un curioso fenómeno cuando se trabaja sobre imágenes, todas tienden a presentar características similares en las primeras capas de convolución, esto es demostrado durante el estudio de [Krizhevsky et al., 2012], estas particularidades se muestran en la Figura 44, donde **Conv 1** extrae características simples como lineas, **Conv 3** extrae características como formas. Finalmente, en **Conv 5** se extrae características más complejas como partes de la imagen. Esta particularidad permite

utilizar imágenes de bases de datos diferentes [Yosinski et al., 2014], debido a que comparten características similares.

Figura 44: *Visualización de los datos de las capas convolucionales.*



Fuente: *[Krizhevsky et al., 2012]*

Finalmente, las ventajas de utilizar esta técnica son:

- Reducción del tiempo de entrenamiento.
- Mitigar el hardware especializado.
- Inicialización de los parámetros (pesos y bias)

Parte III

Desarrollo del Proyecto

Capítulo 3

Método Propuesto

Antes de empezar, es necesario indicar que las imágenes de entrada corresponden a la categoría *data raw*. Es decir, imágenes sin preprocesamiento. Se decidió optar por este tipo de datos para emular el funcionamiento de sistemas reales, donde el tiempo de respuesta de una consulta debe ser inmediata como se hace en [Shcherbatov et al., 2017] o [Bezak, 2016]. Por consiguiente para tratar de afrontar un escenario de la vida real, el único preprocesamiento aplicado es el redimensionamiento de las imágenes.

3.1. Descripción de las fases

El reconocimiento de imágenes de edificios históricos de la ciudad del Cusco se resuelve a partir de dos fases:

- La primera fase (Entrenamiento y construcción del modelo) tiene por objetivo aprender las particularidades de una imagen de un edificio histórico de la ciudad del Cusco.
- La segunda fase (Predicción) tiene por objetivo pronosticar a que categoría pertenece una imagen de consulta.

Por lo general, los problemas de clasificación de imágenes pasan por una etapa previa conocida como detección de la región de interés (ROI). Sin embargo, trabajos previos como [Cho and Kim, 2016], [Torii et al., 2013] o [Bezak, 2016] omiten este proceso, debido a que consideran que el problema del reconocimiento de imágenes de edificios tiene la particularidad de tomar como región de interés no solo una parte de la imagen, sino todo el *background*(fondo de la imagen). Por lo tanto, en este trabajo se omite el proceso de detección. A continuación se detalla el funcionamiento de cada fase.

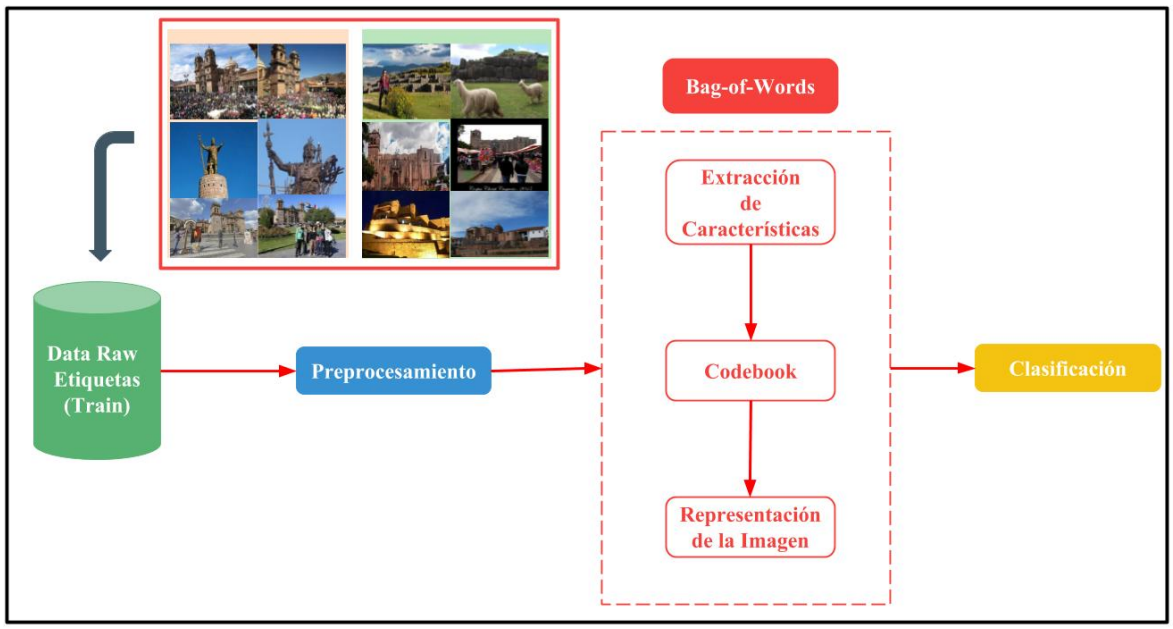
3.1.1. Entrenamiento y construcción del modelo

En esta etapa se extraen todas las características relevantes de una imagen y estas son cuantificadas a través de la construcción de un modelo clasificador. Esta fase se resuelve a partir de dos técnicas como son *Bag-of-Words* o *redes neuronales convolucionales*.

3.1.1.1. *Bag-of-Words*

Antes de empezar con esta etapa, se realiza un preprocesamiento, las imágenes son redimensionadas a un tamaño de 300 × 300. A partir de esta entrada se utiliza el modelo *Bag-of-Words*. Este se divide en 3 fases: Extracción de características, construcción del *codebook* y representación de la imagen, como se muestra en la Figura 45. Finalmente se realiza un proceso de clasificación.

Figura 45: *Entrenamiento y construcción del modelo utilizando Bag-of-Words.*



Fuente: *Elaboración propia*

3.1.1.1.1. Extracción de Características

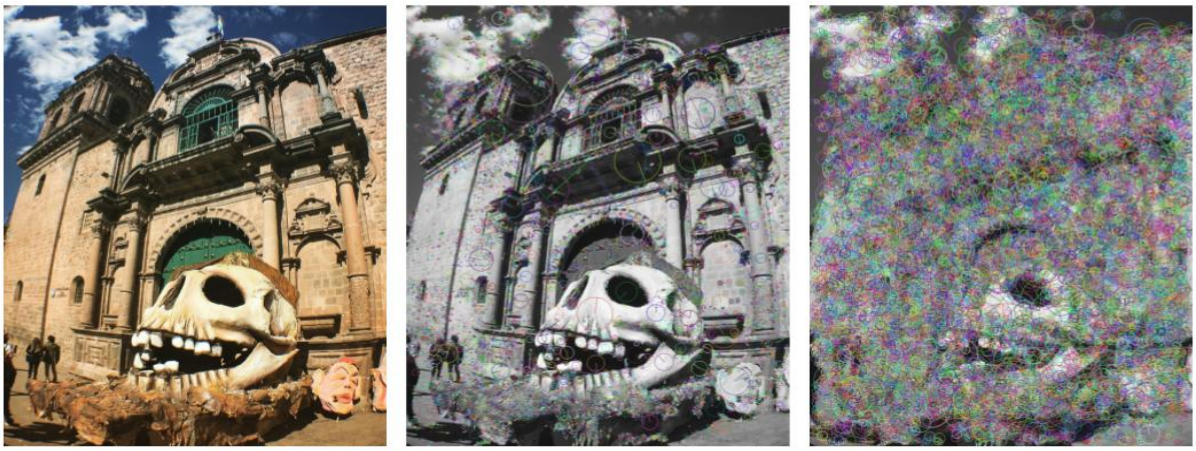
El objetivo de esta fase es detectar y describir los puntos de interés sobre una imagen, esta etapa puede ser solucionada a partir de múltiples algoritmos. En este trabajo se evalúan los algoritmos SIFT [Lowe, 2004] y SURF [Bay et al., 2006], cada uno por separado. La salida final de ambos extractores es un vector de características (*feature vector*).

Para extraer las características se siguen los siguientes pasos. Primero, se desarrolla la etapa conocida como detección, en esta se seleccionan varios puntos de interés dentro de la imagen. Segundo, se desarrolla la descripción, se calculan los descriptores locales sobre los puntos de interés previamente detectados. Finalmente, para generar el vector de características, se concatenan los descriptores de la imagen, como se muestra a continuación.

$$\mathrm{H} = (\mathrm{h}_1, \mathrm{h}_2, ..., \mathrm{h}_n) \tag{43}$$

Donde H corresponde al vector de características y h_i corresponde a los descriptores de la imagen $\forall\ i \in 1, ..., n$. Se puede ver un ejemplo de los descriptores SIFT y SURF aplicados sobre una imagen en la Figura 46.

Figura 46: *Descriptores de características de una imagen.*

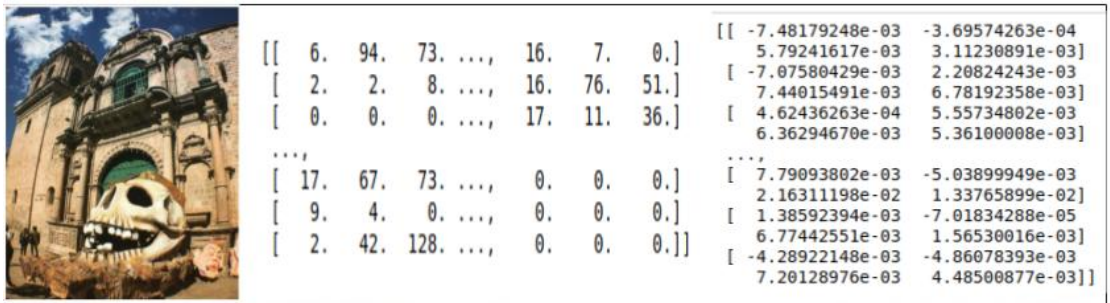


(a) Imagen original (b) SIFT (c) SURF

Fuente: *Elaboración propia*

De forma similar, se puede ver un ejemplo del vector de características SIFT y SURF aplicados sobre una imagen en la Figura 47.

Figura 47: *Vectores de características de una imagen.*



(a) Imagen original (b) SIFT (c) SURF

Fuente: *Elaboración propia*

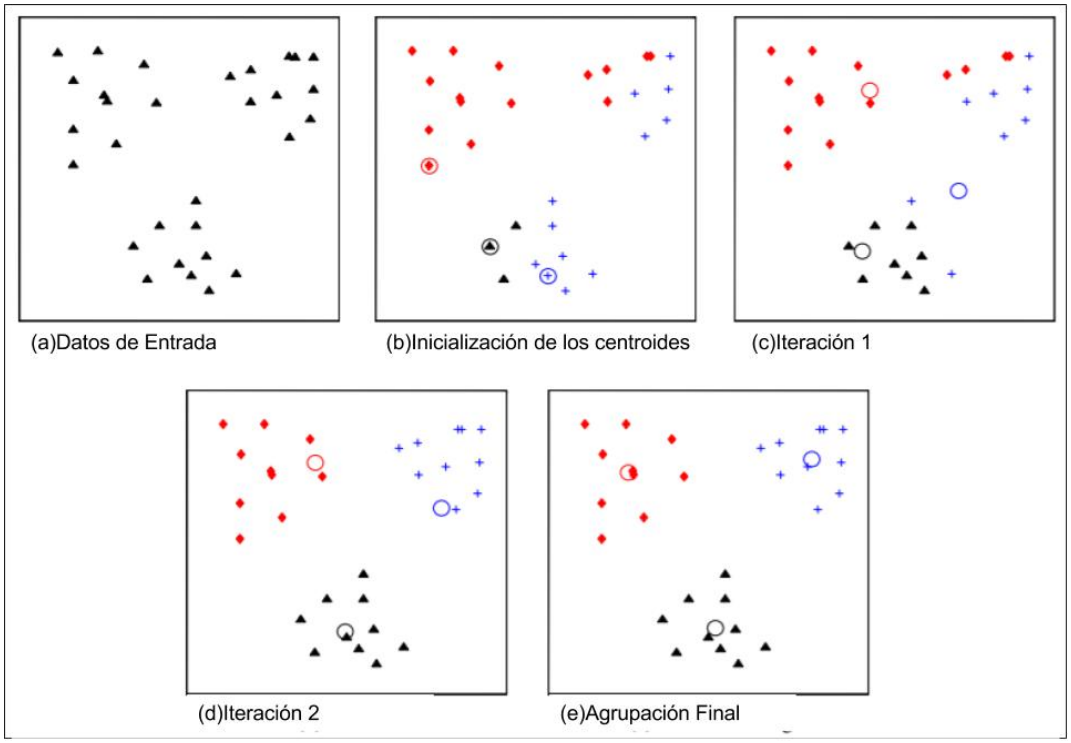
3.1.1.1.2. Codebook

El objetivo del *codebook* es encontrar las palabras visuales más representativas de cada clase. Para su construcción se toma como entrada los vectores de características extraídos en la etapa anterior, mientras que para identificar las palabras visuales más representativas se utiliza un algoritmo de agrupación, el más utilizado es el algoritmo *k-means*, siendo los centroides de cada *cluster* la palabra visual más representativa de cada clase.

Un ejemplo de la construcción del *codebook* utilizando *k-means* se muestra en la Figura 48. Donde (a) corresponde a los vectores de características. (b) representa

a la inicialización de los centroides aleatoriamente. Mientras que (c) y (d) calculan la distancia entre cada punto y los centroides de la agrupación. Finalmente, en (*e*) se calcula la agupación final.

Figura 48: *Construcción del codebook utilizando K-means.*



Fuente: *[Jain, 2010]*

Sin embargo, utilizar *k-means* para construir el *codebook* es un problema, en específico por el elevado costo computacional.

$$\mathrm{O(i \times n \times k \times d)} \tag{44}$$

Donde i, k, d y n repesentan el número de iteraciones, número de palabras visuales, la dimensión de los vectores de características y el número de vectores de características, respectivamente [Jain, 2010]. Una técnica utilizada para minimizar el coste computacional es la programación paralela. Como lo hace [Zhang et al., 2011], donde propone un algoritmo de agrupación paralelo para *K-means* con *MPI (Message Passing Interface)*, llamado *MKmeans*. El algoritmo permite generar agrupaciones de manera efectiva en el entorno paralelo.

Sin embargo, para afrontar este problema, en este trabajo se utiliza un enfoque basada en *stratified*, propuesto por [Sechidis et al., 2011]; la idea es simple, obtener una muestra aleatoria de cada clase de manera que represente de igual forma al conjunto de datos total. Donde la entrada es el conjunto de los vectores de características (*feature vectors*), las etiquetas de cada clase(*labels*), un número deseado de palabras visuales k y una muestra aleatoria representativa de cada clase (*Máximo número de imágenes por clase*). Como se muestra en el **Algoritmo 1**.

Algorithm 1 Stratified k-means

```
1: function STRATIFIED KMEANS(features, labels, CodebookSize,MaxImagesClass)
2:     SelectedIndicesPerLabel = []
3:     IndicesPerLabel = shuffle(features,labels)
4:     SelectedIndicesPerLabel = stratified(IndicesPerLabel, MaxImagesClass)
5:     k = CodebookSize / len(set(labels))
6:     codebook = kmeans(SelectedIndicesPerLabel, k)
7:     return codebook
```

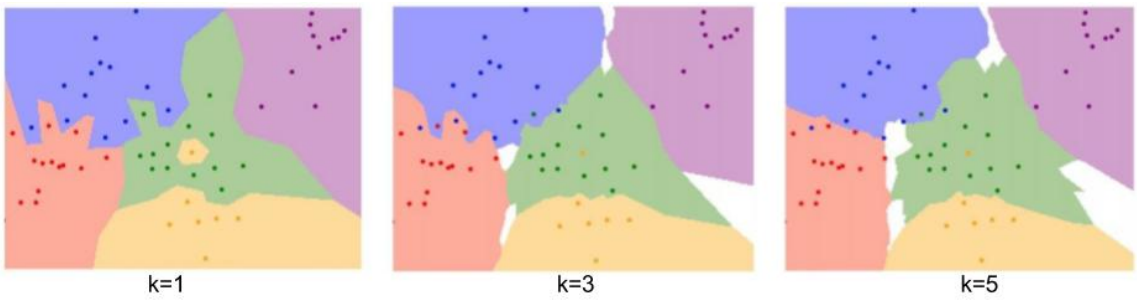
Donde *shuffle(features,labels)* representa el conjunto de *features vectors* desordenados de cada clase(*labels*), *stratified(IndicesPerLabel,MaxImagesClass)* representa a la muestra aleatoria deseada de cada clase, k representa el número de *clusters* y número de centroides a generar. Finalmente, se aplica el algoritmo de agrupación *k-means* en *kmeans(SelectedIndicesPerLabel,k)*, para extraer las palabras visuales más representativas de cada clase.

3.1.1.1.3. Representación de la Imagen

Esta fase tiene por objetivo construir un histograma que represente a la imagen. Donde X es un conjunto desordenado de descriptores extraídos de una imagen. $X = x_j, j \in 1, ..., N$, donde N es el número de descriptores en la imagen. Además, sea un *codebook* $C = c_m, m \in 1, ..., M$, donde M es el número de palabras visuales. Se genera $Z = z_w, w \in 1, ..., P$, donde P es la representación vectorial final de la imagen utilizada durante el proceso de clasificación. Para construir este vector se desarrollan dos fases:

- **Coding:** En esta etapa se asocian los descriptores de la imagen a los centroides más cercanos del *codebook*. En este trabajo se utiliza la técnica *kNN* para medir la distancia, en específico se utiliza un valor de $k = 1$, por presentar los mejores resultados a la hora clasificar un conjunto de datos, como se muestra en la Figura 49.

Figura 49: *Ejemplos de la aplicación de la técnica kNN.*

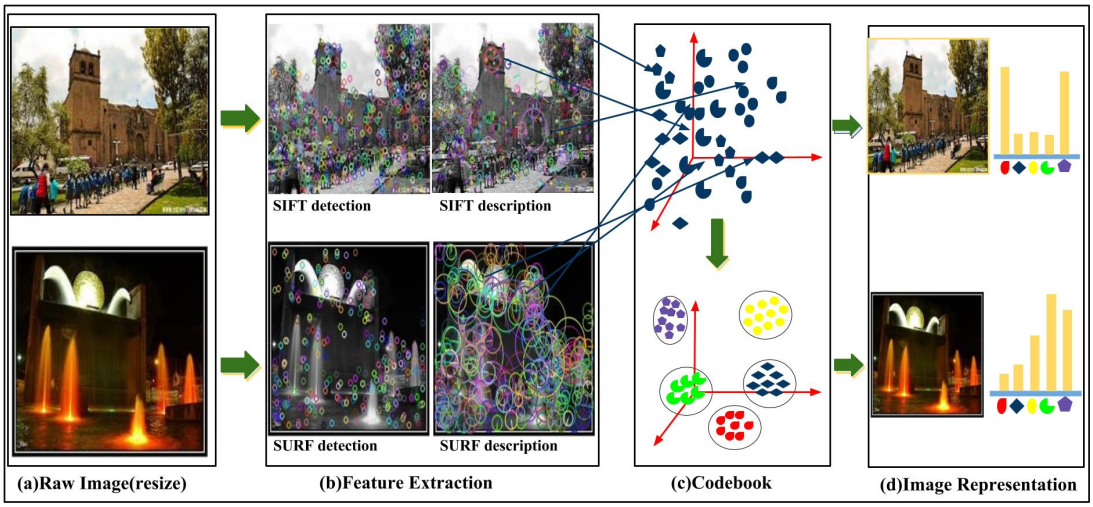


Fuente: *[Li, 2017]*

- **Pooling** La función *pooling* compacta toda la información en un único vector de características Z. Finalmente, el vector generado por *pooling* es el

histograma que representa a la imagen. Por otro lado, en la Figura 50 se muestra el desarrollo completo del modelo *Bag-of-Words*. Primero, se realiza la extracción de características utilizando *SIFT* o *SURF*. Segundo, se construye el *codebook*. Finalmente, se representa una imagen a partir de un histograma.

Figura 50: *Modelo Bag-of-Words desarrollado paso a paso.*



Fuente: *Elaboración propia*

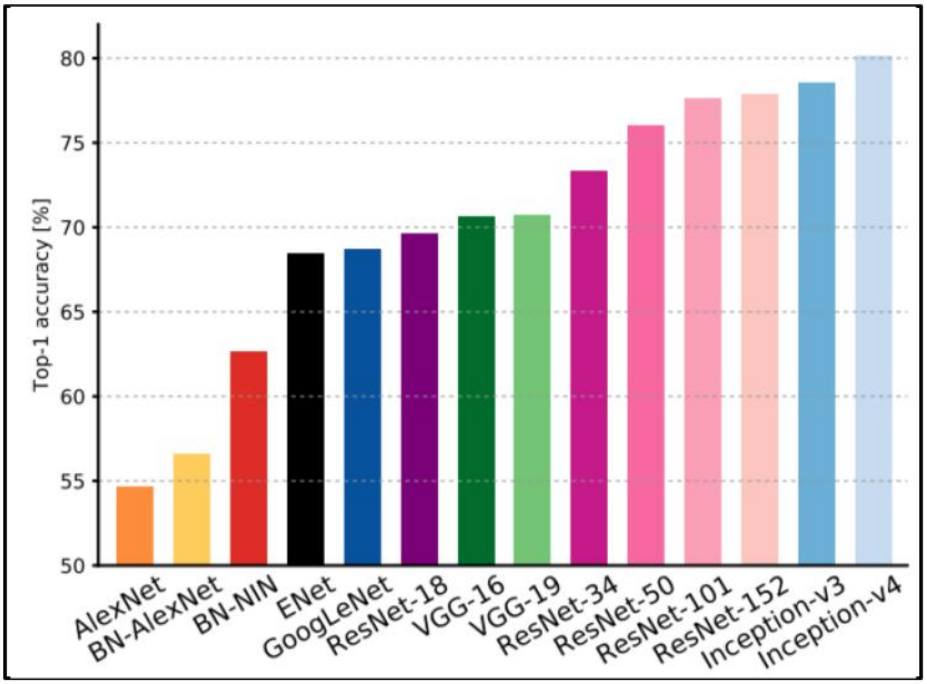
3.1.1.2. *Redes Neuronales Convolucionales (CNN)*

Además del modelo *Bag-of-Words*, en este trabajo se utiliza otra técnica durante la fase de Entrenamiento y Construcción del modelo, como son las *redes neuronales convolucionales*. Sin embargo, implementar una arquitectura CNN desde cero trae problemas como: Una fase de entrenamiento costosa, hardware especializado e inicialización de los parámetros. Por consiguiente, en este trabajo se utiliza una técnica conocida como *Transfer Learning*, esta permite utilizar un arquitectura de *red neuronal convolucional* previamente entrenada y ajustar el modelo con una base de datos personalizada. En otras palabras, el modelo pre-entrenado se utiliza como un extractor de características. En específico, en este trabajo se utilizan varias arquitecturas pre-entrenadas de *redes neuronales convolucionales* desarrolladas durante el *ImageNet Challenge*.

En este trabajo se seleccionaron 3 arquitecturas pre-entrenadas, como son: *VGG-16, VGG-19* e *Inception-V3*. En la Figura 51 se muestra la tasa de precisión de los modelos desarrollados para afrontar el *ImageNet Challenge*. Se seleccionaron los modelos previamente mencionados debido a que presentan resultados elevados durante este desafío. Por otro lado, los modelo *ResNet* e *Inception-V4* fueron omitidos por limitaciones computacionales.

La arquitectura diseñada consta de tres etapas para cada modelo pre-entrenado independientemente: preprocesamiento, extracción de características y clasificación. En la Figura 52 se muestra la arquitectura propuesta. Donde $[n, x]$ es el vector que representa a la imagen. Siendo n el número de imágenes que pasan por

Figura 51: *Top-1 de redes neuronales convolucionales.*



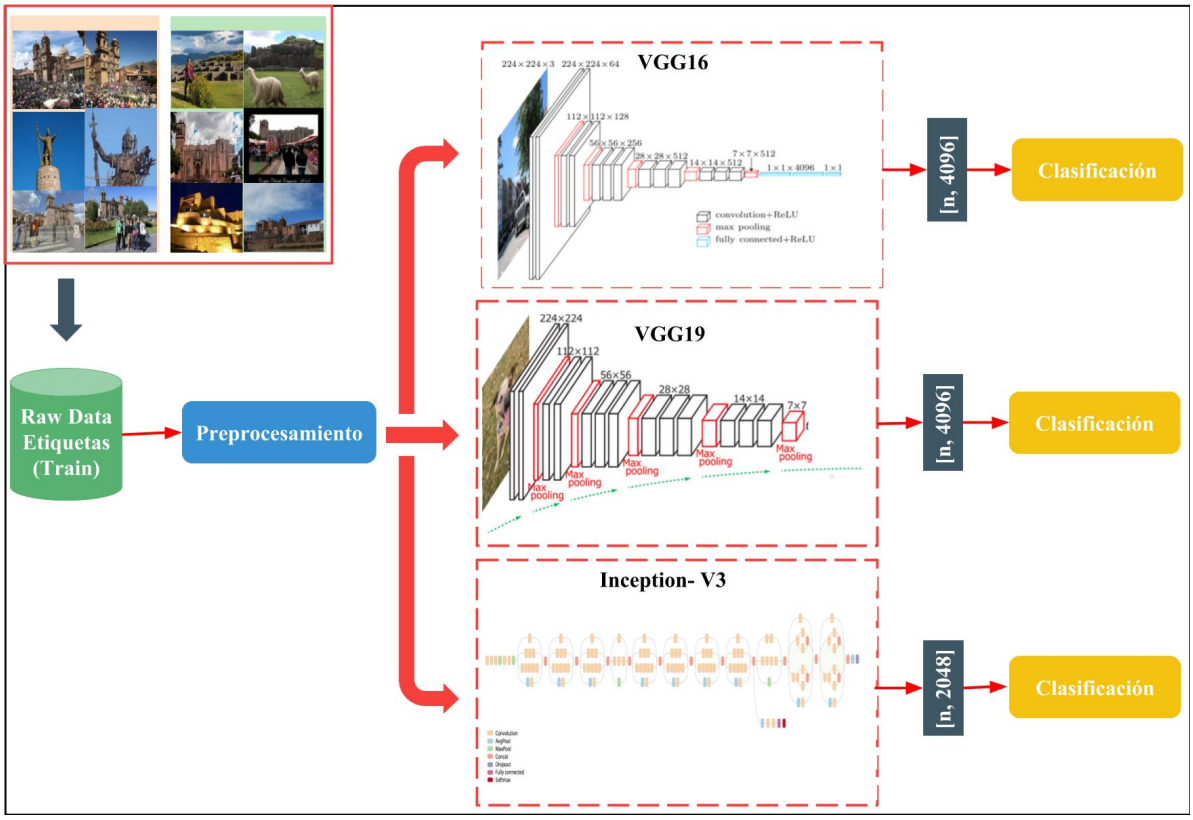
Fuente: *[Canziani et al., 2016]*

la CNN y x el número de características extraídas para cada imagen. Es necesario indicar que las ventajas más importantes de utilizar modelos previamente entrenados son la inicalización de los parámetros y el conocimiento obtenido mientras se entrena en otro conjunto de datos.

En específico el preprocesamiento consiste en redimensionar las imágenes a un tamaño de 224 × 224 de modo que sirvan para alimentar a los modelos *VGG-16, VGG-19* e *Inception-V3*. Por otro lado, con el fin de mejorar la precisión del método se utiliza una técnica muy conocida propia de las *redes neuronales convolucionales*, denominada como *data augmentation*. Esta genera nuevas imágenes a través de transformaciones sobre la imagen original, por ejemplo, rotación, recorte, traslación y otros, con el objetivo de generar un conjunto de imágenes más grande para entrenar el modelo. En particular, se aplica transformaciones como tinte, contraste, brillo y saturación. *Data augmentation* se utiliza con éxito en [Boominathan et al., 2016] y [Krizhevsky et al., 2012]. Sin embargo, en este trabajo solo se aplica esta técnica sobre la *red neuronal convolucional* con mejores resultados. Se opto por esta decisión por las limitaciones computacionales, el objetivo es analizar los resultados con y sin *data augmentation*. En la Figura 53 se muestra un ejemplo de esta técnica.

Por otro lado, la fase de extracción de características tiene por objetivo obtener los *transfer values* (salida de las redes neuronales) para cada modelo pre-entrenado, esta salida es equivalente al vector final en el modelo *Bag-of-Words*. Como se muestra a continuación.

Figura 52: *Transfer Learning utilizando los modelos VGG-16, VGG-19 e Inception-V3.*



Fuente: *Elaboración propia*

Figura 53: *Aplicación de la técnica data augmentation.*



Fuente: *[Chollet, 2016]*

3.1.1.2.1. Extracción de Características: VGG-16

Se toma como entrada una imagen previamente redimensionada y se introduce a la *red neuronal convolucional* VGG-16 (esta arquitectura se muestra en la Tabla 5 en específico en la columna (D)). Sin embargo, en este trabajo la imagen solo se procesa hasta antes de la segunda capa *Fully-Connected* (FC-4096), como se muestra en la Tabla 6.

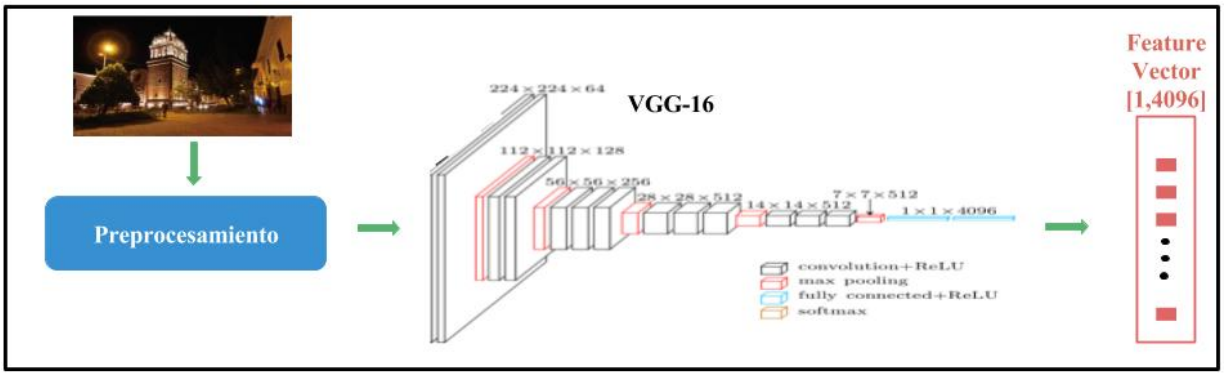
Tabla 6: *Arquitectura VGG-16 utilizada para el Transfer Learning, la función de activación ReLU no se muestra para mayor brevedad. Donde FM, FCONV y FPOL representa los feature maps, el tamaño de filtros de convolución y el tamaño de los filtros de pooling respectivamente.*

Dimensión de la matriz	Operación	Detalles
[224x224x3]	Imagen de Entrada	Image resize
[224x224x64]	Convolución 1	64 FM, 3x3 FCONV, Stride 1
[224x224x64]	Convolución 2	64 FM, 3x3 FCONV, Stride 1
[112x112x64]	Max Pooling 1	2x2 FPOL, Stride 2
[112x112x128]	Convolución 3	128 FM, 3x3 FCONV, Stride 1
[112x112x128]	Convolución 4	128 FM, 3x3 FCONV, Stride 1
[56x56x128]	Max Pooling 2	2x2 FPOL, Stride 2
[56x56x256]	Convolución 5	256 FM, 3x3 FCONV, Stride 1
[56x56x256]	Convolución 6	256 FM, 3x3 FCONV, Stride 1
[56x56x256]	Convolución 7	256 FM, 3x3 FCONV, Stride 1
[28x28x256]	Max Pooling 3	2x2 FPOL, Stride 2
[28x28x512]	Convolución 8	512 FM, 3x3 FCONV, Stride 1
[28x28x512]	Convolución 9	512 FM, 3x3 FCONV, Stride 1
[28x28x512]	Convolución 10	512 FM, 3x3 FCONV, Stride 1
[14x14x512]	Max Pooling 4	2x2 FPOL, Stride 2
[14x14x512]	Convolución 11	512 FM, 5x5 FCONV, Stride 1
[14x14x512]	Convolución 12	512 FM, 5x5 FCONV, Stride 1
[14x14x512]	Convolución 13	512 FM, 5x5 FCONV, Stride 1
[7x7x512]	Max Pooling 5	2x2 FPOL, Stride 2
[1x1x4096]	Fully-Connected 1	4096 Neuronas

Fuente: *Elaboración propia*

Finalmente, la salida sera un vector de 4096 características para cada imagen. Como se muestra en la Figura 54. Este vector se utilizara de entrada para el módulo de clasificación.

Figura 54: *Vector de características extraído a partir del modelo VGG-16.*

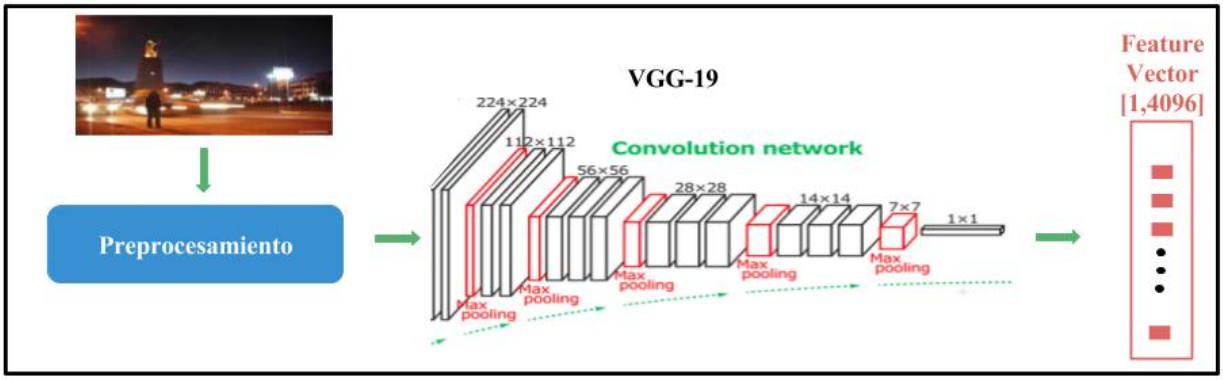


Fuente: *Elaboración propia*

3.1.1.2.2. Extracción de Características: VGG-19

La segunda CNN es la arquitectura VGG-19 (Tabla 5, consta de 19 capas, como se muestra en la columna (E)). Al igual que la arquitectura anterior, el proceso solo se realiza hasta antes de la segunda capa *Fully-Connected* (FC-4096). Por lo tanto, se obtendrá un vector de 4096 características para cada imagen. Como se muestra en la Figura 55

Figura 55: *Vector de características extraído a partir del modelo VGG-19.*



Fuente: *Elaboración propia*

Finalmente, la configuración de la *red neuronal convolucional* VGG-19 es similar a la anterior. Sin embargo, utilizar una arquitectura más profunda como se muestra en la Tabla 7, esta tiene por objetivo generar mejores resultados. El plus principal de utilizar las arquitecturas VGG es el hecho de utilizar filtros convolucionales de 3×3, estas tiene el mismo efectivo que un filtro convolucional de 7×7. Se opto por evaluar los modelos VGG-16 y VGG-19 debido a que en el *ImageNet Challenge*, la arquitectura VGG-16 obtuvo un mejor resultado frente a

VGG-19. Pero, en teoría la segunda arquitectura debería tener un mejor rendimiento, por el hecho de ser más profunda. Por lo tanto, en este trabajo se evalúan ambas arquitecturas en un escenario diferente; con el objetivo de identificar que arquitectura presenta mejores resultados.

Tabla 7: *Arquitectura VGG-19 utilizada para el Transfer Learning, la función de activación ReLU no se muestra para mayor brevedad. Donde FM, FCONV y FPOL representa los feature maps, el tamaño de filtros de convolución y el tamaño de los filtros de pooling respectivamente.*

Dimensión de la matriz	Operación	Detalles
[224x224x3]	Imagen de Entrada	Image resize
[224x224x64]	Convolución 1	64 FM, 3x3 FCONV, Stride 1
[224x224x64]	Convolución 2	64 FM, 3x3 FCONV, Stride 1
[112x112x64]	Max Pooling 1	2x2 FPOL, Stride 2
[112x112x128]	Convolución 3	128 FM, 3x3 FCONV, Stride 1
[112x112x128]	Convolución 4	128 FM, 3x3 FCONV, Stride 1
[56x56x128]	Max Pooling 2	2x2 FPOL, Stride 2
[56x56x256]	Convolución 5	256 FM, 3x3 FCONV, Stride 1
[56x56x256]	Convolución 6	256 FM, 3x3 FCONV, Stride 1
[56x56x256]	Convolución 7	256 FM, 3x3 FCONV, Stride 1
[56x56x256]	Convolución 8	256 FM, 3x3 FCONV, Stride 1
[28x28x256]	Max Pooling 3	2x2 FPOL, Stride 2
[28x28x512]	Convolución 9	512 FM, 3x3 FCONV, Stride 1
[28x28x512]	Convolución 10	512 FM, 3x3 FCONV, Stride 1
[28x28x512]	Convolución 11	512 FM, 3x3 FCONV, Stride 1
[28x28x512]	Convolución 12	512 FM, 3x3 FCONV, Stride 1
[14x14x512]	Max Pooling 4	2x2 FPOL, Stride 2
[14x14x512]	Convolución 13	512 FM, 5x5 FCONV, Stride 1
[14x14x512]	Convolución 14	512 FM, 5x5 FCONV, Stride 1
[14x14x512]	Convolución 15	512 FM, 5x5 FCONV, Stride 1
[14x14x512]	Convolución 16	512 FM, 5x5 FCONV, Stride 1
[7x7x512]	Max Pooling 5	2x2 FPOL, Stride 2
[1x1x4096]	Fully-Connected 1	4096 Neuronas

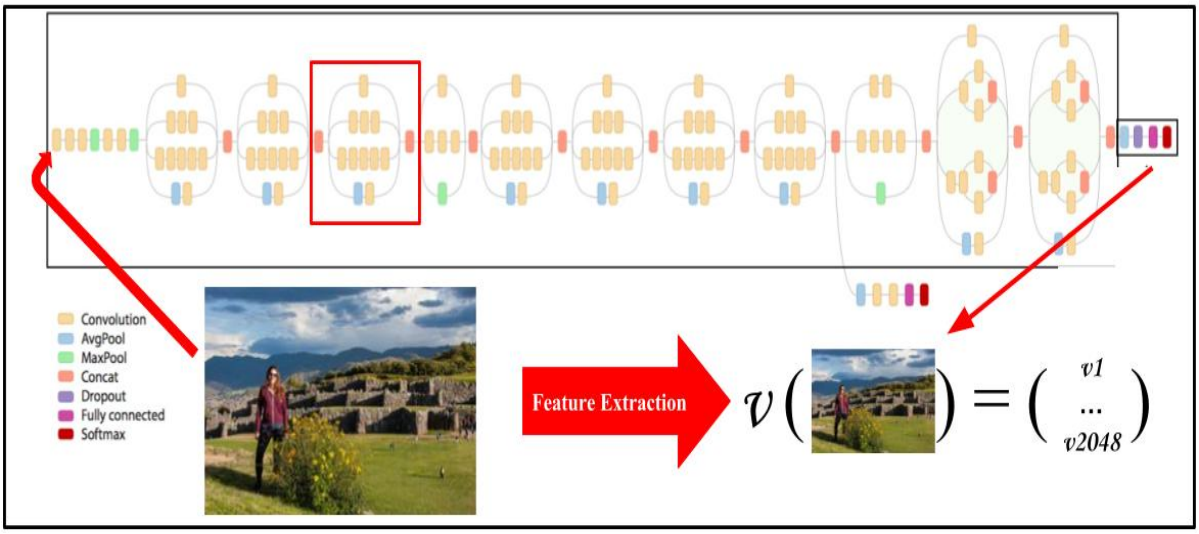
Fuente: *Elaboración propia*

3.1.1.2.3. Extracción de Características: Inception-V3

Este modelo se compone de varios módulos *Inception* (Ver Figura 43) conectados de forma secuencial. Cada módulo consta de varias operaciones de convolución y *pooling* en paralelo. Los resultados de estas operaciones se concatenan para construir un *feature vector* que represente a la imagen. Donde la dimensión del vector es equivalente a 2048 características. En la Figura 56 se muestra el proceso de extracción de características del modelo *Inception-V3*. El proceso se realiza hasta

antes de entrar al módulo de clasificación (*Softmax*). Es decir, se realiza el procesamiento hasta la capa *Fully-Connected*, esta devuelve el vector de características.

Figura 56: *Vector de características extraído a partir del modelo Inception-V3.*



Fuente: *Elaboración propia*

3.1.1.3. Clasificación

En este trabajo se utilizan 4 técnicas de aprendizaje de máquina (se puede ver a detalle el funcionamiento de estas técnicas en la Subsección 2.3) para medir el desempeño de los modelos *Bag-of-Words* y *redes neuronales convolucionales*, en un escenario de clasificación de imágenes de edificios históricos de la ciudad del Cusco.

Las técnicas de aprendizaje de máquina seleccionados en este trabajo son: *Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (kNN)* y *Neural Network (NN)*. Estos toman como entrada un conjunto de datos como son *feature vectors* y etiquetas. Estas técnicas tienen por objetivo realizar un proceso de aprendizaje, construcción del modelo y evaluación sobre un conjunto de datos de prueba. Es necesario mencionar que las salidas de los métodos *Bag-of-Words* y *redes neuronales convolucional* tienen las mismas características. Por lo tanto, se utiliza las mismas técnicas de aprendizaje de máquina durante este proceso. A continuación se describen los parámetros utilizados por cada técnica de aprendizaje de máquina:

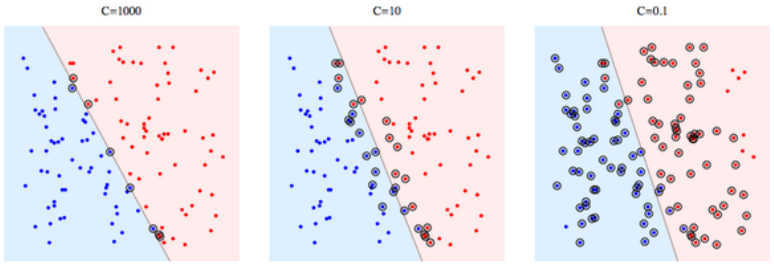
3.1.1.3.1. Clasificación utilizando Support Vector Machine

Para calcular de mejor forma los parámetros de SVM se realiza un proceso de búsqueda (*Grid Search*). Esta búsqueda se utiliza para generar los parámetros más óptimos de C y γ. Donde C define el costo de un error en la clasificación, además

controla la influencia de cada vector de soporte individual y γ es un parámetro del Kernel$_{rbf}$, que define hasta dónde llega la influencia de un único ejemplo de entrenamiento. Por otro lado, el Kernel permite calcular las transformaciones de los *feature vectors* sin que la dimensionalidad crezca exponencialmente. Siendo los más populares Kernel$_{linear}$ y Kernel$_{rbf}$.

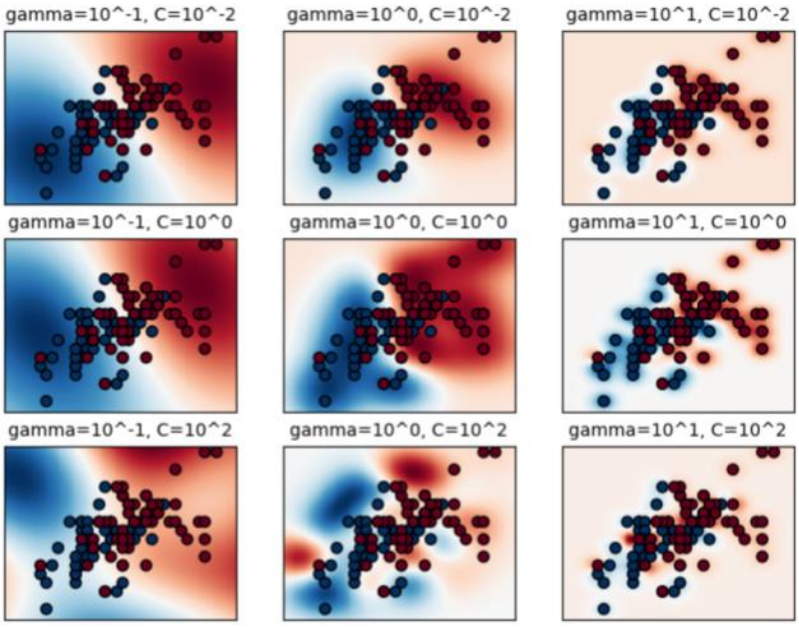
Donde un parámetro C bajo hace que la superficie de la decisión sea suave, mientras que un C alto busca clasificar correctamente todos los ejemplos, como se muestra en la Figura 57. Por otro lado, γ define cuanta influencia tiene un solo ejemplo de entrenamiento, como se muestra en la Figura 58. Cuanto más grande es γ, más cerca deben ser otros ejemplos [Pedregosa et al., 2011]. Los valores utilizados para los parámetros C y γ se detallan en la Tabla 8.

Figura 57: *Variación del parámetro C de la técnica SVM.*



Fuente: *[Buitinck et al., 2013]*

Figura 58: *Variación de los parámetros C y γ de la técnica SVM.*



Fuente: *[Buitinck et al., 2013]*

Tabla 8: *Optimización de los parámetros de SVM.*

Kernel RBF	
Parámetro	**Valor**
C	$10^i,\ i\ \in \{-1, 0, 1, 2\}$
γ	$10^i,\ i\ \in \{-1, 0, 1, 2\}$
Kernel Linear	
Parámetro	**Valor**
C	$10^i,\ i\ \in \{-1, 0, 1, 2\}$

Fuente: *Elaboración propia*

3.1.1.3.2. Clasificación utilizando Random Forest

Para calcular la técnica *Random Forest* es necesario asignar los parámetros principales de forma adecuada, estos son $n_estimators$ y max_depth. El primero corresponde al número de árboles. Cuanto mayor sea es mejor, pero también más tiempo se tardará en hacer los cálculos. Además, es necesario tener en cuenta que los resultados dejarán de mejorar significativamente más allá de un número crítico de árboles. Por otro lado, el parámetro max_depth controla la profundidad máxima del árbol [Pedregosa et al., 2011]. Los valores de estos parámetros se asignan en la Tabla 9.

Tabla 9: *Asignación de los parámetros para Random Forest.*

Random Forest	
Parámetro	**Valor**
$n_estimators$	500
max_depth	50

Fuente: *Elaboración propia*

3.1.1.3.3. Clasificación utilizando K-Nearest Neighbors

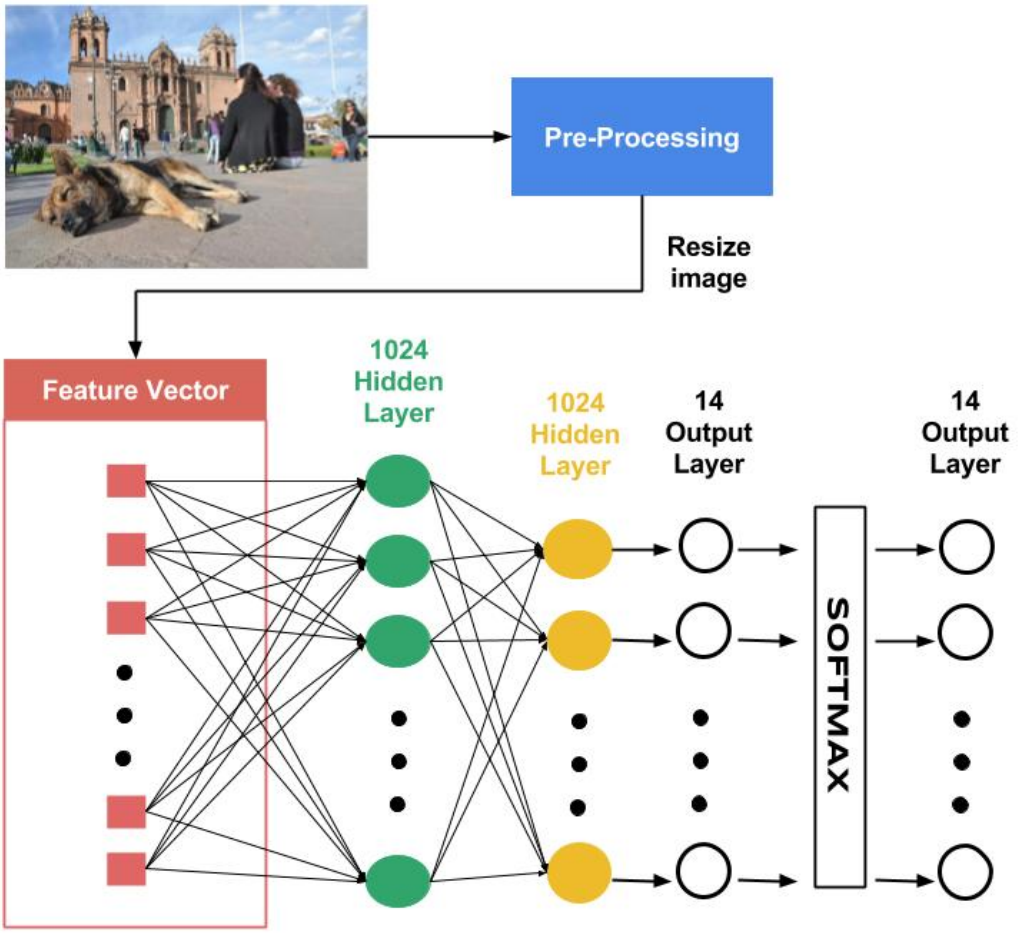
Esta técnica se basa en los k vecinos más cercanos de cada punto de consulta. La elección óptima del valor es altamente dependiente de los datos: En general, un valor alto para k genera efectos como el ruido, pero hace que los límites de clasificación sean menos distintos. Sin embargo, en este trabajo se utiliza un valor de $k = 1$, por los resultados generados en la Figura 49.

3.1.1.3.4. Clasificación utilizando Neural Network

La red neuronal diseñada toma como entrada *feature vectors*, esta contiene 1024 neuronas en la capa de entrada, 1024 neuronas en la capa oculta y 14 neuronas en la capa de salida. A continuación, se utiliza un función de activación, en específico *Softmax* para generar la probabilidad de pertenencia de la imagen a todas las clases. Para evaluar el rendimiento de este clasificador, se utiliza el algoritmo optimizador Adam Optimization (toda la teoría de este y otros conceptos se

muestra en la Sección 2.4) y como función de costo se utiliza la entropía cruzada para actualizar los pesos. El sistema fue entrenado durante 15000 iteraciones para actualizar los pesos del modelo de clasificación. Esta arquitectura de red neuronal se muestra en la Figura 59.

Figura 59: *Arquitectura propuesta de red neuronal.*



Fuente: *Elaboración propia*

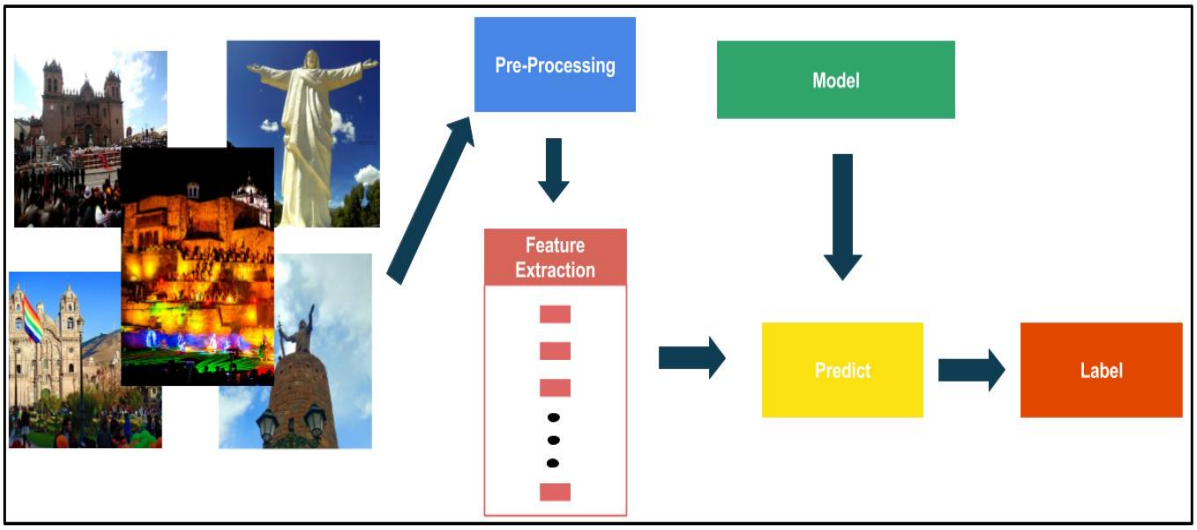
3.1.2. Predicción

En esta etapa se augura a que categoría de la base de datos pertenece una imagen de consulta, esta etapa se aplica sobre el modelo (*Bag-of-Words* o *redes neuronales convolucionales*) que obtuvo el mejor rendimiento durante la fase anterior (Entrenamiento y construcción del modelo). Debido a que los experimentos de esta fase serán más precisos si son aplicados sobre un modelo extractor de características acertado. Sin embargo, es necesario indicar que ninguna técnica presenta una precisión del 100 %. Por lo tanto, en la sección 4.2.3 se detalla porque una imagen de consulta realiza una predicción errónea.

En la Figura 60 se muestra la metodología propuesta para asignar una etiqueta a una imagen de consulta, en esta se visualiza las siguientes etapas:

- ***Pre-Processing.-*** En esta se toma como entrada una imagen externa a la base de datos y se realiza un proceso de redimensionamiento, las dimensiones dependen del modelo *Bag-of-Words* o *redes neuronales convolucionales*.
- ***Feature Extraction.-*** En esta fase se toma como entrada la imagen redimensionada, a esta se aplica un proceso de extracción de características, para obtener como salida final un *feature vector* o vector de características. La forma de extraer características de una imagen depende íntegramente del modelo (*Bag-of-Words* o *redes neuronales convolucionales*).
- ***Predict.-*** Esta fase toma dos entradas. Primero, el *feature vector* obtenido en la etapa previa. Segundo, el modelo entrenado ya sea *Bag-of-Words* o *redes neuronales convolucionales* y como salida se obtiene la etiqueta de la imagen.

Figura 60: *Arquitectura propuesta para identificar la etiqueta de una imagen externa a la base de datos.*



Fuente: *Elaboración propia*

Parte IV

Proceso Experimental

Capítulo 4

CuscoBID y Experimentos

4.1. Cusco Building Image Dataset (CuscoBID)

Cusco Building Database es el nombre del conjunto de datos creado como parte de este trabajo. Actualmente se encuentra en su segunda versión. Siendo la primera versión utilizada como un conjunto de prueba durante los experimentos del articulo científico derivado de este trabajo titulado *"Towards accurate building recognition using convolutional neural networks"*. Sustentado el 18 de Agosto del 2017 en el *IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*[30]. Este trabajo se centra en identificar la técnica de aprendizaje de máquina más óptima. Sin embargo, en el articulo científico agregado en el apéndice A, en específico en la parte denominada *"C. Classification with Rejection"*, se evalúa el escenario de rechazo. Es decir, se propone un método capaz de identificar que una imagen dada no pertenece a ninguna de las categorías definidas.

Las imágenes fueron recopiladas de internet para obtener diferentes condiciones fotométricas, Además, fue necesario capturar imágenes manualmente con problemas de rotación, ruido, escalamiento y otros (en específico imágenes con problemas descritos en la Subsección 1.1.1). El objetivo, fue obtener una base de datos desafiante, superando en cantidad y dificultad a bases de datos como ZuBuD[31] y SBID[32]. A continuación, se detallan las dos versiones de *Cusco Building Image Dataset (CuscoBID)*:

- **Primera Versión.-** La base de datos consta de 2000 imágenes de edificios históricos de la ciudad Cusco. Donde las imágenes de internet representan el 45 % de imágenes por clase, mientras que las imágenes capturadas manualmente representan el 55 % restante. La base de datos consta de 14 clases distribuidas como se muestra en la Tabla 10.

- **Segunda Versión.-** La base de datos consta de 4560 imágenes de edificios históricos de la ciudad Cusco. Donde las imágenes de internet representan

[30]El articulo científico [Farfan Escobedo, 2017] se encuentra indexado al *IEEE Xplore Digital Library*, puede acceder a través de: `http://ieeexplore.ieee.org/document/8079686/`

[31]Base de datos de edificios de Zúrich: `http://www.vision.ee.ethz.ch/showroom/zubud/`

[32]Base de datos de edificios de Sheffield: `https://www.sheffield.ac.uk/eee/research/iel/research/buildings_data`

Tabla 10: *Detalles de la primera versión de Cusco Building Image Dataset.*

Edificio Histórico	# Imágenes	Edificio Histórico	# Imágenes
Casa del Inca Garcilaso	108	Catedral del Cusco	159
La Compañia de Jesus	176	Coricancha	147
Cristo Blanco	146	Templo de la Merced	142
Mural de la Historia Inca	137	Paccha de Pumaqchupan	114
Pileta de San Blas	139	Inca Pachacutec	129
Sacsayhuaman	190	Iglesia de San Francisco	135
Iglesia de San Pedro	146	Iglesia de Santo Domingo	132

Fuente: *Elaboración propia*

el 60 % de imágenes por clase, mientras que las imágenes capturadas manualmente representan el 40 % restante. La base de datos consta de 14 clases distribuidas como se muestra en la Tabla 11.

Tabla 11: *Detalles de la segunda versión de Cusco Building Image Dataset.*

Edificio Histórico	# Imágenes	Edificio Histórico	# Imágenes
Casa del Inca Garcilaso	200	Catedral del Cusco	600
La Compañia de Jesus	600	Coricancha	570
Cristo Blanco	200	Templo de la Merced	260
Mural de la Historia Inca	200	Paccha de Pumaqchupan	200
Pileta de San Blas	320	Inca Pachacutec	300
Sacsayhuaman	350	Iglesia de San Francisco	250
Iglesia de San Pedro	280	Iglesia de Santo Domingo	230

Fuente: *Elaboración propia*

Finalmente, algunas imágenes de *Cusco Building Image Dataset* se muestran en la Figura 61.

Figura 61: *Algunas imágenes de la segunda versión de Cusco Building Image Dataset.*



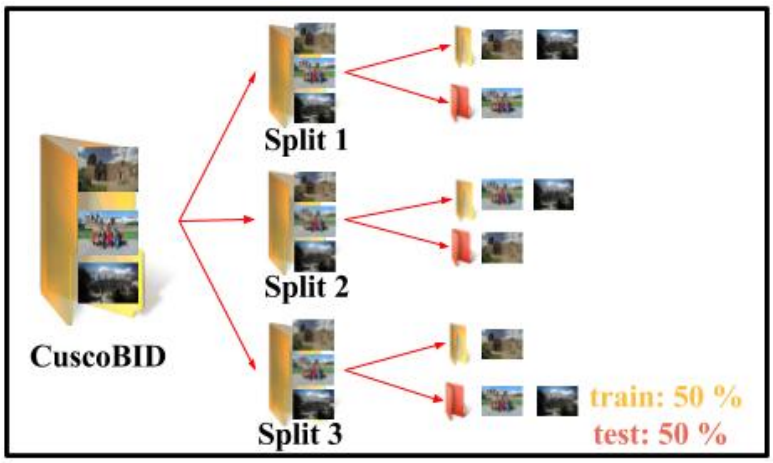
Fuente: *Elaboración propia*

4.2. Proceso Experimental

4.2.1. Distribución de la Base de Datos

Para solucionar el problema del reconocimiento de imágenes de edificios históricos de la ciudad del Cusco, se desarrollaron dos fases: Primero, entrenamiento y construcción del modelo, durante esta se desarrollo una distribución de la base de datos como se muestra en la Figura 62, donde la base de datos se dividió al azar tres veces: Con un 50 % de las muestras para el entrenamiento del clasificador y el 50 % restante para la evaluación como lo hace [Boominathan et al., 2016], a esta técnica se denominada *cross-validation*; esta tiene por objetivo garantizar que el modelo sea independiente de los datos, debido a que asegura que los datos de entrenamiento en algún momento se conviertan en datos de prueba o viceversa, como se muestra en la Figura 62.

Figura 62: *Distribución de la base de datos.*



Fuente: *Elaboración propia*

Esto significa que los modelos *Bag-of-Words* y *redes neuronales convolucionales* se ejecutaran 3 veces (una por cada carpeta generada previamente). Siendo el resultado final de cada modelo, el promedio de la suma de los resultados de cada carpeta. Además, es necesario indicar que la base de datos utilizada fue la segunda versión de *Cusco Building Image Dataset* (CuscoBID). La distribución de los datos de entrenamiento y prueba varían de trabajo en trabajo. Por ejemplo: En [Krizhevsky et al., 2012] utiliza una distribución de 75 % para el entrenamiento y 25 % para la evaluación. Por consiguiente, en este trabajo se exige al máximo a los modelos *Bag-of-Words* y *redes neuronales convolucionales*.

Por otro lado, la segunda fase corresponde a la predicción, durante esta se agregaron 30 imágenes de diferentes clases y se calculó su precisión.

Sin embargo, antes de empezar con los resultados generados, es necesario definir algunos conceptos claves durante esta etapa, como son:

- **Matriz de Confusión:** Es una tabla que se utiliza para describir el rendimiento de una técnica de aprendizaje de máquina durante la clasificación en un conjunto de datos de prueba, como se muestra en la Figura 63. Donde

Verdaderos Positivos (*True Positive (TP)*) son los valores positivos predichos correctamente, Verdaderos Negativos(*True Negatives (TN)*) son los valores negativos correctamente predichos, Falsos Positivos (*False Positives (FP)*) es cuando la observación es negativa, pero se predice positiva. Finalmente, Falsos Negativos (*False Negative (FN)*) cuando la observación es positiva, pero se predice negativa. A partir de estos conceptos se calculan los siguientes[33]:

Figura 63: *Ejemplo de Matriz de Confusión.*

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Fuente: *[Renuka, 2016]*

- ***Accuracy:*** Es la proporción de las observaciones correctamente predichas, como se muestra a continuación.

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{FN} + \text{TN}) \tag{45}$$

- ***Recall*** También se conoce como sensibilidad o tasa positiva verdadera. Es la proporción de eventos positivos correctamente predichos.

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \tag{46}$$

- ***Precisión*** Es la proporción de todas las predicciones positivas que son correctas. La precisión es una medida de cuántas predicciones positivas fueron observaciones positivas reales.

$$\text{Precisión} = \text{TP}/(\text{TP} + \text{FP}) \tag{47}$$

- ***F1 Score*** Es el promedio ponderado de *Precisión* y *Recall*. Por lo tanto, esta puntuación toma tanto falsos positivos como falsos negativos en cuenta.

$$\text{F1_Score} = 2 \cdot (\textit{Precisión} \cdot \textit{Recall})/(\textit{Precisión} + \textit{Recall}) \tag{48}$$

A partir de estas métricas, sera más fácil entender los resultados de cada uno de los modelos evaluados.

[33]Para evaluar técnicas de aprendizaje de máquina es necesario considerar *Métricas*, como son: *Accuracy, Recall, Precisión y F1_Score*

4.2.2. Entrenamiento y Construcción del modelo

En esta fase se plantearon dos modelos, estos son:

- *Bag-of-Words*
- *Redes neuronales convolucionales*

4.2.2.1. *Bag-of-Words (BoW)*

Para evaluar los efectos de este modelo, se comparan los extractores de características SIFT y SURF. Además, se varia el tamaño del Codebook$_{size} \in [300, 500, 700, 900, 1100, 1300]$. Finalmente se utilizan las cuatro técnicas de aprendizaje de máquina de forma independiente para la evaluación del modelo.

1. ***Support Vector Machine (SVM):*** Los resultados de las carpetas Split 1, Split 2 y Split 3 generadas a partir del *cross-validation* se muestran en la Tabla 12. En esta se visualizan los efectos de aplicar un $kernel_{linear}$ con un parámetro C óptimo equivalente a 100, un extractor de características SIFT y conjunto de tamaños variados del *codebook* (Cod$_{size}$).

Tabla 12: *Resumen de los resultados a partir de la aplicación del SVM con un $kernel_{linear}$ con C equivalente a 100 y el extractor de características SIFT.*

SPLIT 1: $kernel_{\text{linear}}$ + SIFT				
Cod$_{size}$	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5754	0.5546	0.5832	0.5592
500	0.6280	0.6105	0.6248	0.6148
700	0.6596	0.6316	0.6784	0.6453
900	**0.6780**	**0.6493**	**0.6991**	**0.6648**
1100	0.6478	0.6139	0.6825	0.6288
1300	0.6570	0.6130	0.6986	0.6280
SPLIT 2: $kernel_{\text{linear}}$ + SIFT				
Cod$_{size}$	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5688	0.5216	0.6201	0.5460
500	0.6175	0.5976	0.6350	0.6088
700	0.6350	0.6102	0.6616	0.6261
900	**0.6442**	**0.6203**	0.6941	**0.6412**
1100	0.6416	0.6114	0.6939	0.6309
1300	0.6298	0.5878	**0.7133**	0.6168
SPLIT 3: $kernel_{\text{linear}}$ + SIFT				
Cod$_{size}$	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5758	0.5509	0.5950	0.5623
500	0.6131	0.5938	0.6124	0.5989
700	0.6543	0.6316	0.6645	0.6402
900	**0.6679**	**0.6408**	0.6879	**0.6534**
1100	0.6671	0.6304	0.7047	0.6474
1300	0.6574	0.6114	**0.7158**	0.6339

Fuente: *Elaboración propia*

En la Tabla 12 se visualizan los resultados por separado para Split 1, Split 2 y Split 3. Por otro lado, la media (U)[34] y desviación estándar (σ)[35] de estos resultados se muestran a detalle en la Tabla 13. Donde U_{Acc}, U_{Pre}, U_{Recall} y U_{F1} representan a la media de la *Accuracy*, *Precisión*, *Recall* y *F1_Score* respectivamente, mientas que σ_{Acc}, σ_{Pre}, σ_{Recall} y σ_{F1} representan la desvición estándar de la *Accuracy*, *Precisión*, *Recall* y *F1_Score* respectivamente.

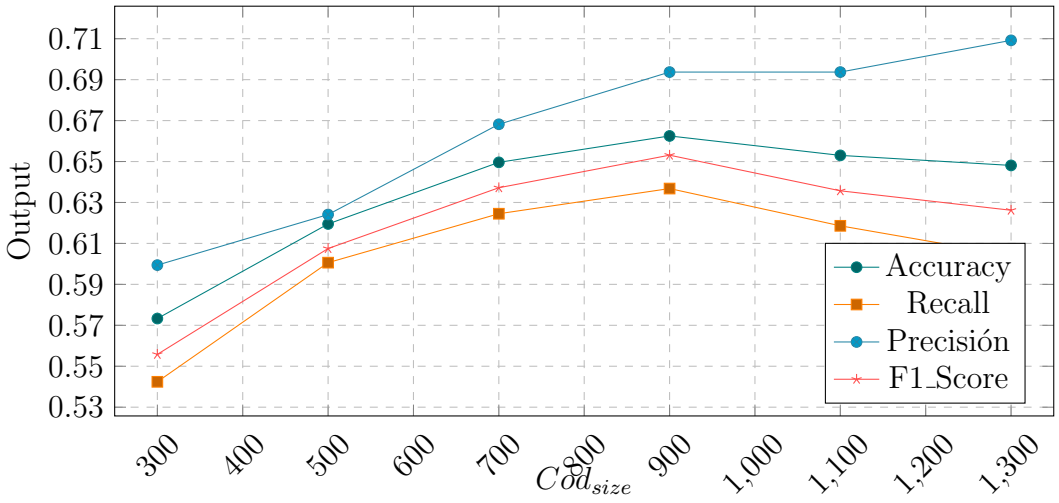
Tabla 13: *Resumen del promedio y desviación estándar a partir de la aplicación del SVM con* $kernel_{linear}$*, C equivalente a 100 y el extractor de características SIFT.*

SPLIT PROMEDIO: $kernel_{\text{linear}}$ + SIFT								
$\textbf{Cod}_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.5733	0.0039	0.5424	0.0181	0.5994	0.0188	0.5558	0.0087
500	0.6195	0.0077	0.6006	0.0088	0.6241	0.0113	0.6075	0.0080
700	0.6496	0.0129	0.6245	0.0124	0.6682	0.0090	0.6372	0.0099
900	**0.6625**	0.0188	**0.6368**	0.0149	0.6937	0.0056	**0.6531**	0.0118
1100	0.6530	0.0123	0.6186	0.0103	0.6937	0.0111	0.6357	0.0102
1300	0.6481	0.0158	0.6041	0.0141	**0.7092**	0.0093	0.6262	0.0087

Fuente: *Elaboración propia*

Los mejores resultados de utilizar los parámetros de la Tabla 13 son para un tamaño de *codebook* (Cod_{size}) equivalente a 900, 900, 1300 y 900 para valores promedio como **66.25 %, 63.68 %, 70.92 %** y **65.31 %** durante la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Estos resultados se visualizan en la Figura 64.

Figura 64: *Promedio de los resultados de aplicar SVM con* $kernel_{linear}$*, C equivalente a 100 y el extractor de características SIFT.*



Fuente: *Elaboración propia*

[34]Representa el valor promedio del conjunto de resultados
[35]Es una medida del grado de dispersión de los datos con respecto al valor promedio

De forma similar, los resultados de aplicar SVM con $kernel_{linear}$, con parámetro C óptimo equivalente a 100, un extractor de características SURF y un tamaño variado de *codebook* (Cod_{size}), se muestran en la Tabla 14. Todos estos resultados fueron generados de las carpetas Split 1, Split 2 y Split 3.

Tabla 14: *Resumen de los resultados a partir de la aplicación del SVM con $kernel_{lineal}$ con C equivalente a 100 y extractor de características SURF.*

SPLIT 1: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5767	0.5566	0.5904	0.5634
500	0.6048	0.5975	0.6170	0.6032
700	0.6342	0.6188	0.6477	0.6286
900	0.6447	**0.6214**	0.6790	**0.6390**
1100	**0.6473**	0.6164	**0.6897**	0.6360
1300	0.6302	0.5941	0.6952	0.6182
SPLIT 2: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5719	0.5485	0.5928	0.5575
500	0.6061	0.5974	0.6209	0.6060
700	**0.6407**	**0.6251**	0.6643	**0.6386**
900	0.6372	0.6114	0.6769	0.6283
1100	0.6210	0.5902	**0.6845**	0.6157
1300	0.5890	0.5478	0.6694	0.5699
SPLIT 3: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5780	0.5576	0.5974	0.5697
500	0.6184	0.6083	0.6327	0.6177
700	0.6315	0.6148	0.6584	0.6317
900	**0.6469**	**0.6205**	0.6764	**0.6400**
1100	0.6447	0.6116	**0.6868**	0.6343
1300	0.6210	0.5802	0.6766	0.6060

Fuente: *Elaboración propia*

En la Tabla 14 se observa que los mejores resultados varían entre las carpetas Split. Por ejemplo, el $\text{Cod}_{size} = 900$ representa el mejor valor para la *Accuracy* de la carpeta Split 3. Sin embargo, un tamaño de *codebook* Cod_{size} equivalente a 1100 y 700 genera un valor alto para las carpetas Split 1 y Split 2 respectivamente, el mismo problema ocurre con la *Precisión*, *Recall* y *F1_Score*. Por lo tanto, existe la necesidad de generar el promedio de las carpetas Split, para identificar el valor medio más representativo, como se muestra en la Tabla 15.

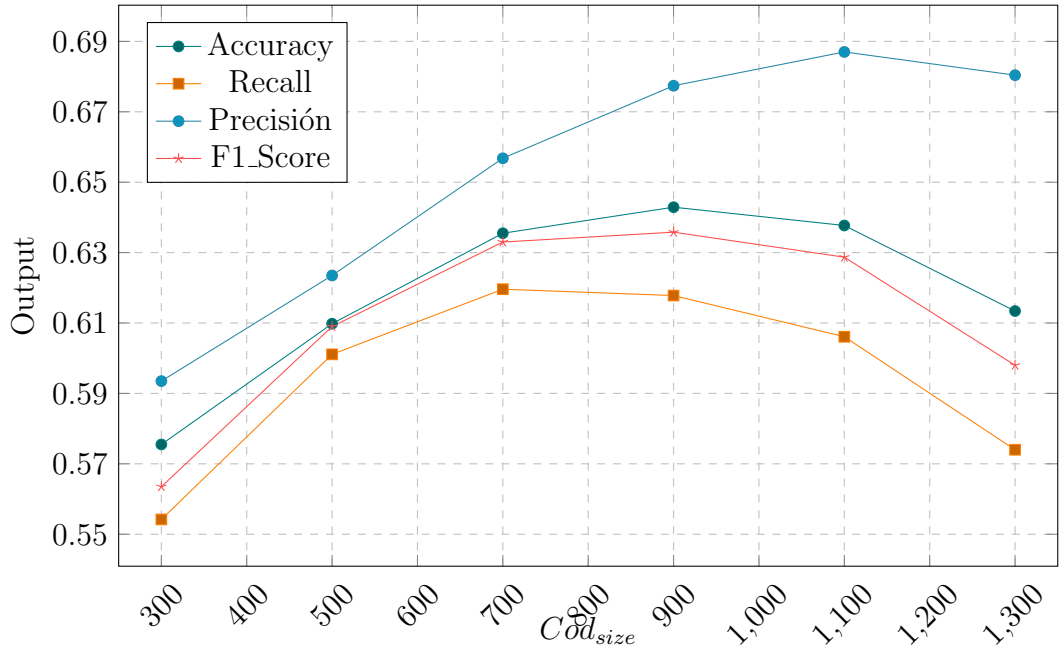
Tabla 15: *Resumen del promedio y desviación estándar a partir de la aplicación del SVM con* $kernel_{linear}$*, C equivalente a 100 y el extractor de características SURF.*

SPLIT PROMEDIO: $kernel_{\text{linear}}$ + SURF								
$\mathbf{Cod}_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.5755	0.0032	0.5542	0.0050	0.5935	0.0036	0.5635	0.0061
500	0.6098	0.0075	0.6011	0.0063	0.6235	0.0082	0.6090	0.0077
700	0.6355	0.0047	**0.6196**	0.0052	0.6568	0.0084	0.6330	0.0051
900	**0.6429**	0.0051	0.6178	0.0055	0.6774	0.0014	**0.6358**	0.0065
1100	0.6377	0.0145	0.6061	0.0139	**0.6870**	0.0026	0.6287	0.0113
1300	0.6134	0.0216	0.5740	0.0238	0.6804	0.0133	0.5980	0.0251

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 15 son para un tamaño de *codebook* (Cod_{size}) equivalente a 900, 700, 1100 y 900. Se obtuvo un **64.29 %**, **61.96 %**, **68.70 %** y **63.58 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Se puede ver en la Figura 65 los detalles de estos resultados.

Figura 65: *Promedio de los resultados de aplicar SVM con* $kernel_{linear}$*, C equivalente a 100 y el extractor de características SURF.*



Fuente: *Elaboración propia*

Por otro lado, además de aplicar un SVM con $kernel_{\mathbf{linear}}$, se aplico un SVM con $\boldsymbol{kernel}_{\mathbf{rbf}}$, con los parámetros C y γ más óptimos equivalentes a 100 y 10 respectivamente y un extractor de características SIFT sobre un conjunto de tamaños variados del *codebook* (Cod_{size}), como se muestra en la Tabla 16.

Tabla 16: *Resumen de los resultados a partir de la aplicación del SVM con $kernel_{rbf}$ con C y γ equivalentes a 100 y 10 respectivamente y extractor de características SIFT.*

SPLIT 1: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5789	0.5794	0.5936	0.5832
500	0.6508	0.6402	0.6709	0.6509
700	0.6421	0.6287	0.6621	0.6391
900	0.6535	**0.6444**	**0.6777**	**0.6548**
1100	0.6434	0.6259	0.6732	0.6400
1300	**0.6578**	0.6407	0.6746	0.6524
SPLIT 2: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.6008	0.5945	0.6188	0.5984
500	0.6425	0.6359	0.6868	0.6513
700	0.6394	0.6269	0.6681	0.6400
900	0.6298	0.6175	0.6610	0.6302
1100	0.6460	0.6358	0.6784	0.6501
1300	**0.6565**	**0.6438**	**0.6932**	**0.6606**
SPLIT 3: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5890	0.5749	0.6130	0.5865
500	0.6342	0.6184	0.6552	0.6299
700	0.6578	**0.6462**	0.6824	**0.6561**
900	0.6429	0.6270	0.6706	0.6416
1100	**0.6618**	0.6408	**0.6826**	0.6539
1300	0.6587	0.6432	0.6824	0.6549

Fuente: *Elaboración propia*

En la Tabla 16, se observa que los mejores resultados para la carpeta Split 1 son valores equivalentes a **65.78 %, 64.44 %, 67.77 %** y **65.48 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para la carpeta Split 2 son **65.65 %, 64.38 %, 69.32 %** y **66.06 %** utilizando un tamaño de *codebook* equivalente a 1300 para todos casos. Por último, en Split 3 se obtiene valores como **66.18 %, 64.62 %, 68.26 %** y **65.61 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 17.

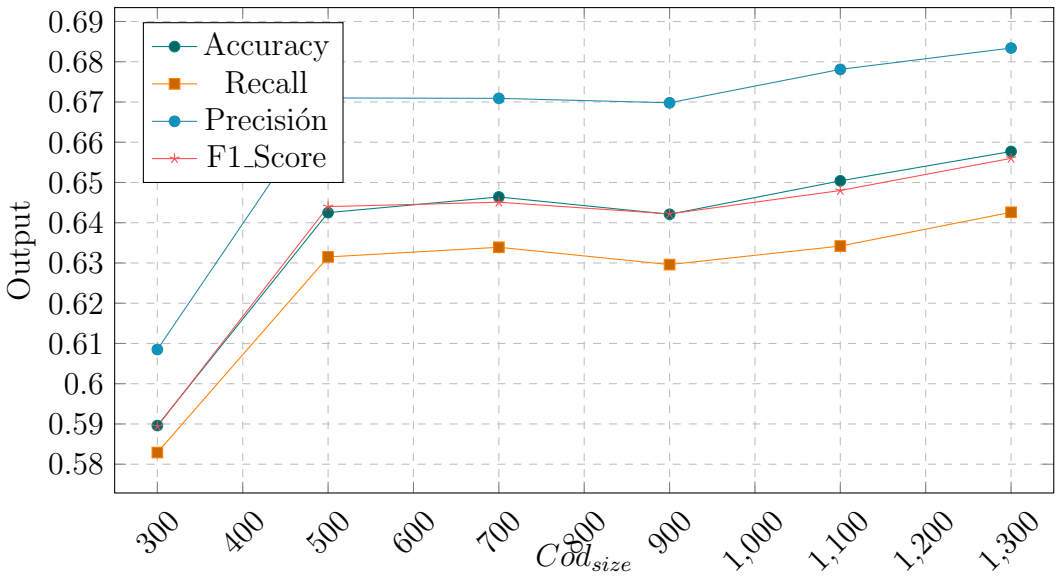
Tabla 17: *Resumen del promedio y desviación estándar a partir de la aplicación del SVM con kernel$_{rbf}$, C y γ equivalentes a 100 y 10 respectivamente y extractor de características SIFT.*

SPLIT PROMEDIO: $kernel_{\mathbf{rbf}}$ + SIFT								
Cod$_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.5896	0.0110	0.5829	0.0103	0.6085	0.0132	0.5894	0.0080
500	0.6425	0.0083	0.6315	0.0115	0.6710	0.0158	0.6440	0.0122
700	0.6464	0.0099	0.6339	0.0107	0.6709	0.0104	0.6451	0.0096
900	0.6421	0.0119	0.6296	0.0136	0.6698	0.0084	0.6422	0.0123
1100	0.6504	0.0100	0.6342	0.0076	0.6781	0.0047	0.6480	0.0072
1300	**0.6577**	0.0011	**0.6426**	0.0016	**0.6834**	0.0093	**0.6560**	0.0042

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 17 son para un tamaño de *codebook* (Cod$_{size}$) equivalente a 1300, se obtuvo un **65.77 %**, **64.26 %**, **68.34 %** y **65.60 %** como resultados de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Se puede ver en la Figura 66 los detalles de los resultados obtenidos.

Figura 66: *Promedio de los resultados de aplicar SVM con kernel$_{rbf}$, C y γ equivalentes a 100 y 10 respectivamente, y el extractor de características SIFT.*



Fuente: *Elaboración propia*

Así mismo, se aplico un SVM con $\boldsymbol{kernel_{rbf}}$, con los parámetros C y γ más óptimos equivalentes a 100 y 10 respectivamente, sobre un extractor de características SURF, utilizando un conjunto de tamaños variados del *codebook* (Cod_{size}), como se muestra en la Tabla 18.

Tabla 18: *Resumen de los resultados a partir de la aplicación del SVM con $kernel_{rbf}$ con C y γ equivalentes a 100 y 10 respectivamente y extractor de características SURF.*

SPLIT 1: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.6035	0.5968	0.6265	0.6064
500	0.6430	0.6368	0.6639	0.6464
700	0.6395	0.6364	0.6546	0.6425
900	0.6360	0.6245	0.6600	0.6375
1100	**0.6649**	**0.6582**	**0.6867**	**0.6688**
1300	0.6570	0.6516	0.6854	0.6636
SPLIT 2: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.5890	0.5785	0.6113	0.5885
500	0.6342	0.6239	0.6629	0.6381
700	0.6329	0.6297	0.6580	0.6397
900	0.6325	0.6189	0.6629	0.6340
1100	**0.6452**	**0.6357**	**0.6955**	**0.6555**
1300	0.6342	0.6245	0.6604	0.6376
SPLIT 3: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.6009	0.5972	0.6267	0.6063
500	0.6359	0.6270	0.6679	0.6413
700	0.6359	0.6285	0.6658	0.6422
900	0.6399	0.6245	0.6631	0.6391
1100	0.6521	**0.6361**	0.6908	**0.6550**
1300	**0.6522**	0.6328	**0.6943**	0.6539

Fuente: *Elaboración propia*

En la Tabla 18 se observa que los mejores resultados para Split 1 son los valores equivalentes a **66.49 %, 65.82 %, 68.67 %** y **66.88 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **64.52 %, 63.57 %, 69.55 %** y **65.55 %** utilizando un tamaño de *codebook* equivalente a 1100 para todos casos. Por último, en Split 3 se obtiene valores como **65.22 %, 63.61 %, 69.43 %** y **65.50 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 19.

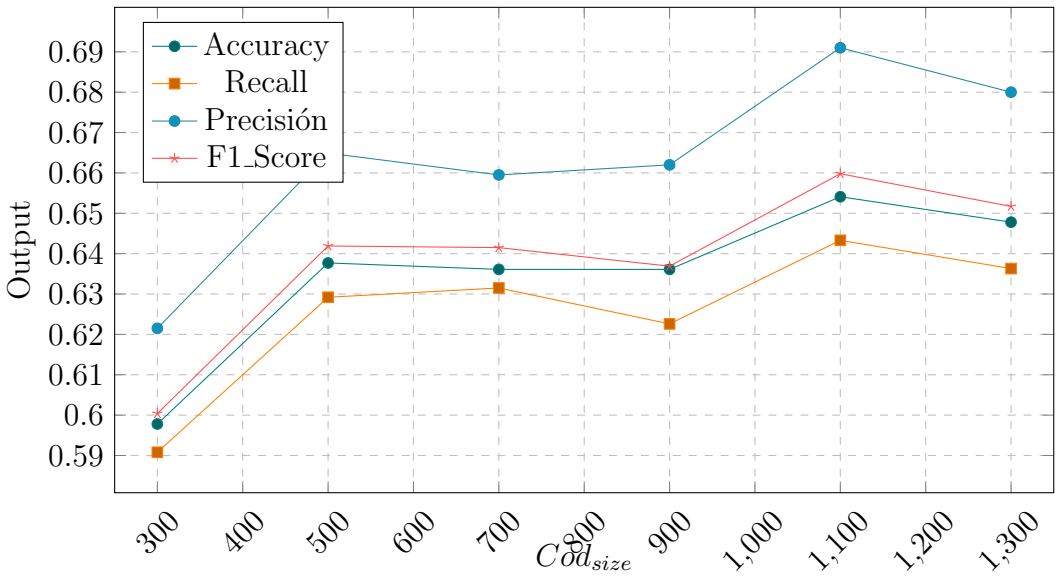
Tabla 19: *Resumen del promedio y desviación estándar a partir de la aplicación del SVM con* $kernel_{rbf}$*,* C *y* γ *equivalentes a 100 y 10 respectivamente y extractor de características SURF.*

SPLIT PROMEDIO: $\boldsymbol{kernel_{rbf}}$ **+ SURF**								
$\mathbf{Cod}_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.5978	0.0077	0.5908	0.0107	0.6215	0.0088	0.6004	0.0103
500	0.6377	0.0047	0.6292	0.0067	0.6649	0.0026	0.6419	0.0042
700	0.6361	0.0033	0.6315	0.0043	0.6595	0.0057	0.6415	0.0015
900	0.6361	0.0037	0.6226	0.0032	0.6620	0.0017	0.6369	0.0026
1100	**0.6541**	0.0100	**0.6433**	0.0129	**0.6910**	0.0044	**0.6598**	0.0078
1300	0.6478	0.0120	0.6363	0.0139	0.6800	0.0176	0.6517	0.0131

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 19 son para un tamaño de *codebook* (Cod_{size}) equivalente a 1100. Se obtuvo un **65.41 %**, **64.33 %**, **69.10 %** y **65.98 %** como los valores más alto de la *Accuracy*, *Precisión*, *Recall* y *F1_Score* respectivamente. Se puede ver en la Figura 67 los detalles de estos resultados.

Figura 67: *Promedio de los resultados de aplicar SVM con* $kernel_{rbf}$*,* C *y* γ *equivalentes a 100 y 10 respectivamente, y extractor de características SURF.*



Fuente: *Elaboración propia*

2. ***Random Forest (RF):*** En la Tabla 20 se visualizan los efectos de aplicar la técnica de aprendizaje *Random Forest*, con un número estimado de árboles (n_estimators) equivalentes a 500 y una profundidad (max_depth) igual a 50, sobre un extractor de características SIFT y un conjunto de tamaños variados del *codebook* (Cod_{size}).

Tabla 20: *Resumen de los resultados a partir de la aplicación del Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SIFT.*

SPLIT 1: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4460	0.3825	0.4495	0.3939
500	0.4438	0.3750	0.4460	0.3850
700	0.4464	0.3704	0.4543	0.3776
900	0.4464	0.3769	0.4526	0.3835
1100	0.4425	0.3704	0.4447	0.3735
1300	**0.4741**	**0.3975**	**0.4714**	**0.3999**
SPLIT 2: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4390	0.3748	0.4368	0.3948
500	0.4289	0.3662	0.4157	0.3907
700	0.4157	0.3497	0.4206	0.3694
900	0.4381	0.3663	0.4368	0.3862
1100	0.4399	0.3698	0.4438	0.3899
1300	**0.4464**	**0.3760**	**0.4508**	**0.3963**
SPLIT 3: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4456	0.3819	0.4521	0.3996
500	0.4596	0.3867	0.4445	0.4054
700	0.4517	0.3746	0.4447	0.3916
900	**0.4692**	0.3886	**0.4706**	0.4070
1100	0.4653	0.3849	0.4592	0.4038
1300	0.4666	**0.3893**	0.4644	**0.4087**

Fuente: *Elaboración propia*

En la Tabla 20 se observa que los mejores resultados para Split 1 son los valores equivalentes a **47.41 %, 39.75 %, 47.14 %** y **39.99 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **44.64 %, 37.60 %, 45.08 %** y **39.63 %** utilizando un tamaño de *codebook* equivalente a 1300 para todos casos. Por último, en Split 3 se obtiene valores como **46.92 %, 38.93 %, 47.06 %** y **40.87 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 21.

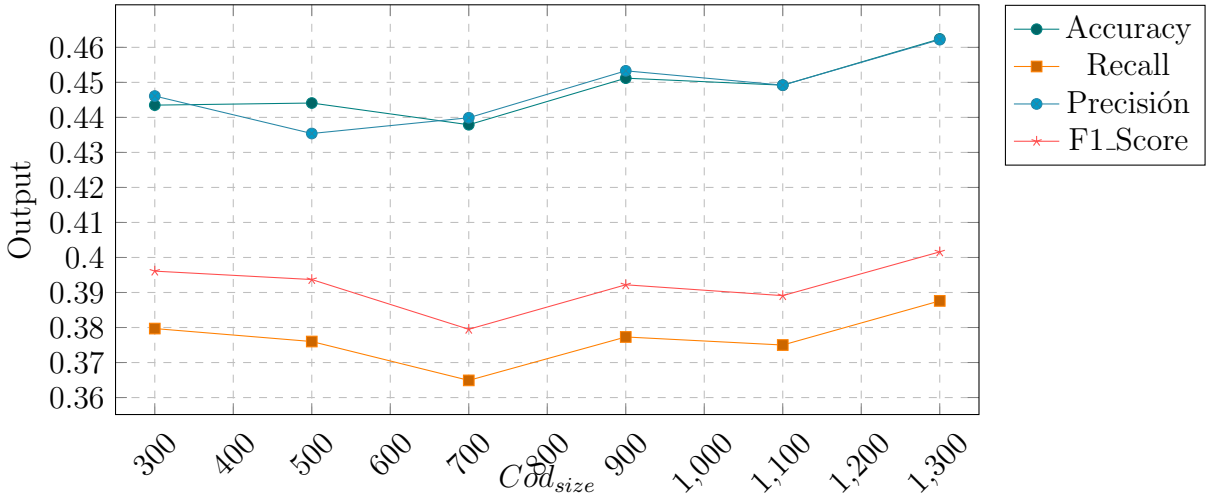
Tabla 21: *Resumen del promedio y desviación estándar a partir de la aplicación de Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SIFT.*

SPLIT PROMEDIO:RF + SIFT								
Cod$_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.4435	0.0039	0.3797	0.0043	0.4461	0.0082	0.3961	0.0031
500	0.4441	0.0154	0.3760	0.0103	0.4354	0.0171	0.3937	0.0105
700	0.4379	0.0194	0.3649	0.0133	0.4399	0.0174	0.3795	0.0112
900	0.4512	0.0161	0.3773	0.0112	0.4533	0.0169	0.3922	0.0129
1100	0.4492	0.0140	0.3750	0.0086	0.4492	0.0086	0.3891	0.0152
1300	**0.4624**	0.0143	**0.3876**	0.0109	**0.4622**	0.0105	**0.4016**	0.0064

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 21 son para un tamaño de *codebook* (Cod$_{size}$) equivalente a 900. Se obtuvo un **46.24 %**, **38.76 %**, **46.22 %** y **40.16 %** como el valor más alto de la *Accuracy*, *Precisión*, *Recall* y *F1_Score* respectivamente. Se puede ver en la Figura 68 los detalles de estos valores.

Figura 68: *Promedio de los resultados de aplicar Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente, y un extractor de características SIFT.*



Fuente: *Elaboración propia*

Así mismo, se aplico la técnica *Random Forest* con un número estimado de árboles (n_estimators) equivalentes a 500 y con una profundidad (max_depth) igual a 50, sobre un extractor de características SURF y un conjunto de tamaños variados del *codebook* (Cod_{size}). Como se muestra en la Tabla 22.

Tabla 22: *Resumen de los resultados a partir de la aplicación del Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SURF.*

SPLIT 1: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4631**	**0.4049**	**0.4610**	**0.4183**
500	0.4592	0.3968	0.4583	0.4077
700	0.4439	0.3749	0.4543	0.3800
900	0.4522	0.3782	0.4469	0.3836
1100	0.4592	0.3835	0.4579	0.3873
1300	0.4548	0.3845	0.4596	0.3908
SPLIT 2: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4469**	**0.3917**	**0.4578**	**0.4158**
500	0.4434	0.3811	0.4311	0.4021
700	0.4281	0.3626	0.4245	0.3795
900	0.4386	0.3699	0.4337	0.3840
1100	0.4329	0.3661	0.4350	0.3839
1300	0.4329	0.3639	0.4307	0.3790
SPLIT 3: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4702**	**0.4092**	**0.4679**	**0.4295**
500	0.4570	0.3886	0.4631	0.4105
700	0.4447	0.3714	0.4478	0.3864
900	0.4469	0.3725	0.4425	0.3813
1100	0.4522	0.3779	0.4548	0.3902
1300	0.4456	0.3709	0.4535	0.3847

Fuente: *Elaboración propia*

En la Tabla 22 se observa que los mejores resultados para Split 1 son los valores equivalentes a **46.31 %, 40.49 %, 46.10 %** y **41.83 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **44.69 %, 39.17 %, 45.78 %** y **41.58 %** utilizando un tamaño de *codebook* equivalente a 300 para todos casos. Por último, en Split 3 se obtiene valores como **47.02 %, 40.92 %, 46.79 %** y **42.95 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 23.

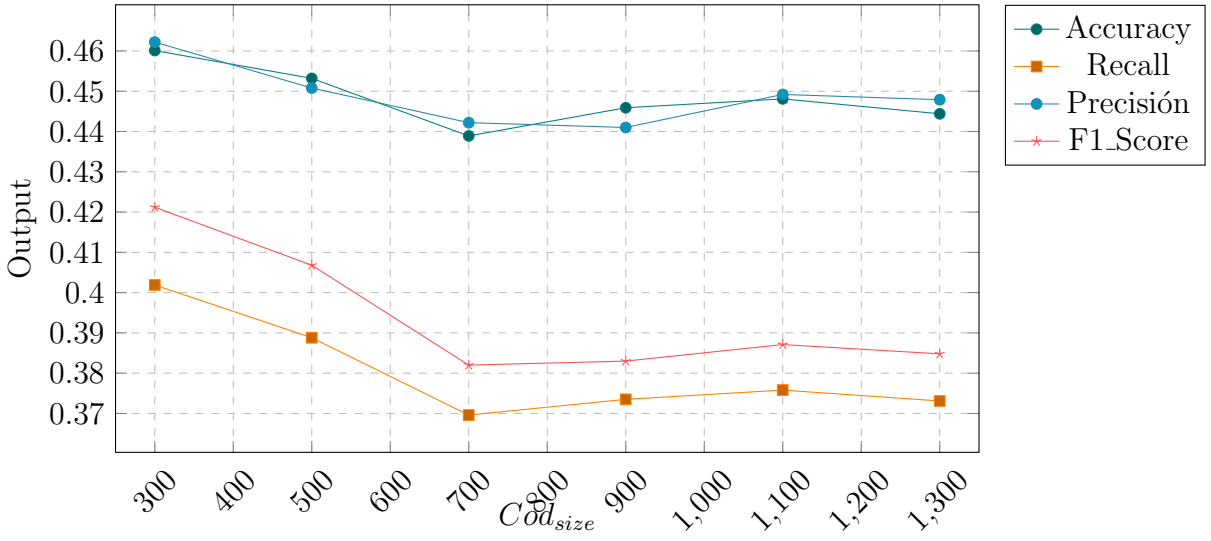
Tabla 23: *Resumen del promedio y desviación estándar a partir de la aplicación de Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente y un extractor de características SURF.*

SPLIT PROMEDIO: RF + SURF								
Cod$_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	**0.4601**	0.0119	**0.4019**	0.0091	**0.4622**	0.0052	**0.4212**	0.0073
500	0.4532	0.0085	0.3888	0.0079	0.4508	0.0173	0.4068	0.0043
700	0.4389	0.0094	0.3696	0.0063	0.4422	0.0157	0.3820	0.0038
900	0.4459	0.0069	0.3735	0.0042	0.4410	0.0067	0.3830	0.0014
1100	0.4481	0.0136	0.3758	0.0089	0.4492	0.0124	0.3871	0.0032
1300	0.4444	0.0110	0.3731	0.0105	0.4479	0.0152	0.3848	0.0059

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 23 son para un tamaño de *codebook* (Cod$_{size}$) equivalente a 300. Se obtuvo un **46.01 %**, **40.19 %**, **46.22 %** y **42.12 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Se puede ver en la Figura 69 los detalles de estos resultados.

Figura 69: *Promedio de los resultados de aplicar Random Forest con parámetros n_estimators y max_depth equivalentes a 500 y 50 respectivamente, y un extractor de características SURF.*



Fuente: *Elaboración propia*

3. ***k-Nearest Neighbor (kNN):*** En la Tabla 24 se visualizan los efectos de aplicar la técnica de aprendizaje *K-Nearest Neighbor*, donde el parámetro k = 1, sobre un extractor de características SIFT y un conjunto de tamaños variados del *codebook* (Cod_{size}).

Tabla 24: *Resumen de los resultados a partir de la aplicación de k-Nearest Neighbor con k = 1 y un extractor de características SIFT.*

SPLIT 1: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4474	0.4654	0.4496	0.4520
500	**0.4618**	**0.4807**	**0.4644**	**0.4616**
700	0.4338	0.4542	0.4356	0.4290
900	0.4263	0.4596	0.4514	0.4295
1100	0.3579	0.3918	0.3991	0.3639
1300	0.3294	0.3583	0.3824	0.3340
SPLIT 2: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4575	0.4687	0.4603	0.4609
500	**0.4583**	**0.4751**	**0.4606**	**0.4610**
700	0.4504	0.4672	0.4579	0.4513
900	0.4154	0.4384	0.4412	0.4188
1100	0.3851	0.4105	0.4354	0.3918
1300	0.3377	0.3633	0.4054	0.3466
SPLIT 3: SIFT				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	0.4482	0.4720	0.4556	0.4573
500	**0.4627**	**0.4824**	**0.4688**	**0.4659**
700	0.4377	0.4598	0.4406	0.4375
900	0.4132	0.4335	0.4306	0.4151
1100	0.3890	0.4080	0.4340	0.3933
1300	0.3526	0.3702	0.4019	0.3567

Fuente: *Elaboración propia*

En la Tabla 24 se observa que los mejores resultados para Split 1 son los valores equivalentes a **46.18 %, 48.07 %, 46.44 %** y **46.16 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **45.83 %, 47.51 %, 46.06 %** y **46.10 %** utilizando un tamaño de *codebook* equivalente a 500 para todos casos. Por último, en Split 3 se obtiene valores como **46.27 %, 48.24 %, 46.88 %** y **46.59 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 25.

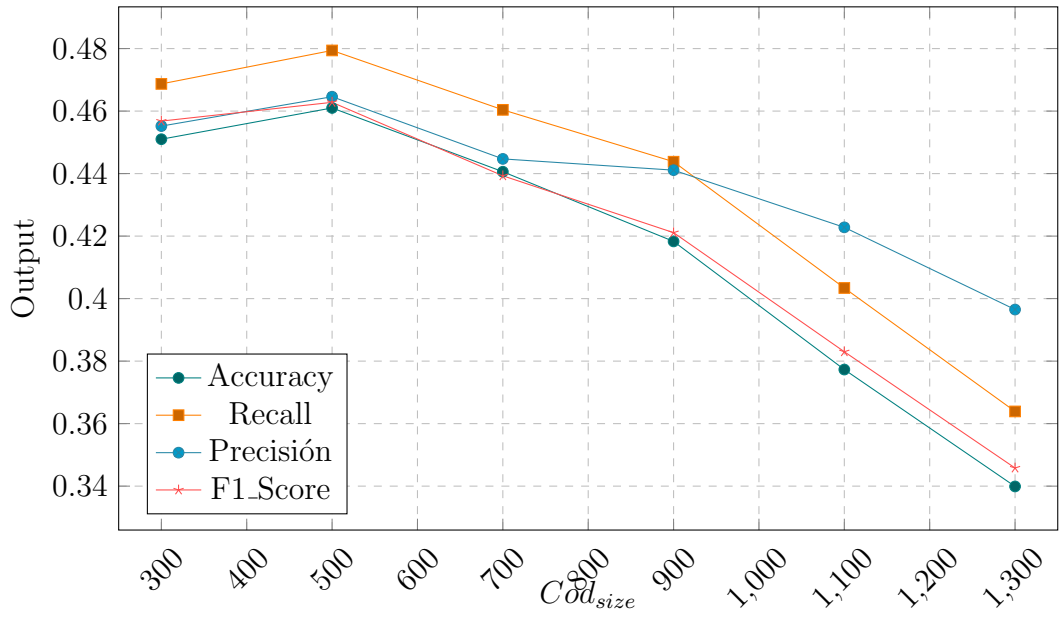
Tabla 25: *Resumen del promedio y desviación estándar a partir de la aplicación de k-Nearest Neighbor con k = 1 y un extractor de características SIFT.*

SPLIT PROMEDIO:kNN + SIFT								
Cod$_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	0.4510	0.0056	0.4687	0.0033	0.4552	0.0054	0.4568	0.0045
500	**0.4610**	0.0023	**0.4794**	0.0038	**0.4646**	0.0041	**0.4628**	0.0026
700	0.4406	0.0087	0.4604	0.0065	0.4447	0.0117	0.4393	0.0113
900	0.4183	0.0070	0.4438	0.0139	0.4411	0.0104	0.4211	0.0075
1100	0.3773	0.0170	0.4034	0.0101	0.4228	0.0206	0.3830	0.0166
1300	0.3399	0.0118	0.3639	0.0060	0.3965	0.0124	0.3458	0.0114

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 25 son para un tamaño de *codebook* (Cod$_{size}$) equivalente a 500. Se obtuvo un **46.10 %**, **47.94 %**, **46.46 %** y **46.28 %** como el valor más alto de la *Accuracy*, *Precisión*, *Recall* y *F1_Score* respectivamente. Se puede ver en la Figura 70 los detalles de estos valores.

Figura 70: *Promedio de los resultados de aplicar k-Nearest Neighbor con k = 1 y un extractor de características SIFT.*



Fuente: *Elaboración propia*

Así mismo, se aplico la técnica k-Nearest Neighbor (kNN) con $k = 1$, sobre un extractor de características SURF y un conjunto de tamaños variados del *codebook* (Cod_{size}). Como se muestra en la Tabla 26.

Tabla 26: *Resumen de los resultados a partir de la aplicación de k-Nearest Neighbor con k = 1 y un extractor de características SURF.*

SPLIT 1: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4697**	**0.4854**	**0.4802**	**0.4752**
500	0.4443	0.4673	0.4586	0.4490
700	0.4118	0.4370	0.4741	0.4255
900	0.3649	0.4020	0.4489	0.3859
1100	0.2965	0.3403	0.4403	0.3335
1300	0.2456	0.2900	0.4547	0.2840
SPLIT 2: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4355**	0.4465	0.4469	**0.4416**
500	0.4351	**0.4467**	**0.4527**	0.4344
700	0.4075	0.4293	0.4438	0.4161
900	0.3680	0.3878	0.4569	0.3813
1100	0.3039	0.3338	0.4395	0.3351
1300	0.2399	0.2782	0.4824	0.2897
SPLIT 3: SURF				
Cod_{size}	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
300	**0.4579**	**0.4725**	0.4690	**0.4637**
500	0.4557	0.4709	0.4645	0.4585
700	0.4281	0.4555	**0.4759**	0.4436
900	0.3487	0.3788	0.4305	0.3661
1100	0.3294	0.3581	0.4414	0.3483
1300	0.2763	0.3125	0.4337	0.2956

Fuente: *Elaboración propia*

En la Tabla 26 se observa que los mejores resultados para Split 1 son los valores equivalentes a **46.97 %, 48.54 %, 48.02 %** y **47.52 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **43.55 %, 44.67 %, 45.27 %** y **44.16 %** utilizando un tamaño de *codebook* equivalente a 300, 500, 500 y 300 respectivamente. Por último, en Split 3 se obtiene valores como **45.79 %, 47.25 %, 46.90 %** y **46.37 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 27.

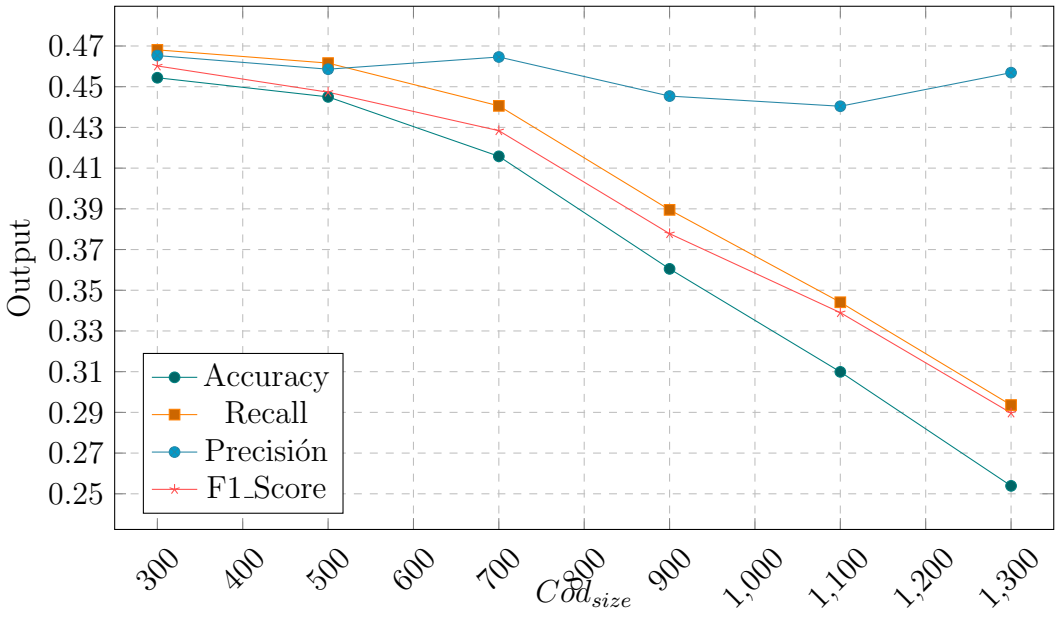
Tabla 27: *Resumen del promedio y desviación estándar a partir de la aplicación de k-Nearest Neighbor con k = 1 y un extractor de características SURF.*

SPLIT PROMEDIO: kNN + SURF								
Cod$_{size}$	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
300	**0.4544**	0.0174	**0.4681**	0.0198	**0.4653**	0.0169	**0.4602**	0.0171
500	0.4450	0.0103	0.4616	0.0131	0.4586	0.0059	0.4473	0.0122
700	0.4158	0.0109	0.4406	0.0135	0.4646	0.0180	0.4284	0.0140
900	0.3605	0.0104	0.3895	0.0117	0.4454	0.0135	0.3778	0.0104
1100	0.3099	0.0172	0.3441	0.0126	0.4404	0.0009	0.3390	0.0081
1300	0.2539	0.0196	0.2936	0.0174	0.4569	0.0244	0.2897	0.0058

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 27 son para un tamaño de *codebook* (Cod$_{size}$) equivalente a 300. Se obtuvo un **45.44 %**, **46.81 %**, **46.53 %** y **46.02 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Se puede ver en la Figura 71 que mientras más grande es el tamaño del *codebook* (Cod$_{size}$) la *Accuracy*, *Recall* y *F1_Score* disminuyen considerablemente. Por otro lado, la Precisión se mantiene relativamente constante.

Figura 71: *Promedio de los resultados de aplicar k-Nearest Neighbor (kNN) con k = 1 y un extractor de características SURF.*



Fuente: *Elaboración propia*

4. ***Neuronal Network (NN):*** A continuación, se visualizan los resultados de aplicar la técnica de aprendizaje de máquina basado en *Neuronal Network*, que consta de 3 capas. Donde la primera capa, segunda capa y tercera capa utilizan 1024 neuronas de entrada, 1024 neuronas ocultas y 14 neuronas de salida respectivamente. Así mismo, sobre las neuronas de salida se aplica la función de activación *Softmax*[36]. Para evaluar el rendimiento de este clasificador, se utiliza el entrenamiento basado en *Adam Optimization* y la función de coste *Entropía Cruzada* para actualizar los pesos. El sistema fue capacitado durante 15000 iteraciones para obtener pesos para el modelo de clasificación. Las neuronas de entrada utilizan el vector de características generado por Bag-of-Words, a partir de un tamaño variado de *codebook* (Cod_{size}), utilizando SIFT y SURF como extractores de características, como se muestra en las Tablas 28, 29, 32, 33, 36, 37, 40, 41, 44, 45, 48 y 49.

   En la Tabla 28 se visualizan los resultados de aplicar *Neural Network* durante 15000 iteraciones, utilizando un extractor de características SIFT, sobre un codebook equivalente a 300. Además, se observa que los mejores resultados para Split 1 son los valores equivalentes a **53.51 %, 52.19 %, 60.54 %** y **51.45 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.54 %, 49.71 %, 61.10 %** y **48.84 %** se presentan durante la iteración 15000, 10000, 15000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **55.09 %, 52.96 %, 60.76 %** y **53.31 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 30.

   En la Tabla 29 se visualizan los resultados de aplicar *Neural Network* durante 15000 iteraciones, utilizando un extractor de características SURF, sobre un codebook equivalente a 300. Además, se observa que los mejores resultados para Split 1 son los valores equivalentes a **55.96 %, 55.17 %, 60.88 %** y **53.98 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **56.01 %, 52.29 %, 63.01 %** y **52.75 %** se presentan durante la iteración 15000, 15000, 11000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **55.83 %, 53.99 %, 61.58 %** y **55.28 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 31.

[36] *Softmax genera la probabilidad de pertenencia de la imagen con respecto a todas las clases*

Tabla 28: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 300.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0355	0.0714	0.0025	0.0049
1000	0.1969	0.1283	0.0972	0.0784
2000	0.2487	0.1839	0.3160	0.1391
3000	0.3158	0.2322	0.4592	0.2244
4000	0.3702	0.3296	0.4263	0.2883
5000	0.3864	0.3048	0.4997	0.2977
6000	0.4325	0.3834	0.4953	0.3866
7000	0.4873	0.4627	0.4958	0.4637
8000	0.4329	0.4246	0.4833	0.4062
9000	0.4281	0.4395	0.5066	0.4073
10000	0.4285	0.3409	**0.6054**	0.3557
11000	0.4974	0.4717	0.5400	0.4630
12000	0.5211	0.4864	0.5531	0.4969
13000	**0.5351**	**0.5219**	0.5437	**0.5145**
14000	0.5346	0.4982	0.5728	0.5002
15000	0.5114	0.5056	0.5660	0.5048
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0601	0.0714	0.0043	0.0081
1000	0.1873	0.1042	0.0278	0.0433
2000	0.2855	0.2196	0.2748	0.1879
3000	0.3127	0.2750	0.2658	0.2247
4000	0.3443	0.2888	0.4890	0.2787
5000	0.4075	0.3727	0.4745	0.3609
6000	0.3886	0.3365	0.4408	0.3092
7000	0.4465	0.4217	0.4830	0.3922
8000	0.4456	0.4133	0.4776	0.4042
9000	0.4500	0.4040	0.4819	0.3876
10000	0.4943	**0.4971**	0.5432	0.4718
11000	0.4991	0.4719	0.5245	0.4645
12000	0.5022	0.4962	0.5775	0.4742
13000	0.5039	0.4693	0.5319	0.4552
14000	0.5066	0.4794	0.5748	0.4765
15000	**0.5254**	0.4950	**0.6110**	**0.4884**
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0496	0.0714	0.0035	0.0067
1000	0.1320	0.0738	0.0157	0.0211
2000	0.2281	0.1577	0.2558	0.1043
3000	0.2732	0.2361	0.3103	0.1968
4000	0.3715	0.3318	0.4033	0.2989
5000	0.4154	0.3545	0.5004	0.3438
6000	0.4404	0.3644	0.5269	0.3547
7000	0.4759	0.4386	0.5425	0.4134
8000	0.4890	0.4585	0.5512	0.4649
9000	0.4921	0.4328	0.5892	0.4496
10000	0.4785	0.4435	0.5945	0.4596
11000	0.4921	0.4697	0.5956	0.4633
12000	0.4636	0.4701	0.5848	0.4777
13000	**0.5509**	0.5192	0.5769	**0.5331**
14000	0.5175	0.4753	**0.6076**	0.4877
15000	0.5404	**0.5296**	0.6010	0.5289

Fuente: *Elaboración propia*

Tabla 29: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 300.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0811	0.0714	0.0058	0.0107
1000	0.1667	0.1202	0.2047	0.0748
2000	0.2943	0.2627	0.3437	0.2228
3000	0.3118	0.2936	0.4969	0.2699
4000	0.4171	0.3958	0.4678	0.3654
5000	0.4412	0.3763	0.4533	0.3755
6000	0.4627	0.4032	0.4869	0.3805
7000	0.4346	0.4195	0.5072	0.4153
8000	0.4816	0.4720	0.5525	0.4740
9000	0.4908	0.4564	0.5866	0.4611
10000	0.5057	0.4623	0.5747	0.4847
11000	0.5048	0.4700	0.5635	0.4570
12000	0.5018	0.4763	0.5907	0.4771
13000	0.5298	0.4891	0.5960	0.5060
14000	0.5250	0.5260	0.5856	0.5205
15000	**0.5596**	**0.5517**	**0.6088**	**0.5398**
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0601	0.0714	0.0043	0.0081
1000	0.2149	0.1194	0.0554	0.0667
2000	0.2496	0.2377	0.2282	0.1729
3000	0.2899	0.2256	0.3728	0.2092
4000	0.3838	0.3159	0.4842	0.3107
5000	0.4083	0.3639	0.5034	0.3544
6000	0.4675	0.4301	0.5597	0.4193
7000	0.4895	0.4828	0.5448	0.4758
8000	0.4684	0.4874	0.5093	0.4513
9000	0.5268	0.5018	0.5609	0.4964
10000	0.5325	0.5059	0.5527	0.5065
11000	0.4904	0.4206	**0.6301**	0.4462
12000	0.5246	0.5092	0.5703	0.4892
13000	0.4732	0.4272	0.6002	0.4502
14000	0.5268	0.4833	0.6128	0.4862
15000	**0.5601**	**0.5229**	0.6135	**0.5275**
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0496	0.0714	0.0035	0.0067
1000	0.1697	0.1151	0.1705	0.0565
2000	0.2632	0.1834	0.3171	0.1265
3000	0.3741	0.3210	0.3693	0.3010
4000	0.3671	0.3504	0.4653	0.3289
5000	0.4237	0.3825	0.4705	0.3673
6000	0.4702	0.4562	0.5203	0.4319
7000	0.4965	0.4795	0.4627	0.4546
8000	0.4781	0.4862	0.5000	0.4638
9000	0.4925	0.4701	0.5092	0.4517
10000	0.5026	0.5123	0.5743	0.5055
11000	0.5154	0.4860	**0.6158**	0.4845
12000	0.5193	0.5210	0.5442	0.5037
13000	0.5276	0.5248	0.5922	0.5201
14000	0.5158	0.4893	0.6055	0.5035
15000	**0.5583**	**0.5399**	0.5995	**0.5528**

Fuente: *Elaboración propia*

Tabla 30: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 300.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0484	0.0123	0.0714	0.0000	0.0035	0.0009	0.0066	0.0016
1000	0.1721	0.0350	0.1021	0.0273	0.0469	0.0440	0.0476	0.0289
2000	0.2541	0.0291	0.1871	0.0311	0.2822	0.0308	0.1438	0.0420
3000	0.3006	0.0237	0.2478	0.0236	0.3451	0.1013	0.2153	0.0161
4000	0.3620	0.0153	0.3167	0.0242	0.4395	0.0443	0.2887	0.0101
5000	0.4031	0.0150	0.3440	0.0351	0.4915	0.0148	0.3342	0.0327
6000	0.4205	0.0279	0.3615	0.0236	0.4877	0.0435	0.3502	0.0389
7000	0.4699	0.0210	0.4410	0.0206	0.5071	0.0314	0.4231	0.0367
8000	0.4558	0.0294	0.4321	0.0235	0.5040	0.0409	0.4251	0.0345
9000	0.4567	0.0325	0.4255	0.0189	0.5259	0.0562	0.4148	0.0317
10000	0.4671	0.0343	0.4272	0.0793	0.5810	0.0332	0.4290	0.0638
11000	0.4962	0.0037	0.4711	0.0012	0.5534	0.0373	0.4636	0.0008
12000	0.4956	0.0293	0.4842	0.0132	0.5718	0.0166	0.4830	0.0122
13000	**0.5300**	0.0239	0.5035	0.0296	0.5508	0.0233	0.5009	0.0407
14000	0.5196	0.0141	0.4843	0.0122	0.5851	0.0195	0.4882	0.0119
15000	0.5257	0.0145	**0.5101**	0.0177	**0.5927**	0.0237	**0.5073**	0.0204

Fuente: *Elaboración propia*

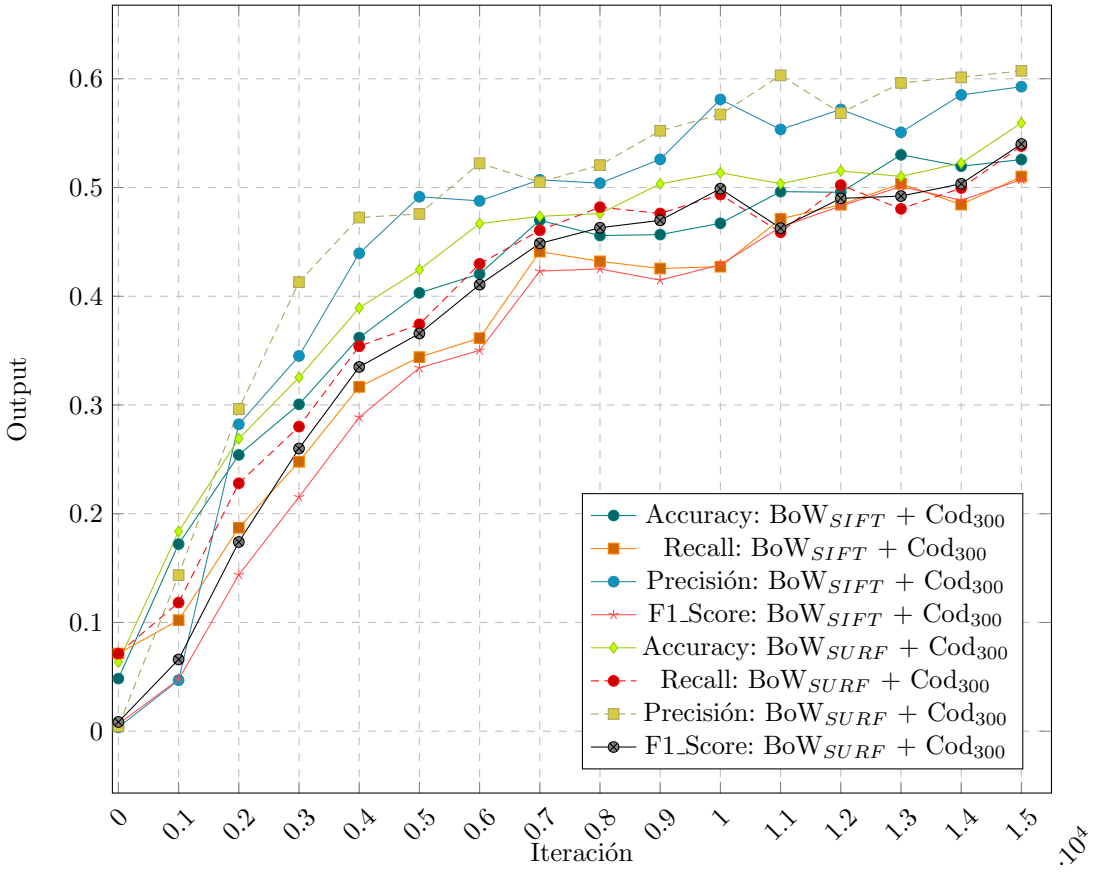
Los mejores resultados de la Tabla 30 son equivalentes a **53.00 %**, **51.01 %**, **59.27 %** y **50.73 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, en la Tabla 31 se observa que los valores mas elevados como **55.94 %, 53.82 %, 60.73 %** y **54.00 %** se presentan en la iteración 15000 en todos los casos. Por otro lado, se puede ver en la Figura 72 el detalle de estos valores.

Tabla 31: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network (NN) durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 300.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0636	0.0161	0.0714	0.0000	0.0045	0.0011	0.0085	0.0020
1000	0.1838	0.0270	0.1183	0.0027	0.1435	0.0782	0.0660	0.0092
2000	0.2690	0.0229	0.2280	0.0405	0.2963	0.0605	0.1740	0.0481
3000	0.3253	0.0437	0.2801	0.0491	0.4130	0.0727	0.2600	0.0467
4000	0.3893	0.0255	0.3540	0.0401	0.4724	0.0103	0.3350	0.0279
5000	0.4244	0.0165	0.3742	0.0095	0.4757	0.0255	0.3657	0.0106
6000	0.4668	0.0038	0.4298	0.0265	0.5223	0.0365	0.4106	0.0268
7000	0.4735	0.0339	0.4606	0.0356	0.5049	0.0411	0.4486	0.0307
8000	0.4760	0.0068	0.4819	0.0085	0.5206	0.0280	0.4630	0.0113
9000	0.5034	0.0203	0.4761	0.0233	0.5522	0.0394	0.4698	0.0236
10000	0.5136	0.0164	0.4935	0.0272	0.5672	0.0126	0.4989	0.0123
11000	0.5035	0.0126	0.4588	0.0341	0.6031	0.0351	0.4626	0.0197
12000	0.5152	0.0119	0.5022	0.0232	0.5684	0.0233	0.4900	0.0134
13000	0.5102	0.0321	0.4804	0.0494	0.5962	0.0040	0.4921	0.0370
14000	0.5225	0.0059	0.4995	0.0231	0.6013	0.0141	0.5034	0.0171
15000	**0.5594**	0.0009	**0.5382**	0.0145	**0.6073**	0.0072	**0.5400**	0.0127

Fuente: *Elaboración propia*

Figura 72: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF, sobre un tamaño de codebook (Cod_{size}) equivalente a 300.*



Fuente: *Elaboración propia*

De la Figura 72 se observa la evolución de los extractores de características durante cada iteración. Por otro lado, en la Tabla 32 y 33 se aplico la técnica *Neruonal Network* sobre los extractores de características SIFT y SURF respectivamente, utilizando un conjunto de tamaño fijo de *codebook* equivalente a 500.

Tabla 32: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 500.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0531	0.0719	0.0241	0.0080
1000	0.1461	0.0823	0.0537	0.0345
2000	0.2303	0.1293	0.0583	0.0708
3000	0.2263	0.1604	0.2814	0.1346
4000	0.3022	0.2470	0.3090	0.2114
5000	0.3504	0.2843	0.4445	0.2576
6000	0.3952	0.3543	0.3933	0.3036
7000	0.4272	0.3603	0.4851	0.3435
8000	0.4246	0.3705	0.5050	0.3497
9000	0.4921	0.4770	0.5488	0.4696
10000	0.4969	0.4576	0.5404	0.4603
11000	0.5018	0.4562	0.5225	0.4320
12000	0.5000	0.4842	0.5741	0.4744
13000	0.5289	0.4975	0.6056	**0.5030**
14000	**0.5461**	0.4808	**0.6530**	0.4737
15000	0.5154	**0.4983**	0.5660	0.4766
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1307	0.0741	0.0129	0.0212
1000	0.1798	0.1010	0.0482	0.0465
2000	0.1636	0.1200	0.1137	0.0488
3000	0.2553	0.1638	0.3521	0.1370
4000	0.3000	0.2832	0.4615	0.2428
5000	0.3246	0.2867	0.5124	0.2328
6000	0.3728	0.3048	0.4709	0.2937
7000	0.4158	0.4018	0.5117	0.3858
8000	0.4110	0.3535	0.3900	0.3292
9000	0.4855	0.4506	0.4955	0.4358
10000	0.3947	0.3796	0.4826	0.3415
11000	0.4811	0.4603	0.5960	0.4430
12000	0.5276	0.4712	0.5985	0.4700
13000	0.5272	**0.5073**	0.6014	0.4829
14000	0.5114	0.4727	0.5804	0.4444
15000	**0.5294**	0.5013	**0.6458**	**0.5096**
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0009	0.0012	0.0078	0.0020
1000	0.1294	0.1069	0.0321	0.0411
2000	0.1974	0.1683	0.2177	0.1055
3000	0.2061	0.1228	0.1976	0.0852
4000	0.2768	0.2515	0.3920	0.2151
5000	0.4022	0.3500	0.4141	0.3335
6000	0.3482	0.3424	0.3627	0.2663
7000	0.4154	0.3531	0.5182	0.3422
8000	0.4285	0.3845	0.5082	0.3790
9000	0.4623	0.4072	0.5328	0.4031
10000	0.4741	0.4479	0.5865	0.4361
11000	0.4689	0.4335	0.5858	0.4141
12000	0.4667	0.4243	0.6093	0.4278
13000	0.5026	0.4819	0.6179	0.4867
14000	0.5325	0.4861	**0.6298**	0.4843
15000	**0.5443**	**0.5178**	0.6098	**0.5252**

Fuente: *Elaboración propia*

Tabla 33: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 500.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0614	0.0714	0.0044	0.0083
1000	0.1544	0.1379	0.0847	0.0566
2000	0.2417	0.1450	0.3274	0.1082
3000	0.2754	0.2141	0.3859	0.1747
4000	0.3075	0.2793	0.4994	0.2646
5000	0.3961	0.3650	0.5037	0.3422
6000	0.4219	0.3791	0.5121	0.3735
7000	0.4399	0.4015	0.4999	0.3923
8000	0.5031	0.4763	0.5145	0.4753
9000	0.4789	0.4535	0.5831	0.4495
10000	0.5162	0.4822	0.5551	0.4771
11000	0.5241	0.4822	0.5773	0.4677
12000	0.5333	0.5061	**0.5959**	0.5161
13000	0.5373	0.4974	0.5912	0.4997
14000	0.5439	0.5305	0.5853	0.5250
15000	**0.5579**	**0.5437**	0.5692	**0.5275**
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0531	0.0714	0.0038	0.0072
1000	0.2039	0.1238	0.0996	0.0689
2000	0.1772	0.1592	0.1239	0.0745
3000	0.2601	0.1822	0.2646	0.1336
4000	0.3246	0.2802	0.4796	0.2689
5000	0.3969	0.3536	0.4244	0.3134
6000	0.4202	0.3888	0.4833	0.3671
7000	0.4386	0.3799	0.5707	0.3567
8000	0.4469	0.4163	0.5338	0.4117
9000	0.5092	0.4778	0.5443	0.4654
10000	0.4259	0.4174	0.5516	0.4155
11000	0.5092	0.4691	0.5780	0.4654
12000	0.5009	0.4547	**0.5997**	0.4640
13000	0.5114	0.5145	0.5774	0.5039
14000	0.5175	0.5016	0.5797	0.4927
15000	**0.5496**	**0.5493**	0.5720	**0.5406**
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0715	0.0714	0.0051	0.0095
1000	0.1338	0.0714	0.0096	0.0169
2000	0.1395	0.0751	0.0546	0.0242
3000	0.3083	0.2579	0.3428	0.2160
4000	0.3241	0.2807	0.4323	0.2416
5000	0.3851	0.3470	0.5052	0.3281
6000	0.4276	0.3749	0.4957	0.3713
7000	0.4377	0.4266	0.5466	0.4190
8000	0.4487	0.4273	0.5177	0.4153
9000	0.4697	0.4530	0.5302	0.4401
10000	0.4811	0.4718	0.4887	0.4358
11000	0.4996	0.4920	0.5796	0.4961
12000	0.5219	0.4835	**0.6028**	0.4928
13000	0.5154	0.5308	0.5776	0.5161
14000	**0.5404**	**0.5411**	0.5684	**0.5180**
15000	0.5329	0.5166	0.5871	0.5169

Fuente: *Elaboración propia*

En la Tabla 32 se observa que los mejores resultados para Split 1 son los valores equivalentes a **54.61 %, 49.83 %, 65.30 %** y **50.30 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.94 %, 50.73 %, 64.58 %** y **50.96 %** durante la iteración 15000, 13000, 15000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **54.43 %, 51.78 %, 62.98 %** y **52.52 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 34.

Tabla 34: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 500.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0615	0.0653	0.0491	0.0415	0.0150	0.0084	0.0104	0.0098
1000	0.1518	0.0257	0.0967	0.0128	0.0447	0.0112	0.0407	0.0060
2000	0.1971	0.0333	0.1392	0.0256	0.1299	0.0810	0.0750	0.0286
3000	0.2292	0.0247	0.1490	0.0228	0.2770	0.0774	0.1189	0.0293
4000	0.2930	0.0141	0.2605	0.0197	0.3875	0.0764	0.2231	0.0171
5000	0.3591	0.0395	0.3070	0.0373	0.4570	0.0503	0.2746	0.0525
6000	0.3721	0.0235	0.3339	0.0259	0.4090	0.0558	0.2879	0.0193
7000	0.4194	0.0067	0.3717	0.0263	0.5050	0.0176	0.3572	0.0248
8000	0.4213	0.0092	0.3695	0.0155	0.4677	0.0674	0.3526	0.0250
9000	0.4800	0.0157	0.4449	0.0353	0.5257	0.0274	0.4362	0.0332
10000	0.4553	0.0536	0.4283	0.0425	0.5365	0.0521	0.4126	0.0628
11000	0.4839	0.0166	0.4500	0.0145	0.5681	0.0398	0.4297	0.0146
12000	0.4981	0.0305	0.4599	0.0315	0.5940	0.0180	0.4574	0.0258
13000	0.5196	0.0147	0.4956	0.0128	0.6083	0.0086	0.4909	0.0106
14000	**0.5300**	0.0175	0.4798	0.0067	**0.6211**	0.0371	0.4675	0.0206
15000	0.5297	0.0145	**0.5058**	0.0105	0.6072	0.0400	**0.5038**	0.0248

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 34 son equivalentes a **53.00 %**, **50.58 %**, **62.11 %** y **50.38 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente.

En la Tabla 33 se observa que los mejores resultados para Split 1 son los valores equivalentes a **55.79 %, 54.37 %, 59.59 %** y **52.75 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **54.96 %, 54.93 %, 59.97 %** y **54.06 %** durante la iteración 15000, 15000, 12000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **54.04 %, 54.11 %, 60.28 %** y **51.80 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 35.

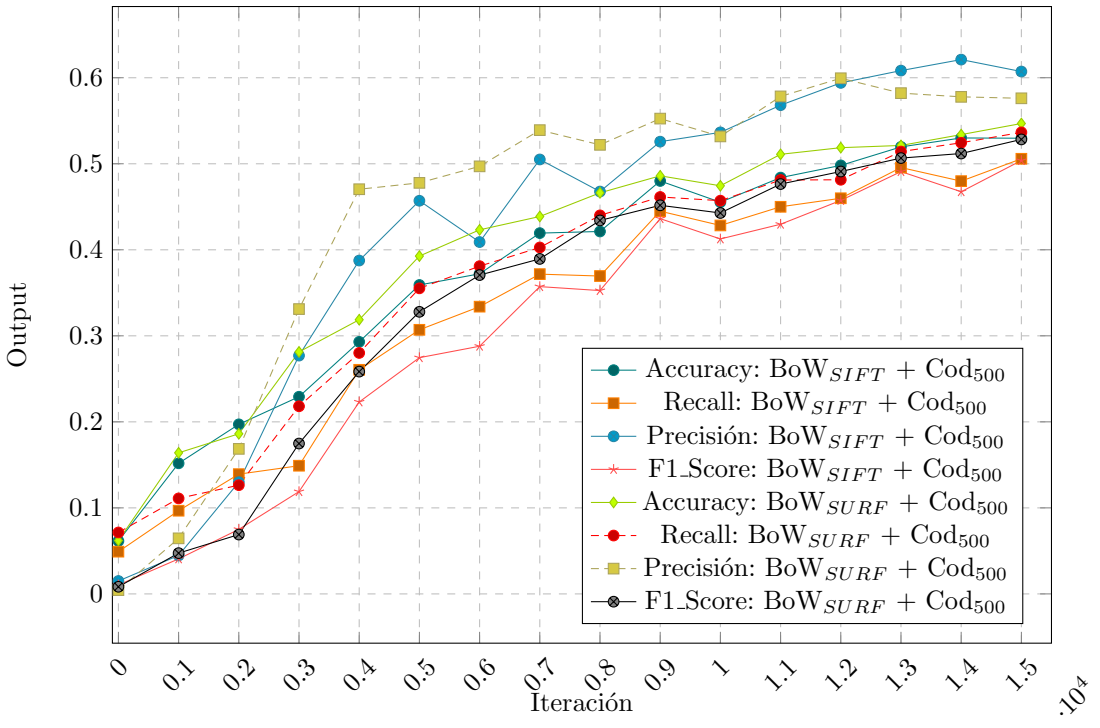
En la Tabla 35 se observa que los valores más elevados para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* corresponde a **54.68 %**, **53.65 %**, **59.94 %** y **52.83 %** respectivamente. En la Figura 73 se observa a detalle los resultados de las Tablas 34 y 35. Por otro lado, en las Tablas 36 y 37 se aplico la técnica Neuronal Network sobre los extractores de características SIFT y SURF respectivamente, utilizando un *codebook* (Cod_{size}) equivalente a 700.

Tabla 35: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 500.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0620	0.0092	0.0714	0.0000	0.0044	0.0007	0.0083	0.0012
1000	0.1640	0.0361	0.1110	0.0350	0.0646	0.0482	0.0474	0.0272
2000	0.1861	0.0517	0.1264	0.0450	0.1686	0.1418	0.0690	0.0423
3000	0.2813	0.0246	0.2181	0.0380	0.3311	0.0615	0.1748	0.0412
4000	0.3187	0.0098	0.2801	0.0007	0.4704	0.0345	0.2584	0.0147
5000	0.3927	0.0066	0.3552	0.0091	0.4778	0.0462	0.3279	0.0144
6000	0.4232	0.0039	0.3810	0.0071	0.4970	0.0145	0.3706	0.0032
7000	0.4387	0.0011	0.4027	0.0233	0.5391	0.0360	0.3894	0.0313
8000	0.4662	0.0319	0.4400	0.0320	0.5220	0.0104	0.4341	0.0357
9000	0.4860	0.0207	0.4614	0.0142	0.5525	0.0274	0.4517	0.0128
10000	0.4744	0.0455	0.4571	0.0348	0.5318	0.0374	0.4428	0.0314
11000	0.5110	0.0124	0.4811	0.0115	0.5783	0.0012	0.4764	0.0171
12000	0.5187	0.0165	0.4814	0.0257	**0.5994**	0.0035	0.4910	0.0261
13000	0.5213	0.0139	0.5142	0.0167	0.5821	0.0079	0.5066	0.0085
14000	0.5339	0.0143	0.5244	0.0204	0.5778	0.0086	0.5119	0.0170
15000	**0.5468**	0.0127	**0.5365**	0.0175	0.5761	0.0097	**0.5283**	0.0119

Fuente: *Elaboración propia*

Figura 73: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF con un tamaño de codebook (Cod_{size}) equivalente a 500.*



Fuente: *Elaboración propia*

Tabla 36: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 700.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0355	0.0714	0.0025	0.0049
1000	0.1754	0.0909	0.0258	0.0401
2000	0.2000	0.1150	0.1112	0.0592
3000	0.2276	0.1698	0.2442	0.1114
4000	0.2614	0.2195	0.2787	0.1489
5000	0.3443	0.2891	0.4155	0.2478
6000	0.3750	0.3077	0.3802	0.2877
7000	0.3702	0.3010	0.4517	0.3075
8000	0.4162	0.3855	0.4916	0.3527
9000	0.4522	0.4497	0.4887	0.4267
10000	0.4364	0.3573	0.5972	0.3751
11000	0.4618	0.4415	0.5440	0.4184
12000	0.5018	0.4128	0.5835	0.4253
13000	0.5004	0.4591	0.5915	0.4594
14000	**0.5632**	**0.5364**	0.5695	**0.5327**
15000	0.5456	0.4821	**0.6250**	0.4916
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0430	0.0759	0.0031	0.0059
1000	0.1316	0.0714	0.0094	0.0166
2000	0.1570	0.0856	0.0854	0.0385
3000	0.2132	0.1188	0.0607	0.0642
4000	0.2728	0.1770	0.3652	0.1521
5000	0.2803	0.2246	0.4343	0.1999
6000	0.3811	0.3703	0.4635	0.3614
7000	0.3838	0.3365	0.4315	0.3236
8000	0.4640	0.4260	0.5184	0.4150
9000	0.4680	0.4321	0.4897	0.4248
10000	0.4377	0.4027	0.5333	0.3626
11000	0.4768	0.4631	0.4737	0.4043
12000	0.4798	0.4436	0.5278	0.4222
13000	0.4732	0.4123	0.5903	0.4196
14000	0.5149	0.4733	0.5841	0.4833
15000	**0.5320**	**0.4745**	**0.6180**	**0.4963**
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0684	0.0714	0.0049	0.0091
1000	0.1338	0.0863	0.0096	0.0169
2000	0.2189	0.1337	0.1867	0.0824
3000	0.1961	0.1102	0.1469	0.0497
4000	0.2649	0.2102	0.3418	0.1594
5000	0.3193	0.2873	0.3746	0.2445
6000	0.3798	0.3661	0.4509	0.3338
7000	0.3737	0.3338	0.4500	0.3249
8000	0.4579	0.3990	0.4122	0.3865
9000	0.3934	0.3938	0.4265	0.3567
10000	0.4272	0.3455	0.5018	0.3584
11000	0.5110	0.4712	0.5612	0.4777
12000	0.5167	0.5149	0.5265	0.5060
13000	0.5281	0.4853	0.5539	0.4746
14000	0.5368	0.5144	**0.6133**	0.5199
15000	**0.5654**	**0.5338**	0.5601	**0.5199**

Fuente: *Elaboración propia*

Tabla 37: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 700.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0430	0.0714	0.0031	0.0059
1000	0.1342	0.0714	0.0096	0.0169
2000	0.2298	0.1646	0.1374	0.1051
3000	0.2193	0.1515	0.3349	0.1081
4000	0.2794	0.1978	0.3451	0.1467
5000	0.2763	0.2317	0.4139	0.2082
6000	0.4061	0.3386	0.4315	0.3058
7000	0.3794	0.4027	0.4738	0.3692
8000	0.4513	0.3886	0.5303	0.3757
9000	0.4654	0.4181	0.5284	0.4151
10000	0.5250	0.4945	0.5429	0.4954
11000	0.5127	0.4945	0.5749	0.4951
12000	0.5075	0.4842	0.5830	0.4632
13000	0.5443	0.5094	0.5957	0.5278
14000	0.5320	0.4988	**0.6080**	0.4959
15000	**0.5557**	**0.5348**	0.5967	**0.5349**
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0737	0.0714	0.0053	0.0098
1000	0.1263	0.0724	0.0407	0.0177
2000	0.1825	0.0995	0.1043	0.0459
3000	0.1899	0.1067	0.0532	0.0476
4000	0.2724	0.1787	0.3082	0.1528
5000	0.3469	0.2770	0.3952	0.2723
6000	0.3965	0.3607	0.4994	0.3634
7000	0.4412	0.4119	0.4728	0.3921
8000	0.4654	0.4215	0.5543	0.4016
9000	0.4811	0.4844	0.5434	0.4665
10000	0.4939	0.4623	0.5198	0.4487
11000	0.4798	0.4616	0.5653	0.4315
12000	0.5408	0.5043	0.6010	0.5202
13000	0.5096	0.4827	**0.6321**	0.4873
14000	0.4298	0.3964	0.5904	0.4132
15000	**0.5544**	**0.5455**	0.6005	**0.5340**
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0715	0.0714	0.0051	0.0095
1000	0.1610	0.0864	0.0235	0.0334
2000	0.1895	0.1048	0.0513	0.0436
3000	0.2140	0.1253	0.1559	0.0729
4000	0.3421	0.3078	0.3279	0.2621
5000	0.2794	0.2612	0.3575	0.2033
6000	0.3807	0.3611	0.4205	0.3386
7000	0.3368	0.2755	0.5828	0.2617
8000	0.4066	0.3753	0.5857	0.3761
9000	0.4434	0.4012	0.5021	0.3706
10000	0.4706	0.4400	0.5551	0.4402
11000	0.5123	0.4534	0.5586	0.4501
12000	0.5224	0.5078	0.5601	0.5133
13000	0.5232	0.5251	0.5638	0.5165
14000	0.5281	0.4768	**0.6350**	0.4883
15000	**0.5649**	**0.5293**	0.6268	**0.5286**

Fuente: *Elaboración propia*

En la Tabla 36 se observa que los mejores resultados para Split 1 son los valores equivalentes a **56.32 %, 53.64 %, 62.50 %** y **53.27 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **53.20 %, 47.45 %, 61.80 %** y **49.63 %** durante la iteración 15000. Por último, en Split 3 se obtiene valores como **56.54 %, 53.38 %, 61.33 %** y **51.99 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 38.

Tabla 38: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 700.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0490	0.0172	0.0714	0.0000	0.0035	0.0012	0.0066	0.0022
1000	0.1469	0.0247	0.0779	0.0112	0.0149	0.0094	0.0245	0.0135
2000	0.1920	0.0317	0.1114	0.0242	0.1278	0.0526	0.0600	0.0219
3000	0.2123	0.0158	0.1329	0.0322	0.1506	0.0918	0.0751	0.0323
4000	0.2664	0.0058	0.2022	0.0224	0.3286	0.0447	0.1535	0.0053
5000	0.3146	0.0323	0.2670	0.0367	0.4081	0.0305	0.2307	0.0268
6000	0.3787	0.0032	0.3480	0.0350	0.4315	0.0449	0.3276	0.0372
7000	0.3759	0.0071	0.3238	0.0197	0.4444	0.0112	0.3187	0.0097
8000	0.4461	0.0260	0.4035	0.0206	0.4741	0.0552	0.3847	0.0312
9000	0.4379	0.0393	0.4252	0.0286	0.4683	0.0362	0.4028	0.0399
10000	0.4338	0.0057	0.3685	0.0302	0.5441	0.0486	0.3654	0.0087
11000	0.4832	0.0252	0.4586	0.0153	0.5263	0.0464	0.4335	0.0389
12000	0.4994	0.0185	0.4571	0.0524	0.5459	0.0326	0.4511	0.0476
13000	0.5006	0.0274	0.4522	0.0370	0.5786	0.0214	0.4512	0.0284
14000	0.5383	0.0242	**0.5080**	0.0320	0.5890	0.0223	**0.5120**	0.0256
15000	**0.5477**	0.0168	0.4968	0.0323	**0.6010**	0.0356	0.5026	0.0152

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 38 son equivalentes a **54.77 %**, **50.80 %**, **60.10 %** y **51.20 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente.

En la Tabla 37 se observa que los mejores resultados para Split 1 son los valores equivalentes a **55.57 %, 53.48 %, 60.80 %** y **53.49 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **55.44 %, 54.55 %, 63.21 %** y **53.40 %** durante la iteración 15000, 15000, 13000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **56.49 %, 52.93 %, 63.50 %** y **51.65 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 39.

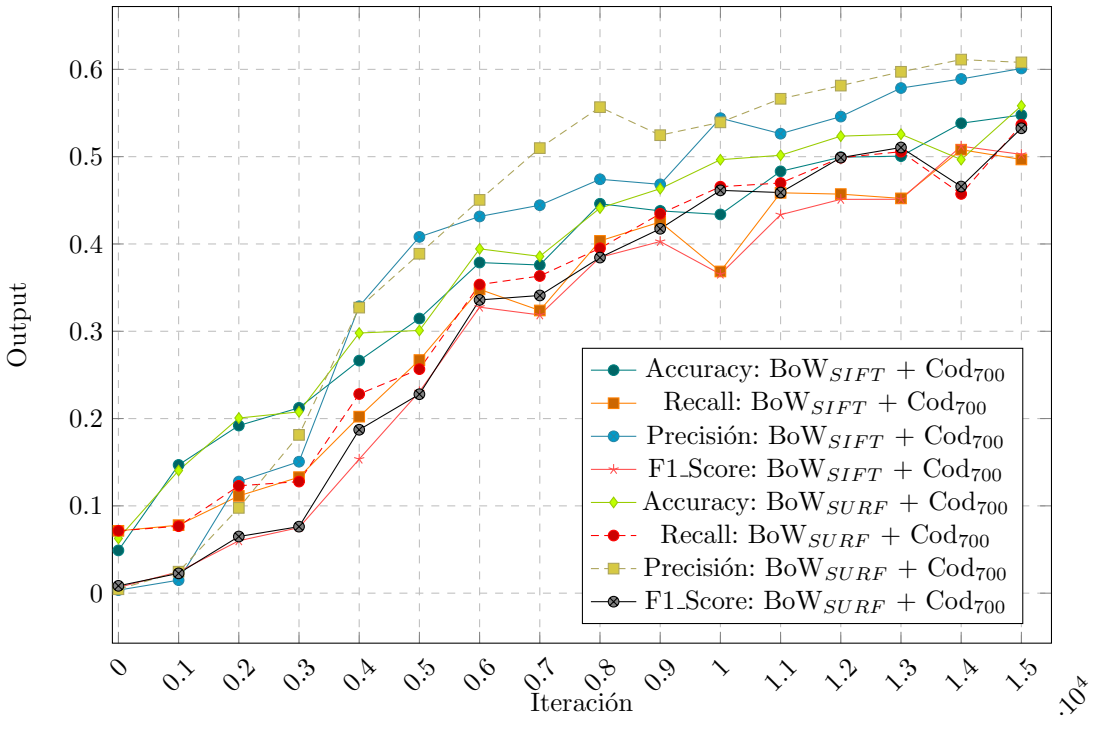
En la Tabla 39 se observa que los valores más elevados para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* corresponde a **55.83 %**, **53.65 %**, **61.12 %** y **53.25 %** respectivamente. En la Figura 74 se observa a detalle los resultados de las Tablas 38 y 39. Por otro lado, en las Tablas 40 y 41 se aplico la técnica Neuronal Network sobre los extractores de características SIFT y SURF respectivamente, utilizando un *codebook* (Cod_{size}) equivalente a 900.

Tabla 39: *Promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 700.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0627	0.0171	0.0714	0.0000	0.0045	0.0012	0.0084	0.0022
1000	0.1405	0.0182	0.0767	0.0084	0.0246	0.0156	0.0227	0.0093
2000	0.2006	0.0256	0.1230	0.0361	0.0977	0.0434	0.0649	0.0349
3000	0.2077	0.0157	0.1278	0.0225	0.1813	0.1426	0.0762	0.0304
4000	0.2980	0.0384	0.2281	0.0697	0.3271	0.0184	0.1872	0.0649
5000	0.3009	0.0399	0.2566	0.0230	0.3888	0.0287	0.2279	0.0385
6000	0.3944	0.0128	0.3535	0.0129	0.4505	0.0427	0.3359	0.0289
7000	0.3858	0.0525	0.3633	0.0762	0.5098	0.0632	0.3410	0.0696
8000	0.4411	0.0307	0.3951	0.0238	0.5567	0.0278	0.3844	0.0148
9000	0.4633	0.0189	0.4346	0.0440	0.5246	0.0209	0.4174	0.0480
10000	0.4965	0.0273	0.4656	0.0274	0.5393	0.0179	0.4614	0.0297
11000	0.5016	0.0189	0.4698	0.0218	0.5663	0.0082	0.4589	0.0327
12000	0.5235	0.0167	0.4988	0.0127	0.5814	0.0205	0.4989	0.0311
13000	0.5257	0.0175	0.5057	0.0214	0.5972	0.0341	0.5105	0.0209
14000	0.4966	0.0579	0.4573	0.0539	**0.6112**	0.0225	0.4658	0.0457
15000	**0.5583**	0.0057	**0.5365**	0.0083	0.6080	0.0164	**0.5325**	0.0034

Fuente: *Elaboración propia*

Figura 74: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF con un tamaño de codebook (Cod_{size}) equivalente a 700.*



Fuente: *Elaboración propia*

Tabla 40: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 900.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0614	0.0714	0.0044	0.0083
1000	0.1912	0.1112	0.0528	0.0598
2000	0.0882	0.0903	0.0260	0.0286
3000	0.1895	0.1018	0.1579	0.0525
4000	0.2105	0.1634	0.2719	0.1218
5000	0.3417	0.2996	0.4269	0.2879
6000	0.3346	0.3063	0.4287	0.2597
7000	0.4167	0.3815	0.4534	0.3762
8000	0.4140	0.3446	0.4770	0.3399
9000	0.4439	0.3520	0.5243	0.3608
10000	0.4711	0.4276	0.5461	0.4051
11000	0.5013	0.4348	0.5666	0.4459
12000	0.4864	0.4081	0.5603	0.4221
13000	0.5180	0.4757	0.5592	0.4616
14000	**0.5544**	0.4967	0.5815	0.4989
15000	0.5553	**0.5057**	**0.6121**	**0.5042**
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0689	0.0714	0.0049	0.0092
1000	0.1246	0.0714	0.0089	0.0158
2000	0.1588	0.1220	0.0451	0.0472
3000	0.1346	0.0964	0.2242	0.0433
4000	0.2456	0.1915	0.1204	0.1272
5000	0.2899	0.2423	0.2926	0.1994
6000	0.3434	0.2857	0.3933	0.2554
7000	0.3360	0.2592	0.4996	0.2556
8000	0.3921	0.3595	0.4324	0.3356
9000	0.4035	0.3327	0.4770	0.3434
10000	0.4675	0.4196	0.5196	0.4057
11000	0.4447	0.3709	0.4871	0.3744
12000	0.5004	0.4661	0.5680	0.4500
13000	0.4781	0.4705	0.5539	0.4448
14000	**0.5228**	**0.4872**	**0.5704**	0.4909
15000	0.4991	0.4730	0.5632	**0.4994**
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0500	0.0714	0.0036	0.0068
1000	0.1307	0.0717	0.0141	0.0170
2000	0.1289	0.0737	0.0520	0.0204
3000	0.2285	0.1360	0.1454	0.0816
4000	0.1982	0.1393	0.1893	0.0723
5000	0.2759	0.2663	0.3746	0.2196
6000	0.3715	0.2997	0.5097	0.2959
7000	0.3816	0.3237	0.4021	0.3095
8000	0.3789	0.3021	0.4427	0.2859
9000	0.4671	0.4208	0.3781	0.3748
10000	0.4140	0.3775	0.4216	0.3412
11000	0.5189	0.4717	0.5250	0.4526
12000	0.4961	0.4718	0.5651	0.4541
13000	0.5197	0.4983	0.5808	0.4751
14000	0.5206	0.4777	0.4833	0.4522
15000	**0.5601**	**0.5422**	**0.6032**	**0.5232**

Fuente: *Elaboración propia*

Tabla 41: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 900.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0496	0.0714	0.0035	0.0067
1000	0.1904	0.1019	0.0366	0.0479
2000	0.1342	0.0714	0.0096	0.0169
3000	0.2035	0.1106	0.1655	0.0697
4000	0.2886	0.2719	0.4167	0.2148
5000	0.3228	0.2707	0.4321	0.2231
6000	0.3741	0.3070	0.4160	0.2820
7000	0.4088	0.3348	0.3941	0.3077
8000	0.4654	0.4582	0.5071	0.4327
9000	0.4706	0.4328	0.5226	0.4124
10000	0.5004	0.4688	0.5366	0.4576
11000	0.4860	0.4513	0.5933	0.4208
12000	0.5259	0.4788	0.5913	0.4819
13000	0.5316	0.4721	0.6113	0.4718
14000	0.5193	0.4753	**0.6277**	0.4842
15000	**0.5649**	**0.5416**	0.6077	**0.5412**
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0474	0.0714	0.0034	0.0065
1000	0.0636	0.0721	0.1116	0.0098
2000	0.1259	0.0724	0.1089	0.0177
3000	0.1842	0.1062	0.1696	0.0586
4000	0.2526	0.1658	0.2530	0.1239
5000	0.2640	0.1989	0.3693	0.1674
6000	0.3864	0.3814	0.4000	0.3471
7000	0.3912	0.3447	0.5209	0.3446
8000	0.4557	0.4107	0.4154	0.3954
9000	0.4421	0.4136	0.4703	0.3951
10000	0.4846	0.4576	0.5376	0.4369
11000	0.4132	0.3659	0.5826	0.3698
12000	0.4478	0.3985	**0.6173**	0.4002
13000	0.5092	0.4842	0.5917	**0.4926**
14000	**0.5132**	0.4735	0.5738	0.4616
15000	0.4969	**0.5074**	0.5414	0.4680
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0684	0.0714	0.0049	0.0091
1000	0.0956	0.0878	0.1269	0.0356
2000	0.1768	0.1255	0.0881	0.0790
3000	0.1925	0.1406	0.1072	0.0655
4000	0.2232	0.1768	0.2650	0.1134
5000	0.3404	0.3060	0.2878	0.2347
6000	0.2417	0.2257	0.4766	0.1894
7000	0.3623	0.3231	0.5036	0.3122
8000	0.4254	0.4024	0.4384	0.3827
9000	0.4298	0.3660	0.5447	0.3875
10000	0.4351	0.3985	0.5942	0.3950
11000	0.5096	0.4803	0.5722	0.4837
12000	0.5162	0.4749	0.5596	0.4670
13000	0.5311	0.5053	0.5677	0.5005
14000	0.4965	0.4838	0.6032	0.4796
15000	**0.5333**	**0.5061**	**0.6351**	**0.5110**

Fuente: *Elaboración propia*

En la Tabla 40 se observa que los mejores resultados para Split 1 son los valores equivalentes a **55.53 %, 50.57 %, 61.21 %** y **50.42 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.28 %, 48.72 %, 57.04 %** y **49.94 %** durante la iteración 14000, 14000, 14000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **56.01 %, 54.22 %, 60.32 %** y **52.32 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 42.

Tabla 42: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 900.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0601	0.0095	0.0714	0.0000	0.0043	0.0007	0.0081	0.0012
1000	0.1488	0.0368	0.0848	0.0229	0.0253	0.0240	0.0309	0.0250
2000	0.1253	0.0354	0.0953	0.0246	0.0410	0.0135	0.0321	0.0137
3000	0.1842	0.0472	0.1114	0.0215	0.1758	0.0423	0.0591	0.0200
4000	0.2181	0.0246	0.1647	0.0261	0.1939	0.0759	0.1071	0.0302
5000	0.3025	0.0347	0.2694	0.0288	0.3647	0.0677	0.2357	0.0464
6000	0.3499	0.0192	0.2972	0.0105	0.4439	0.0597	0.2703	0.0222
7000	0.3781	0.0405	0.3214	0.0612	0.4517	0.0488	0.3138	0.0604
8000	0.3950	0.0177	0.3354	0.0298	0.4507	0.0233	0.3205	0.0300
9000	0.4382	0.0322	0.3685	0.0463	0.4598	0.0746	0.3597	0.0157
10000	0.4509	0.0320	0.4083	0.0269	0.4958	0.0656	0.3840	0.0371
11000	0.4883	0.0387	0.4258	0.0510	0.5262	0.0398	0.4243	0.0433
12000	0.4943	0.0072	0.4487	0.0353	0.5645	0.0039	0.4421	0.0174
13000	0.5053	0.0236	0.4815	0.0148	0.5646	0.0142	0.4605	0.0152
14000	0.5326	0.0189	0.4872	0.0095	0.5451	0.0538	0.4807	0.0250
15000	**0.5382**	0.0339	**0.5070**	0.0346	**0.5928**	0.0260	**0.5090**	0.0126

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 42 son equivalentes a **53.82 %**, **50.70 %**, **59.28 %** y **50.90 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente.

En la Tabla 41 se observa que los mejores resultados para Split 1 son los valores equivalentes a **56.49 %, 54.16 %, 62.77 %** y **54.12 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **51.32 %, 50.74 %, 61.73 %** y **49.26 %** durante la iteración 14000, 15000, 12000 y 13000 respectivamente. Por último, en Split 3 se obtiene valores como **53.33 %, 50.61 %, 63.51 %** y **51.10 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 43.

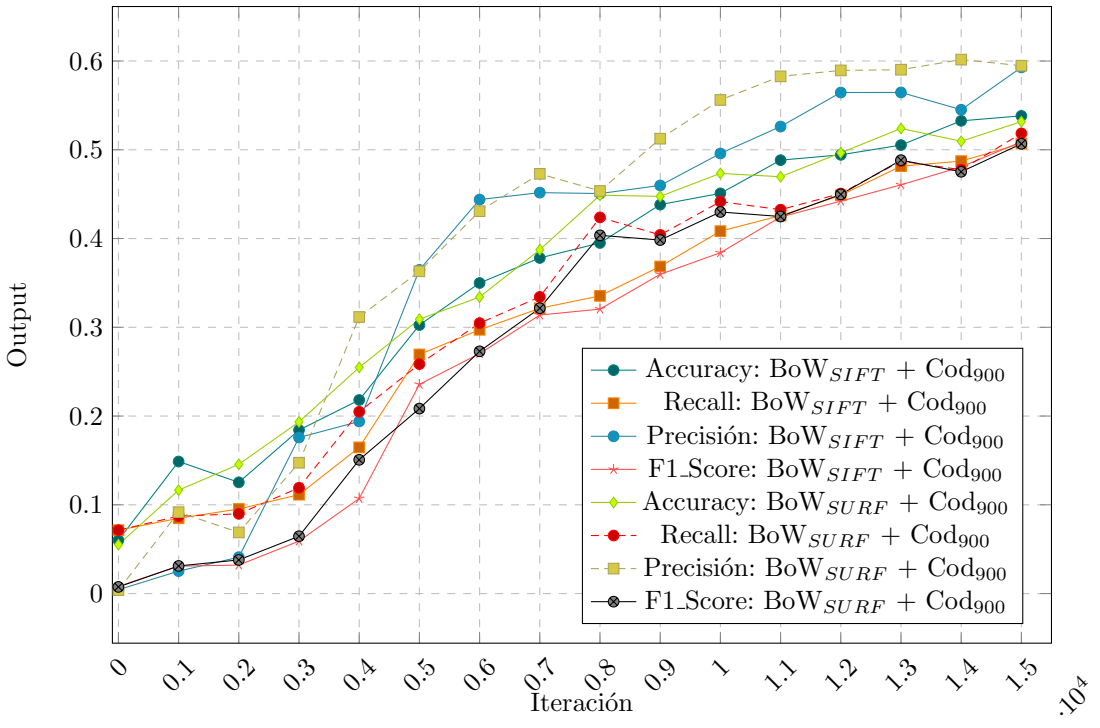
En la Tabla 43 se observa que los valores más elevados para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* corresponde a **53.17 %**, **51.84 %**, **60.16 %** y **50.67 %** respectivamente. En la Figura 75 se observa a detalle los resultados de las Tablas 42 y 43. Por otro lado, en las Tablas 44 y 45 se aplico la técnica Neuronal Network sobre los extractores de características SIFT y SURF respectivamente, utilizando un *codebook* (Cod_{size}) equivalente a 1100.

Tabla 43: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 900.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0551	0.0116	0.0714	0.0000	0.0039	0.0008	0.0075	0.0015
1000	0.1165	0.0659	0.0873	0.0149	0.0917	0.0483	0.0311	0.0194
2000	0.1456	0.0273	0.0898	0.0310	0.0689	0.0524	0.0379	0.0356
3000	0.1934	0.0097	0.1192	0.0187	0.1474	0.0349	0.0646	0.0056
4000	0.2548	0.0327	0.2048	0.0584	0.3116	0.0913	0.1507	0.0557
5000	0.3091	0.0400	0.2585	0.0546	0.3631	0.0724	0.2084	0.0360
6000	0.3341	0.0803	0.3047	0.0779	0.4308	0.0404	0.2729	0.0792
7000	0.3874	0.0235	0.3342	0.0108	0.4729	0.0688	0.3215	0.0201
8000	0.4488	0.0208	0.4238	0.0301	0.4536	0.0477	0.4036	0.0260
9000	0.4475	0.0209	0.4041	0.0344	0.5125	0.0382	0.3983	0.0127
10000	0.4734	0.0341	0.4416	0.0378	0.5561	0.0330	0.4299	0.0319
11000	0.4696	0.0503	0.4325	0.0595	0.5827	0.0105	0.4248	0.0571
12000	0.4966	0.0426	0.4507	0.0453	0.5894	0.0289	0.4497	0.0435
13000	0.5240	0.0128	0.4872	0.0168	0.5902	0.0218	0.4883	0.0149
14000	0.5096	0.0118	0.4775	0.0055	**0.6016**	0.0270	0.4752	0.0120
15000	**0.5317**	0.0340	**0.5184**	0.0201	0.5947	0.0482	**0.5067**	0.0368

Fuente: *Elaboración propia*

Figura 75: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, utilizando los extractor de características SIFT y SURF, con un tamaño de codebook (Cod_{size}) equivalente a 900.*



Fuente: *Elaboración propia*

Tabla 44: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 1100.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0355	0.0714	0.0025	0.0049
1000	0.1500	0.0844	0.1065	0.0382
2000	0.1509	0.1060	0.0408	0.0435
3000	0.2171	0.1342	0.2251	0.0937
4000	0.2557	0.1610	0.3540	0.1225
5000	0.3035	0.2618	0.3171	0.2158
6000	0.3368	0.2663	0.4411	0.2464
7000	0.4061	0.3370	0.5014	0.3282
8000	0.3680	0.3589	0.4843	0.3303
9000	0.4368	0.3929	0.4802	0.3864
10000	0.4399	0.4180	0.5043	0.3960
11000	0.4645	0.3784	0.4947	0.3601
12000	0.5022	0.4235	**0.5581**	0.4225
13000	0.5035	0.4460	0.5171	0.4391
14000	0.4930	0.4868	0.5273	0.4584
15000	**0.5197**	**0.5016**	0.5530	**0.4714**
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0531	0.0714	0.0038	0.0072
1000	0.1013	0.1094	0.0464	0.0308
2000	0.1360	0.0772	0.0142	0.0228
3000	0.1364	0.0864	0.0640	0.0383
4000	0.2061	0.1335	0.2455	0.1043
5000	0.2368	0.1407	0.1407	0.0905
6000	0.3246	0.2945	0.4059	0.2589
7000	0.3776	0.2934	0.4752	0.2977
8000	0.3807	0.3413	0.3754	0.2861
9000	0.4009	0.3268	0.5172	0.2975
10000	0.4583	0.3978	0.4681	0.3981
11000	0.4925	0.4424	0.5362	0.4477
12000	0.4719	0.4433	0.5049	0.4215
13000	0.5127	0.4683	0.5488	0.4724
14000	0.4886	0.4222	**0.5833**	0.4210
15000	**0.5224**	**0.4778**	0.5494	**0.4553**
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0684	0.0714	0.0049	0.0091
1000	0.1390	0.0761	0.0361	0.0254
2000	0.1873	0.1038	0.0268	0.0425
3000	0.1825	0.1241	0.1691	0.0826
4000	0.2061	0.1264	0.2525	0.1264
5000	0.3461	0.3381	0.3644	0.2860
6000	0.2627	0.2402	0.2387	0.1791
7000	0.3965	0.3318	0.4153	0.3229
8000	0.3754	0.3088	0.4660	0.2881
9000	0.3719	0.3422	0.4394	0.3019
10000	0.4404	0.4030	0.4977	0.4013
11000	0.4654	0.3983	0.5088	0.4013
12000	0.4246	0.3610	0.5631	0.3488
13000	0.4838	0.4062	0.5797	0.3986
14000	**0.5197**	**0.5065**	0.5290	**0.4799**
15000	0.4864	0.4351	**0.6261**	0.4392

Fuente: *Elaboración propia*

Tabla 45: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 1100.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0430	0.0714	0.0031	0.0059
1000	0.1991	0.1042	0.0285	0.0440
2000	0.1864	0.0968	0.0293	0.0445
3000	0.2719	0.1991	0.1355	0.1355
4000	0.2092	0.1370	0.1422	0.0797
5000	0.2364	0.1744	0.3682	0.1348
6000	0.3925	0.3827	0.3719	0.3206
7000	0.4096	0.3646	0.3629	0.3328
8000	0.3645	0.2843	0.4807	0.2885
9000	0.4263	0.4122	0.5397	0.3868
10000	0.4430	0.3686	0.4613	0.3490
11000	0.4346	0.4242	0.5219	0.4096
12000	0.4684	0.4473	0.5310	0.4350
13000	0.5189	0.4767	0.5771	0.4765
14000	0.4833	0.4566	0.5625	0.4561
15000	**0.5583**	**0.5127**	**0.5968**	**0.5149**
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0737	0.0714	0.0053	0.0098
1000	0.1311	0.0714	0.0094	0.0166
2000	0.1750	0.1422	0.0402	0.0605
3000	0.2171	0.1993	0.2984	0.1294
4000	0.2746	0.2070	0.1609	0.1533
5000	0.3289	0.2723	0.3021	0.2378
6000	0.2873	0.2371	0.4095	0.2134
7000	0.3259	0.2793	0.3681	0.2608
8000	0.4259	0.3714	0.4250	0.3415
9000	0.4031	0.3639	0.5306	0.3417
10000	0.4504	0.4394	0.5135	0.4053
11000	0.4820	0.4581	0.5298	0.4607
12000	0.4671	0.4193	0.4969	0.4162
13000	0.4241	0.3557	**0.6670**	0.3736
14000	0.4851	0.4457	0.5269	0.4331
15000	**0.5237**	**0.4741**	0.5933	**0.4735**
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1303	0.0714	0.0093	0.0165
1000	0.0724	0.0714	0.0052	0.0096
2000	0.1276	0.0714	0.0091	0.0162
3000	0.2123	0.1229	0.1834	0.0890
4000	0.2162	0.1830	0.1723	0.1178
5000	0.1890	0.2061	0.3260	0.1643
6000	0.3544	0.2710	0.4053	0.2650
7000	0.3496	0.3181	0.4872	0.2843
8000	0.3956	0.3467	0.5090	0.3427
9000	0.4088	0.3790	0.5140	0.3660
10000	0.4763	0.4210	0.5608	0.4326
11000	0.4667	0.4216	0.4408	0.3904
12000	0.4768	0.4538	0.5345	0.4596
13000	0.4680	0.4580	0.5189	0.4297
14000	0.5158	0.4735	**0.6053**	0.4845
15000	**0.5355**	**0.4786**	0.5764	**0.4912**

Fuente: *Elaboración propia*

En la Tabla 44 se observa que los mejores resultados para Split 1 son los valores equivalentes a **51.97 %, 50.16 %, 55.81 %** y **47.14 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.24 %, 47.78 %, 58.33 %** y **45.53 %** durante la iteración 15000, 15000, 14000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **51.97 %, 50.65 %, 62.61 %** y **47.99 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 46.

Tabla 46: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 1100.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0523	0.0165	0.0714	0.0000	0.0037	0.0012	0.0071	0.0021
1000	0.1301	0.0255	0.0900	0.0173	0.0630	0.0380	0.0315	0.0064
2000	0.1580	0.0264	0.0957	0.0160	0.0272	0.0133	0.0363	0.0117
3000	0.1787	0.0405	0.1149	0.0252	0.1527	0.0818	0.0715	0.0293
4000	0.2227	0.0286	0.1403	0.0183	0.2840	0.0607	0.1177	0.0118
5000	0.2955	0.0550	0.2468	0.0995	0.2741	0.1179	0.1974	0.0990
6000	0.3080	0.0397	0.2670	0.0272	0.3619	0.1082	0.2281	0.0429
7000	0.3934	0.0145	0.3207	0.0238	0.4640	0.0441	0.3163	0.0163
8000	0.3747	0.0064	0.3363	0.0254	0.4419	0.0583	0.3015	0.0250
9000	0.4032	0.0325	0.3540	0.0346	0.4789	0.0390	0.3286	0.0501
10000	0.4462	0.0105	0.4063	0.0105	0.4900	0.0193	0.3985	0.0026
11000	0.4741	0.0160	0.4064	0.0328	0.5132	0.0211	0.4031	0.0438
12000	0.4662	0.0391	0.4093	0.0429	0.5421	0.0323	0.3976	0.0423
13000	0.5000	0.0148	0.4402	0.0315	0.5485	0.0313	0.4367	0.0370
14000	0.5004	0.0169	**0.4718**	0.0441	0.5465	0.0318	0.4531	0.0298
15000	**0.5095**	0.0200	0.4715	0.0337	**0.5762**	0.0433	**0.4553**	0.0161

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 46 son equivalentes a **50.95 %**, **47.18 %**, **57.62 %** y **45.53 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente.

En la Tabla 45 se observa que los mejores resultados para Split 1 son los valores equivalentes a **55.83 %, 51.27 %, 59.68 %** y **51.49 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.37 %, 47.41 %, 66.70 %** y **47.35 %** durante la iteración 15000, 15000, 13000 y 15000 respectivamente. Por último, en Split 3 se obtiene valores como **53.55 %, 47.86 %, 60.53 %** y **49.12 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 47.

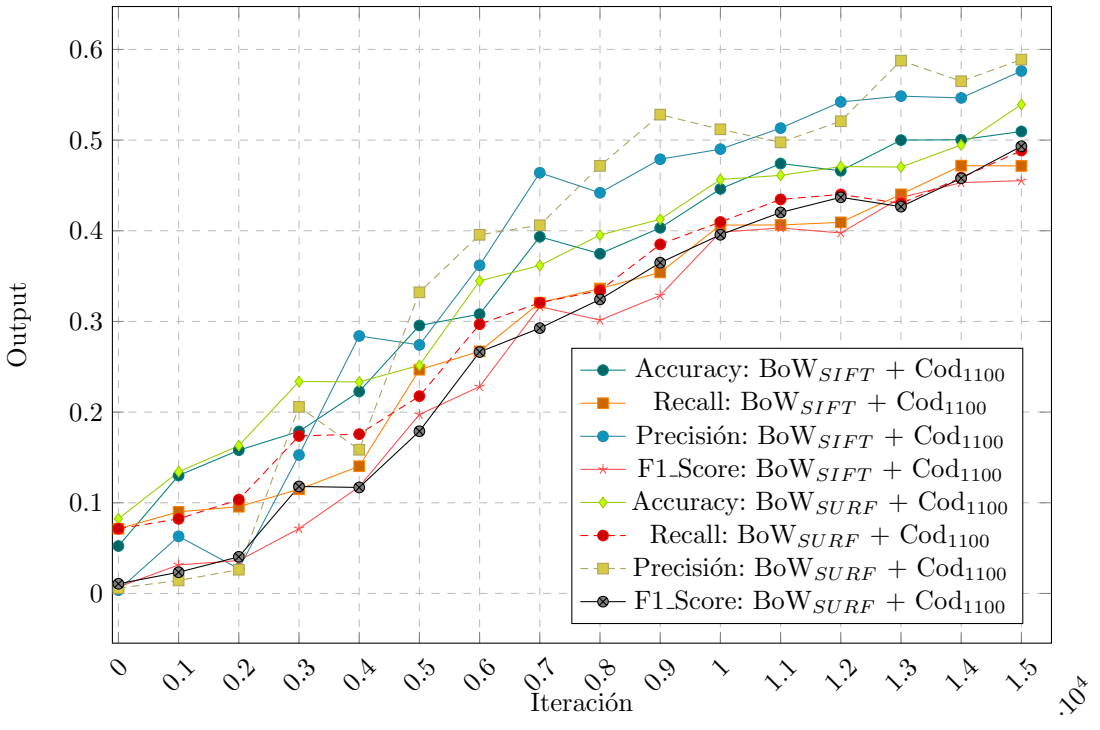
En la Tabla 47 se observa que los valores más elevados para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* corresponde a **53.92 %**, **48.85 %**, **58.88 %** y **49.32 %** respectivamente. En la Figura 76 se observa a detalle los resultados de las Tablas 46 y 47. Por otro lado, en las Tablas 48 y 49 se aplico la técnica Neuronal Network sobre los extractores de características SIFT y SURF respectivamente, utilizando *codebook* (Cod_{size}) equivalente a 1300.

Tabla 47: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 1100.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0823	0.0443	0.0714	0.0000	0.0059	0.0032	0.0107	0.0053
1000	0.1342	0.0634	0.0823	0.0189	0.0143	0.0124	0.0234	0.0182
2000	0.1630	0.0312	0.1035	0.0359	0.0262	0.0158	0.0404	0.0225
3000	0.2338	0.0331	0.1737	0.0441	0.2058	0.0837	0.1180	0.0253
4000	0.2333	0.0359	0.1757	0.0356	0.1585	0.0152	0.1169	0.0368
5000	0.2515	0.0712	0.2176	0.0499	0.3321	0.0335	0.1789	0.0531
6000	0.3447	0.0533	0.2969	0.0762	0.3956	0.0206	0.2663	0.0536
7000	0.3617	0.0432	0.3207	0.0427	0.4061	0.0703	0.2926	0.0367
8000	0.3953	0.0307	0.3341	0.0449	0.4716	0.0427	0.3243	0.0309
9000	0.4127	0.0121	0.3850	0.0247	0.5281	0.0130	0.3648	0.0226
10000	0.4566	0.0175	0.4097	0.0367	0.5119	0.0498	0.3956	0.0426
11000	0.4611	0.0242	0.4346	0.0204	0.4975	0.0493	0.4202	0.0363
12000	0.4708	0.0052	0.4401	0.0183	0.5208	0.0208	0.4369	0.0218
13000	0.4703	0.0474	0.4301	0.0652	0.5877	0.0746	0.4266	0.0515
14000	0.4947	0.0183	0.4586	0.0140	0.5649	0.0393	0.4579	0.0257
15000	**0.5392**	0.0176	**0.4885**	0.0211	**0.5888**	0.0109	**0.4932**	0.0208

Fuente: *Elaboración propia*

Figura 76: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, utilizando los extractores de características SIFT y SURF, con un tamaño de codebook (Cod_{size}) equivalente a 1100.*



Fuente: *Elaboración propia*

Tabla 48: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 1300.*

SPLIT 1: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0430	0.0714	0.0031	0.0059
1000	0.1307	0.0717	0.0807	0.0169
2000	0.1342	0.0714	0.0096	0.0169
3000	0.1654	0.1622	0.0904	0.0693
4000	0.2513	0.1404	0.1192	0.0944
5000	0.2158	0.1815	0.3640	0.1493
6000	0.2750	0.2782	0.3613	0.2453
7000	0.2706	0.2421	0.3877	0.1909
8000	0.3237	0.3008	0.4775	0.2808
9000	0.4044	0.3301	0.3606	0.2992
10000	0.4382	0.3983	0.4948	0.3772
11000	0.4855	0.4432	0.5398	0.4403
12000	0.4899	0.4629	0.5435	0.4400
13000	0.5101	0.4342	**0.6128**	0.4392
14000	0.4583	0.4395	0.5249	0.4155
15000	**0.5452**	**0.5174**	0.5754	**0.5231**
SPLIT 2: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0434	0.0714	0.0031	0.0059
1000	0.1246	0.0714	0.0089	0.0158
2000	0.2013	0.1136	0.0680	0.0677
3000	0.2057	0.1206	0.1227	0.0733
4000	0.2294	0.1590	0.0902	0.1001
5000	0.3118	0.2434	0.3458	0.2116
6000	0.3404	0.2792	0.3274	0.2446
7000	0.3206	0.3152	0.3166	0.2735
8000	0.3697	0.3328	**0.5889**	0.3228
9000	0.4184	0.3672	0.5498	0.3547
10000	0.4557	0.4385	0.5160	0.4233
11000	0.4645	0.3967	0.5478	0.3966
12000	0.4469	0.3712	0.5462	0.3740
13000	0.4702	0.4140	0.5392	0.4153
14000	**0.5219**	**0.4828**	0.5381	**0.4677**
15000	0.4943	0.4572	0.5579	0.4533
SPLIT 3: SIFT				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0627	0.0714	0.0045	0.0084
1000	0.1382	0.0789	0.0981	0.0282
2000	0.0890	0.0935	0.0696	0.0383
3000	0.1544	0.1114	0.0783	0.0412
4000	0.1978	0.1720	0.2334	0.1049
5000	0.2513	0.1893	0.3455	0.1623
6000	0.2930	0.2263	0.2606	0.1809
7000	0.3579	0.2847	0.4519	0.2873
8000	0.3390	0.3000	0.4159	0.2479
9000	0.3868	0.3664	0.4308	0.3447
10000	0.4346	0.3519	0.4962	0.3495
11000	0.4088	0.4025	0.5580	0.3562
12000	0.4579	0.4183	0.5068	0.4012
13000	0.4711	0.4648	0.4977	0.4231
14000	**0.5167**	**0.4789**	0.5682	**0.4610**
15000	0.4868	0.4435	**0.5915**	0.4249

Fuente: *Elaboración propia*

Tabla 49: *Resumen de los resultados a partir de la aplicación de Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 1300.*

SPLIT 1: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1399	0.0714	0.0100	0.0175
1000	0.1399	0.0714	0.0100	0.0175
2000	0.1439	0.0758	0.0440	0.0251
3000	0.2316	0.1373	0.1236	0.1026
4000	0.2018	0.1808	0.2906	0.1121
5000	0.2794	0.1891	0.2530	0.1399
6000	0.2425	0.2352	0.3205	0.2148
7000	0.4057	0.3434	0.4051	0.3353
8000	0.3009	0.2559	0.4111	0.2330
9000	0.4254	0.3584	0.4184	0.3348
10000	0.4215	0.3509	0.5143	0.3252
11000	0.4899	0.4488	0.5426	**0.4550**
12000	0.4895	0.4664	0.4646	0.4370
13000	0.4614	0.4316	0.5803	0.4001
14000	**0.5022**	**0.4677**	**0.6241**	0.4502
15000	0.4588	0.4271	0.5680	0.4063
SPLIT 2: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1246	0.0714	0.0089	0.0158
1000	0.1333	0.0724	0.0412	0.0186
2000	0.0768	0.0729	0.0175	0.0132
3000	0.1548	0.0875	0.0361	0.0404
4000	0.2215	0.1655	0.1512	0.1056
5000	0.2360	0.1553	0.2873	0.1129
6000	0.2912	0.2405	0.3427	0.1924
7000	0.2838	0.2173	0.3009	0.2002
8000	0.3873	0.3463	0.4861	0.3199
9000	0.3601	0.2983	0.5147	0.2848
10000	0.4461	0.4324	0.5200	0.4075
11000	0.4715	0.4565	0.4934	0.4377
12000	0.4474	0.3927	**0.6106**	0.4113
13000	**0.5145**	**0.4858**	0.5414	**0.4773**
14000	0.4789	0.4046	0.5052	0.3993
15000	0.4882	0.4547	0.5709	0.4324
SPLIT 3: SURF				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0724	0.0714	0.0052	0.0096
1000	0.1289	0.0721	0.0627	0.0176
2000	0.0610	0.0811	0.0817	0.0237
3000	0.2044	0.1444	0.1355	0.0998
4000	0.2070	0.1182	0.1970	0.0658
5000	0.2868	0.2516	0.4541	0.2056
6000	0.2667	0.2073	0.3992	0.1794
7000	0.3118	0.2886	0.3714	0.2375
8000	0.2939	0.2704	0.4799	0.2087
9000	0.3930	0.3430	0.4194	0.2939
10000	0.4246	0.4494	0.5034	0.4201
11000	0.4873	0.4360	0.5573	0.4535
12000	0.4925	0.4375	**0.6316**	0.4622
13000	0.4789	0.4607	0.5793	0.4447
14000	0.4465	0.4415	0.5643	0.4169
15000	**0.5009**	**0.4609**	0.5813	**0.4753**

Fuente: *Elaboración propia*

En la Tabla 48 se observa que los mejores resultados para Split 1 son los valores equivalentes a **54.52 %, 51.74 %, 61.28 %** y **52.31 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **52.19 %, 48.28 %, 58.89 %** y **46.77 %** durante la iteración 14000, 14000, 8000 y 14000 respectivamente. Por último, en Split 3 se obtiene valores como **51.67 %, 47.89 %, 59.15 %** y **46.10 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 50.

Tabla 50: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SIFT y un tamaño de codebook (Cod_{size}) equivalente a 1300.*

SPLIT PROMEDIO: NN + SIFT								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0497	0.0113	0.0714	0.0000	0.0036	0.0008	0.0068	0.0015
1000	0.1311	0.0068	0.0740	0.0043	0.0626	0.0473	0.0203	0.0068
2000	0.1415	0.0565	0.0928	0.0211	0.0491	0.0342	0.0410	0.0255
3000	0.1751	0.0270	0.1314	0.0271	0.0971	0.0229	0.0613	0.0175
4000	0.2262	0.0269	0.1571	0.0159	0.1476	0.0757	0.0998	0.0053
5000	0.2596	0.0486	0.2047	0.0337	0.3518	0.0106	0.1744	0.0329
6000	0.3028	0.0338	0.2612	0.0302	0.3164	0.0513	0.2236	0.0370
7000	0.3164	0.0438	0.2807	0.0367	0.3854	0.0677	0.2505	0.0521
8000	0.3442	0.0234	0.3112	0.0187	0.4941	0.0877	0.2838	0.0376
9000	0.4032	0.0158	0.3546	0.0212	0.4471	0.0956	0.3328	0.0296
10000	0.4428	0.0113	0.3962	0.0433	0.5024	0.0118	0.3833	0.0373
11000	0.4529	0.0397	0.4141	0.0253	0.5485	0.0091	0.3977	0.0421
12000	0.4649	0.0223	0.4175	0.0458	0.5322	0.0220	0.4051	0.0331
13000	0.4838	0.0228	0.4376	0.0256	0.5499	0.0583	0.4259	0.0122
14000	0.4990	0.0353	0.4671	0.0239	0.5437	0.0222	0.4481	0.0284
15000	**0.5088**	0.0317	**0.4727**	0.0393	**0.5749**	0.0168	**0.4671**	0.0506

Fuente: *Elaboración propia*

Los mejores resultados de la Tabla 50 son equivalentes a **50.88 %**, **47.27 %**, **57.49 %** y **46.71 %** como el valor más alto de la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente.

En la Tabla 49 se observa que los mejores resultados para Split 1 son los valores equivalentes a **50.22 %, 46.77 %, 62.41 %** y **45.50 %** que corresponden a la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, los mejores resultados para Split 2 son **51.45 %, 48.58 %, 61.06 %** y **47.73 %** durante la iteración 13000, 13000, 12000 y 13000 respectivamente. Por último, en Split 3 se obtiene valores como **50.09 %, 46.09 %, 63.16 %** y **47.53 %** para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. El promedio de las carpetas Split se muestra en la Tabla 51.

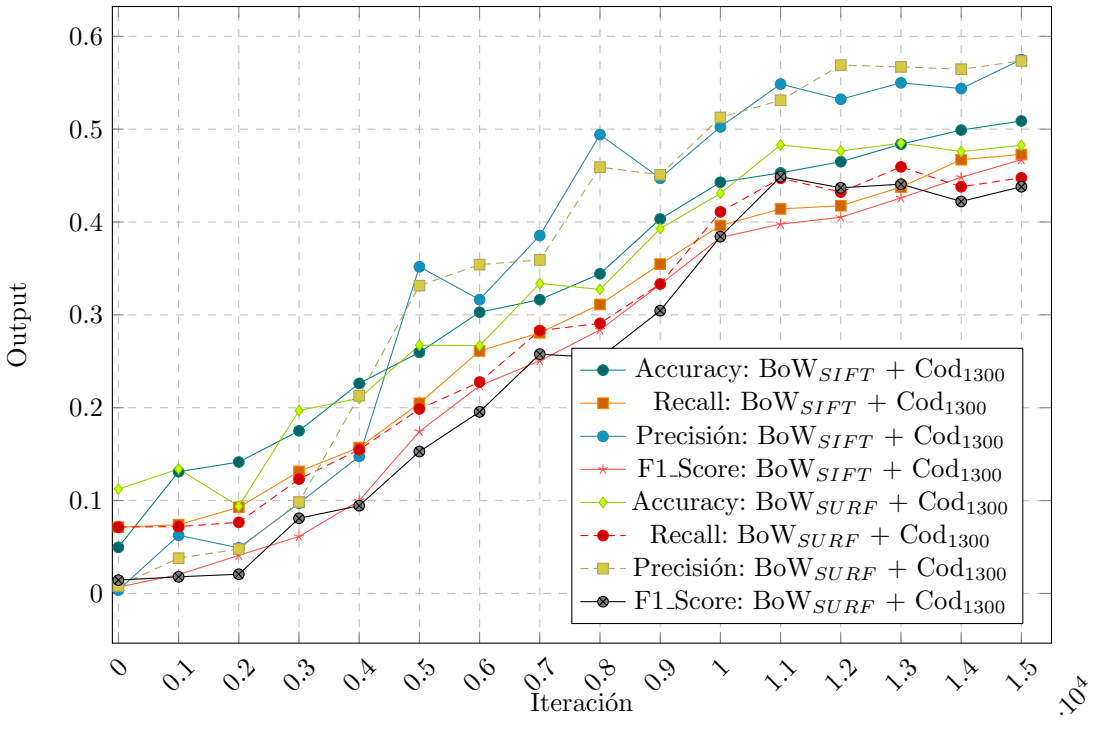
En la Tabla 51 se observa que los valores más elevados para la *Accuracy*, *Recall*, *Precisión* y *F1_Score* corresponde a **48.49 %**, **45.94 %**, **57.34 %** y **44.87 %** respectivamente. En la Figura 77 se observa a detalle los resultados de las Tablas 50 y 51.

Tabla 51: *Resumen del promedio y desviación estandar a partir de aplicar Neuronal Network durante 15000 iteraciones, utilizando un extractor de características SURF y un tamaño de codebook (Cod_{size}) equivalente a 1300.*

SPLIT PROMEDIO: NN + SURF								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.1123	0.0354	0.0714	0.0000	0.0080	0.0025	0.0143	0.0042
1000	0.1341	0.0055	0.0720	0.0005	0.0380	0.0265	0.0179	0.0006
2000	0.0939	0.0440	0.0766	0.0042	0.0478	0.0323	0.0207	0.0065
3000	0.1969	0.0389	0.1231	0.0310	0.0984	0.0542	0.0809	0.0352
4000	0.2101	0.0102	0.1548	0.0326	0.2129	0.0711	0.0945	0.0251
5000	0.2674	0.0275	0.1987	0.0488	0.3315	0.1076	0.1528	0.0477
6000	0.2668	0.0243	0.2277	0.0178	0.3541	0.0406	0.1955	0.0179
7000	0.3338	0.0639	0.2831	0.0632	0.3591	0.0532	0.2577	0.0698
8000	0.3273	0.0520	0.2908	0.0486	0.4591	0.0416	0.2539	0.0584
9000	0.3928	0.0327	0.3332	0.0312	0.4508	0.0553	0.3045	0.0266
10000	0.4307	0.0134	0.4109	0.0527	0.5126	0.0084	0.3843	0.0515
11000	0.4829	0.0100	0.4471	0.0103	0.5311	0.0335	**0.4487**	0.0096
12000	0.4765	0.0252	0.4322	0.0372	0.5689	0.0910	0.4369	0.0255
13000	**0.4849**	0.0270	**0.4594**	0.0271	0.5670	0.0222	0.4407	0.0388
14000	0.4759	0.0280	0.4379	0.0317	0.5645	0.0594	0.4221	0.0258
15000	0.4826	0.0216	0.4476	0.0180	**0.5734**	0.0070	0.4380	0.0348

Fuente: *Elaboración propia*

Figura 77: *Resultados de aplicar Neuronal Network durante 15000 iteraciones, los extractores de características SIFT y SURF con un tamaño de codebook (Cod_{size}) equivalente a 1300.*



Fuente: *Elaboración propia*

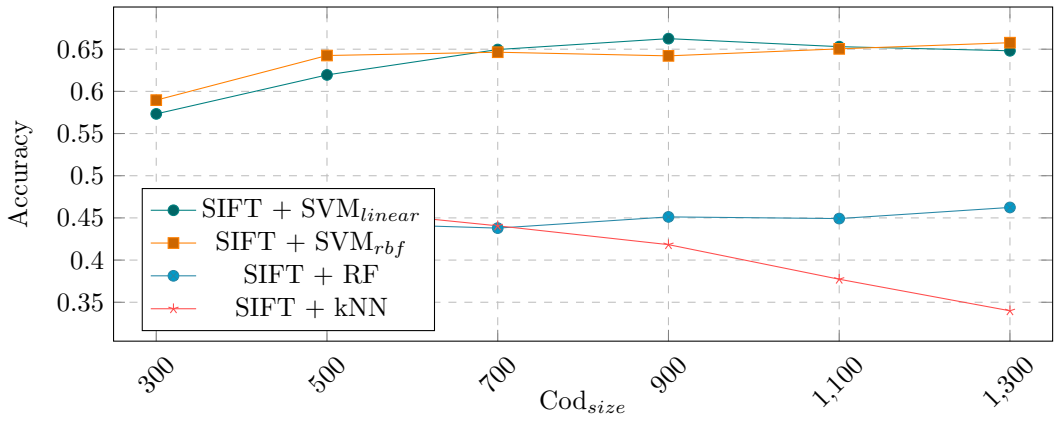
En la Tabla 52 se visualiza el resumen de los resultados de los Split Promedio para la *Accuracy* a partir del modelo *Bag-of-Words*, basado en los extractores de características SIFT y SURF, utilizando las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor*. Así mismo, en las Tablas 53 y 54 se visualizan los resultados de aplicar Neural Network sobre los extractores de características SIFT y SURF respectivamente, durante 15000 iteraciones y utilizando diferentes tamaños del *codebook*. La visualización de estos resultados se muestran en las Figuras 78, 79, 80 y 81.

Tabla 52: *Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre los extractores de características SIFT y SURF.*

Accuracy: SIFT				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5733	0.5896	0.4435	0.4510
500	0.6195	0.6425	0.4441	0.4610
700	0.6496	0.6464	0.4379	0.4406
900	**0.6625**	0.6421	0.4512	0.4183
1100	0.6530	0.6504	0.4492	0.3773
1300	0.6481	0.6577	0.4624	0.3399
Accuracy: SURF				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5755	0.5978	0.4601	0.4544
500	0.6098	0.6377	0.4532	0.4450
700	0.6355	0.6361	0.4389	0.4158
900	0.6429	0.6361	0.4459	0.3605
1100	0.6377	**0.6541**	0.4481	0.3099
1300	0.6134	0.6478	0.4444	0.2539

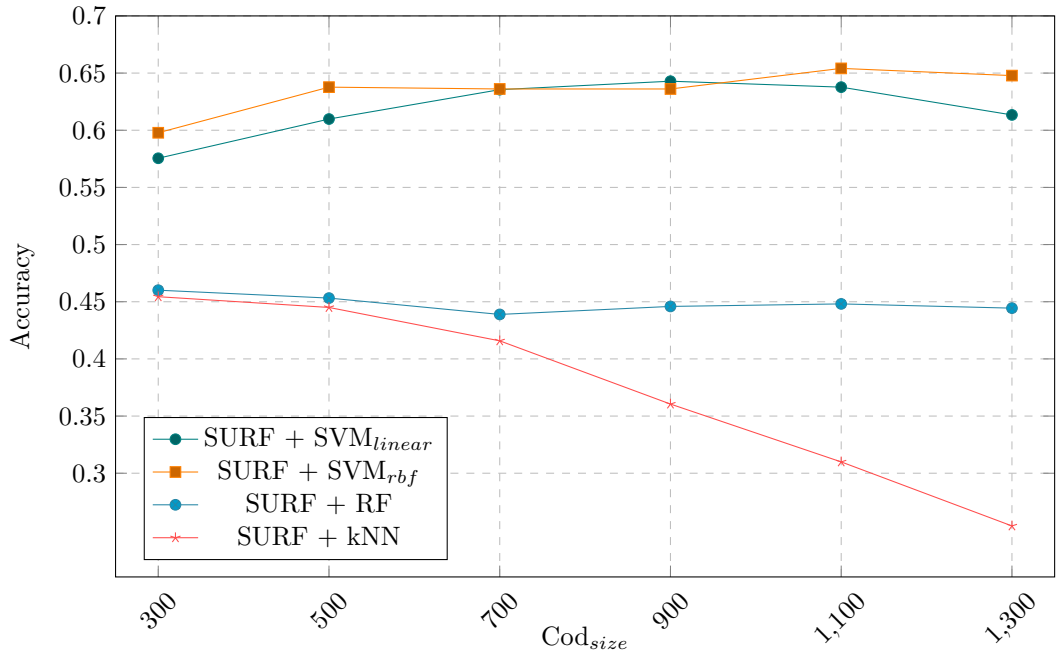
Fuente: *Elaboración propia*

Figura 78: *Resultados más elevados de SVM, RF y kNN basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT.*



Fuente: *Elaboración propia*

Figura 79: *Resultados más elevados de SVM, RF y kNN basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF.*



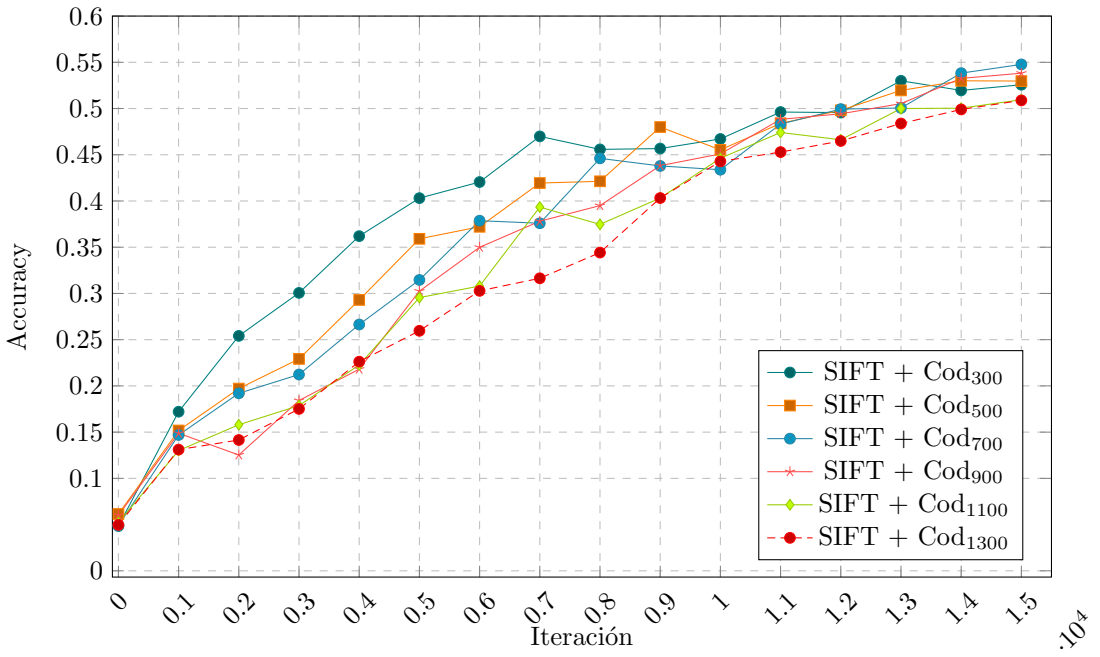
Fuente: *Elaboración propia*

Tabla 53: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT, durante 15000 iteraciones.*

Accuracy: NN + SIFT						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0484	0.0615	0.0490	0.0601	0.0523	0.0497
1000	0.1721	0.1518	0.1469	0.1488	0.1301	0.1311
2000	0.2541	0.1971	0.1920	0.1253	0.1580	0.1415
3000	0.3006	0.2292	0.2123	0.1842	0.1787	0.1751
4000	0.3620	0.2930	0.2664	0.2181	0.2227	0.2262
5000	0.4031	0.3591	0.3146	0.3025	0.2955	0.2596
6000	0.4205	0.3721	0.3787	0.3499	0.3080	0.3028
7000	0.4699	0.4194	0.3759	0.3781	0.3934	0.3164
8000	0.4558	0.4213	0.4461	0.3950	0.3747	0.3442
9000	0.4567	0.4800	0.4379	0.4382	0.4032	0.4032
10000	0.4671	0.4553	0.4338	0.4509	0.4462	0.4428
11000	0.4962	0.4839	0.4832	0.4883	0.4741	0.4529
12000	0.4956	0.4981	0.4994	0.4943	0.4662	0.4649
13000	0.5300	0.5196	0.5006	0.5053	0.5000	0.4838
14000	0.5196	0.5300	0.5383	0.5326	0.5004	0.4990
15000	0.5257	0.5297	**0.5477**	0.5382	0.5095	0.5088

Fuente: *Elaboración propia*

Figura 80: *Resultados más elevados de Neural Network basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones.*



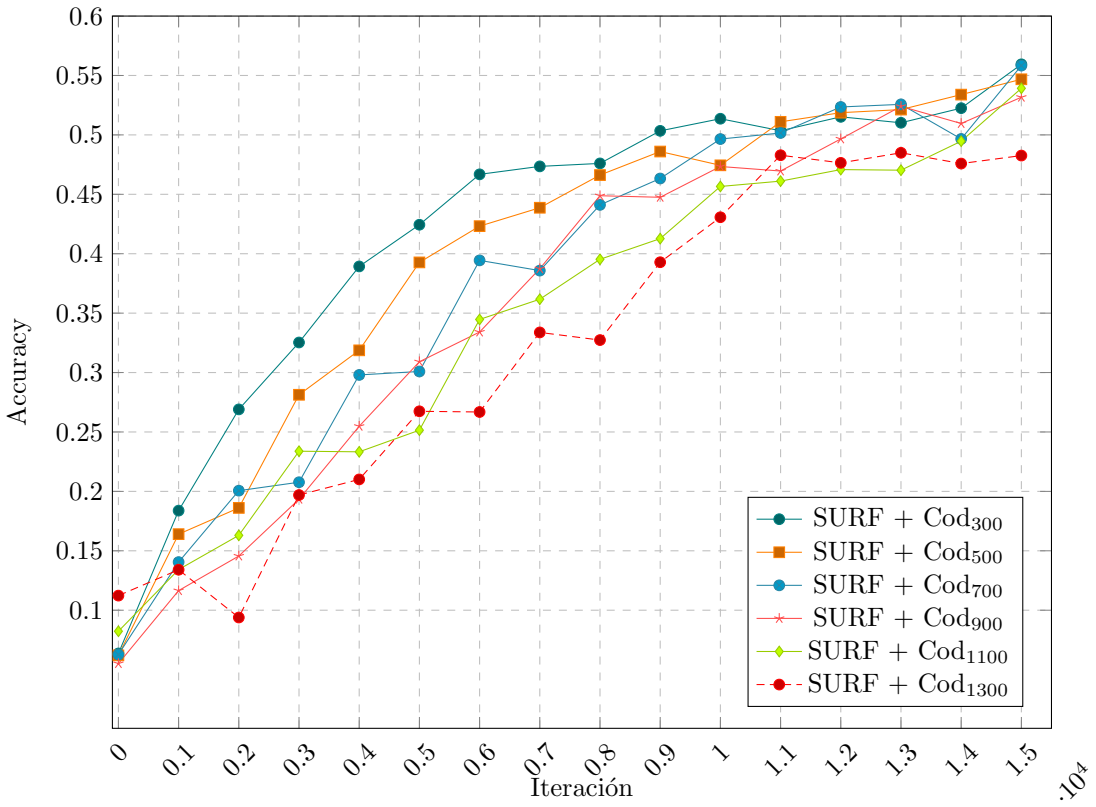
Fuente: *Elaboración propia*

Tabla 54: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF, durante 15000 iteraciones.*

Accuracy: NN + SURF						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0636	0.0620	0.0627	0.0551	0.0823	0.1123
1000	0.1838	0.1640	0.1405	0.1165	0.1342	0.1341
2000	0.2690	0.1861	0.2006	0.1456	0.1630	0.0939
3000	0.3253	0.2813	0.2077	0.1934	0.2338	0.1969
4000	0.3893	0.3187	0.2980	0.2548	0.2333	0.2101
5000	0.4244	0.3927	0.3009	0.3091	0.2515	0.2674
6000	0.4668	0.4232	0.3944	0.3341	0.3447	0.2668
7000	0.4735	0.4387	0.3858	0.3874	0.3617	0.3338
8000	0.4760	0.4662	0.4411	0.4488	0.3953	0.3273
9000	0.5034	0.4860	0.4633	0.4475	0.4127	0.3928
10000	0.5136	0.4744	0.4965	0.4734	0.4566	0.4307
11000	0.5035	0.5110	0.5016	0.4696	0.4611	0.4829
12000	0.5152	0.5187	0.5235	0.4966	0.4708	0.4765
13000	0.5102	0.5213	0.5257	0.5240	0.4703	0.4849
14000	0.5225	0.5339	0.4966	0.5096	0.4947	0.4759
15000	**0.5594**	0.5468	0.5583	0.5317	0.5392	0.4826

Fuente: *Elaboración propia*

Figura 81: *Resultados más elevados de Neural Network basado en la métrica Accuracy, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones.*



Fuente: *Elaboración propia*

De la Figura 78, se puede abstraer que la técnica de aprendizaje de máquina SVM presenta el mejor resultado equivalente a **66.25 %** al utilizar un kernel_{linear}. En específico sobre un extractor de características SIFT, cuando el $\text{Cod}_{size} = 700$. Mientras que en la Figura 79, cuando se analiza el extractor de características SURF la mejor técnica de aprendizaje vuelve a ser SVM. Sin embargo, utiliza un kernel_{rbf}, siendo el valor más elevado equivalente a **65.41 %**, utilizando un $\text{Cod}_{size} = 1100$. Por otro lado, las Figuras 80 y 81 obtienen los valores más elevados durante la iteración 15000; se puede observar en las Figuras 80 y 81 que los resultados mejoran a medida que el número de iteraciones crece.

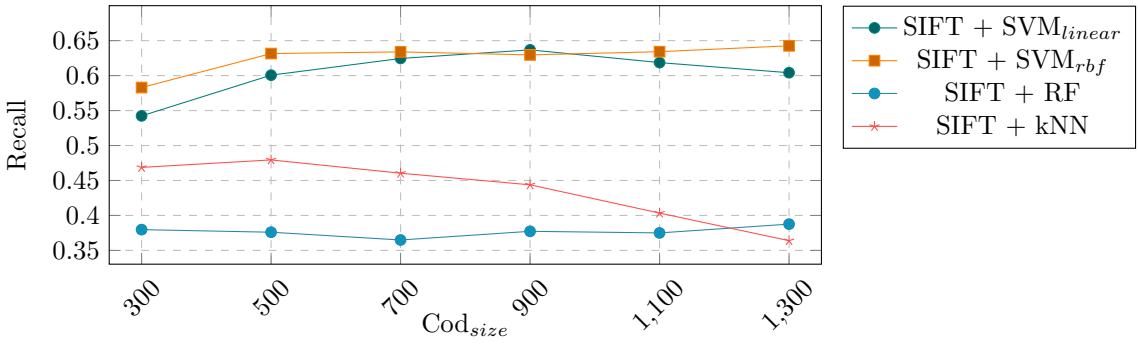
En la Tabla 55 se visualiza el resumen de los resultados de los Split Promedio para el *Recall* a partir del modelo *Bag-of-Words*, basado en los extractores de características SIFT y SURF, utilizando las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor*. Así mismo, en las Tablas 56 y 57 se visualizan los resultados de aplicar Neural Network sobre los extractores de características SIFT y SURF respectivamente, durante 15000 iteraciones y utilizando diferentes tamaños del *codebook*. La visualización de estos resultados se muestran en las Figuras 82, 83, 84 y 85.

Tabla 55: *Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre los extractores de características SIFT y SURF.*

Recall: SIFT				
Cod$_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5424	0.5829	0.3797	0.4687
500	0.6006	0.6315	0.3760	0.4794
700	0.6245	0.6339	0.3649	0.4604
900	0.6368	0.6296	0.3773	0.4438
1100	0.6186	0.6342	0.3750	0.4034
1300	0.6041	**0.6426**	0.3876	0.3639
Recall: SURF				
Cod$_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5542	0.5908	0.4019	0.4681
500	0.6011	0.6292	0.3888	0.4616
700	0.6196	0.6315	0.3696	0.4406
900	0.6178	0.6226	0.3735	0.3895
1100	0.6061	**0.6433**	0.3758	0.3441
1300	0.5740	0.6363	0.3731	0.2936

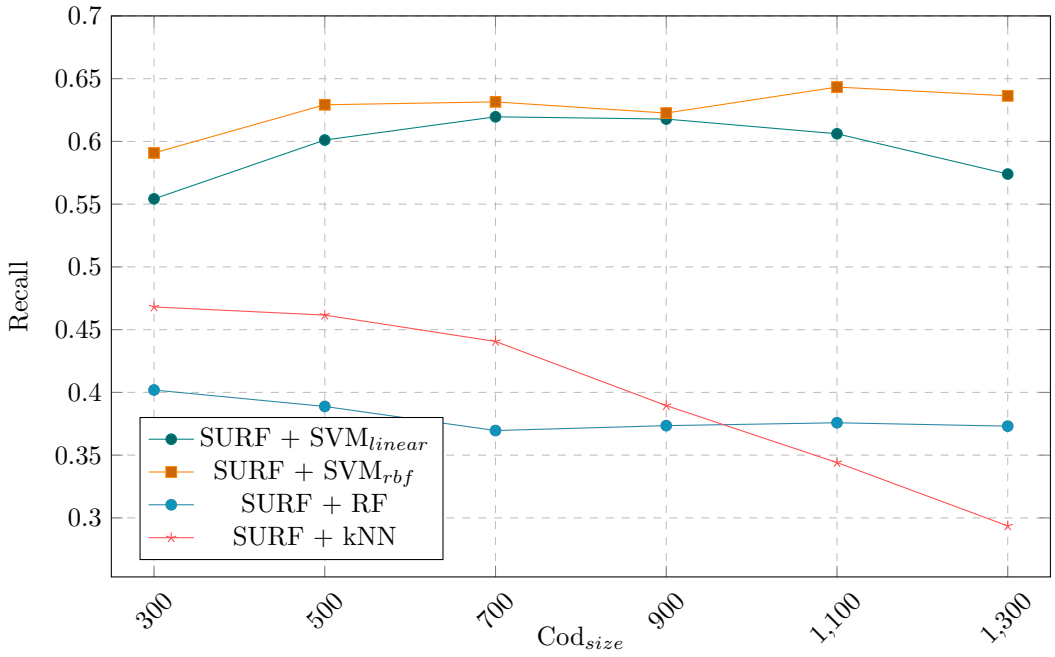
Fuente: *Elaboración propia*

Figura 82: *Resultados más elevados de SVM, RF y kNN basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT.*



Fuente: *Elaboración propia*

Figura 83: *Resultados más elevados de SVM, RF y kNN basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF.*



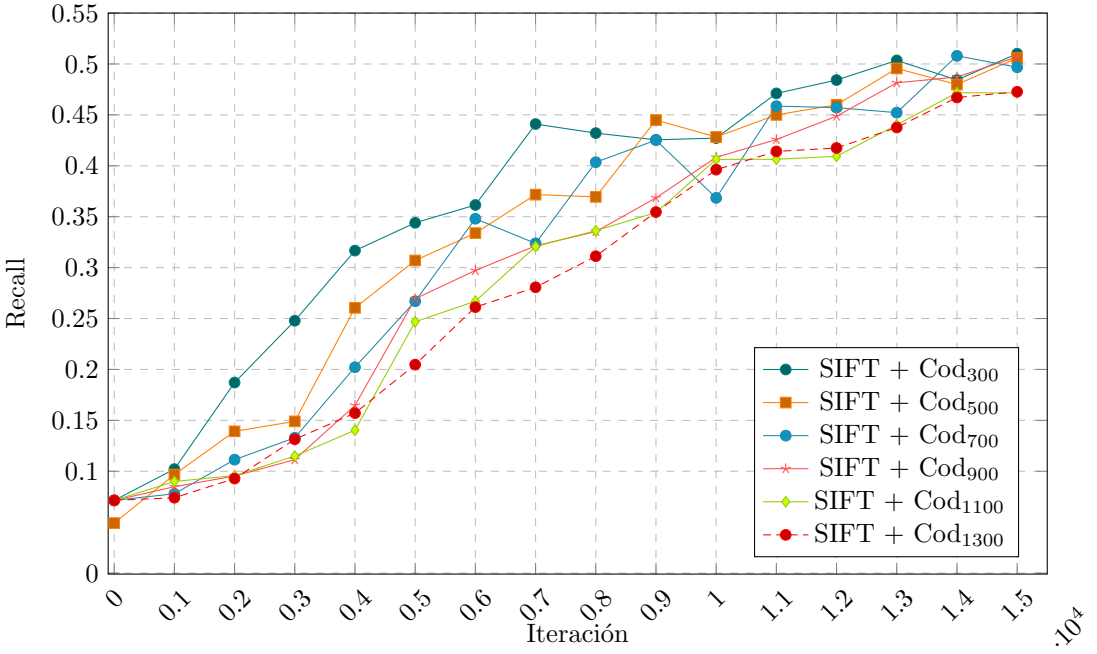
Fuente: *Elaboración propia*

Tabla 56: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT, durante 15000 iteraciones.*

Recall: NN + SIFT						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0714	0.0491	0.0714	0.0714	0.0714	0.0714
1000	0.1021	0.0967	0.0779	0.0848	0.0900	0.0740
2000	0.1871	0.1392	0.1114	0.0953	0.0957	0.0928
3000	0.2478	0.1490	0.1329	0.1114	0.1149	0.1314
4000	0.3167	0.2605	0.2022	0.1647	0.1403	0.1571
5000	0.3440	0.3070	0.2670	0.2694	0.2468	0.2047
6000	0.3615	0.3339	0.3480	0.2972	0.2670	0.2612
7000	0.4410	0.3717	0.3238	0.3214	0.3207	0.2807
8000	0.4321	0.3695	0.4035	0.3354	0.3363	0.3112
9000	0.4255	0.4449	0.4252	0.3685	0.3540	0.3546
10000	0.4272	0.4283	0.3685	0.4083	0.4063	0.3962
11000	0.4711	0.4500	0.4586	0.4258	0.4064	0.4141
12000	0.4842	0.4599	0.4571	0.4487	0.4093	0.4175
13000	0.5035	0.4956	0.4522	0.4815	0.4402	0.4376
14000	0.4843	0.4798	0.5080	0.4872	0.4718	0.4671
15000	**0.5101**	0.5058	0.4968	0.5070	0.4715	0.4727

Fuente: *Elaboración propia*

Figura 84: *Resultados más elevados de Neural Network basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones.*



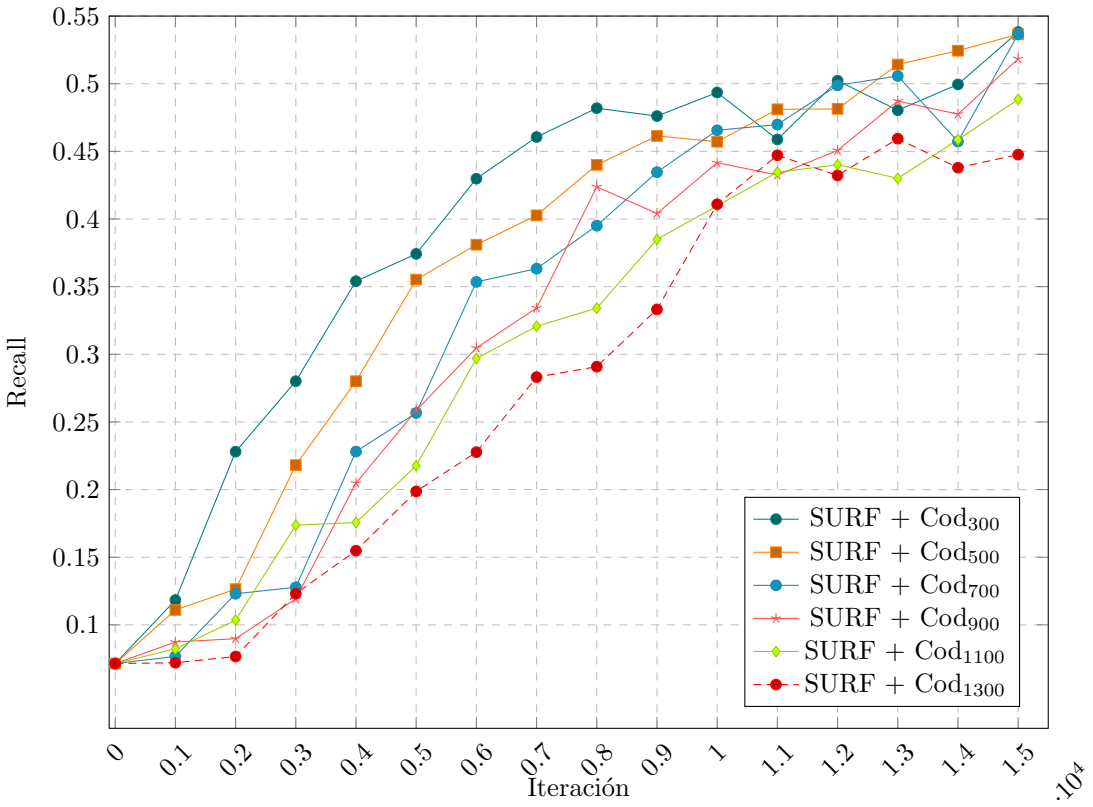
Fuente: *Elaboración propia*

Tabla 57: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF, durante 15000 iteraciones.*

Recall: NN + SURF						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714
1000	0.1183	0.1110	0.0767	0.0873	0.0823	0.0720
2000	0.2280	0.1264	0.1230	0.0898	0.1035	0.0766
3000	0.2801	0.2181	0.1278	0.1192	0.1737	0.1231
4000	0.3540	0.2801	0.2281	0.2048	0.1757	0.1548
5000	0.3742	0.3552	0.2566	0.2585	0.2176	0.1987
6000	0.4298	0.3810	0.3535	0.3047	0.2969	0.2277
7000	0.4606	0.4027	0.3633	0.3342	0.3207	0.2831
8000	0.4819	0.4400	0.3951	0.4238	0.3341	0.2908
9000	0.4761	0.4614	0.4346	0.4041	0.3850	0.3332
10000	0.4935	0.4571	0.4656	0.4416	0.4097	0.4109
11000	0.4588	0.4811	0.4698	0.4325	0.4346	0.4471
12000	0.5022	0.4814	0.4988	0.4507	0.4401	0.4322
13000	0.4804	0.5142	0.5057	0.4872	0.4301	0.4594
14000	0.4995	0.5244	0.4573	0.4775	0.4586	0.4379
15000	**0.5382**	0.5365	0.5365	0.5184	0.4885	0.4476

Fuente: *Elaboración propia*

Figura 85: *Resultados más elevados de Neural Network basado en la métrica Recall, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones.*



Fuente: *Elaboración propia*

De la Figura 82, se puede abstraer que la técnica de aprendizaje de máquina SVM presenta el mejor resultado equivalente a **64.26 %** al utilizar un kernel_{rbf}. En específico sobre un extractor de características SIFT, cuando el $\text{Cod}_{size} = 1300$. Mientras que en la Figura 83, cuando se analiza el extractor de características SURF la mejor técnica de aprendizaje vuelve a ser SVM. Sin embargo, utiliza un kernel_{rbf}, siendo el valor más elevado equivalente a **64.33 %**, utilizando un $\text{Cod}_{size} = 1100$. Por otro lado, las Figuras 84 y 85 obtienen los valores más elevados durante la iteración 15000; se puede observar en las Figuras 84 y 85 que los resultados mejoran a medida que el número de iteraciones crece, sobre un $\text{Cod}_{size} = 300$ e independiente del extractor de características utilizado.

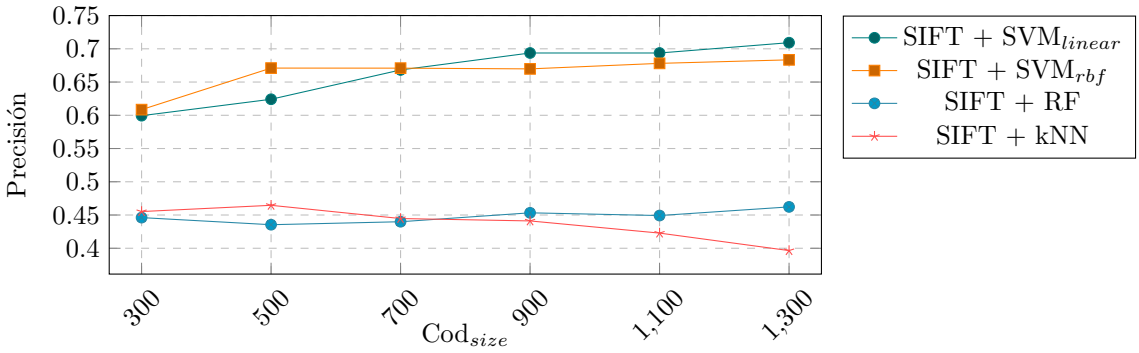
En la Tabla 58 se visualiza el resumen de los resultados de los Split Promedio para la *Precisión* a partir del modelo *Bag-of-Words*, basado en los extractores de características SIFT y SURF, utilizando las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor*. Así mismo, en las Tablas 59 y 60 se visualizan los resultados de aplicar Neural Network sobre los extractores de características SIFT y SURF respectivamente, durante 15000 iteraciones y utilizando diferentes tamaños del *codebook*. La visualización de estos resultados se muestran en las Figuras 86, 87, 88 y 89.

Tabla 58: *Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre los extractores de características SIFT y SURF.*

Precisión: SIFT				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5994	0.6085	0.4461	0.4552
500	0.6241	0.6710	0.4354	0.4646
700	0.6682	0.6709	0.4399	0.4447
900	0.6937	0.6698	0.4533	0.4411
1100	0.6937	0.6781	0.4492	0.4228
1300	**0.7092**	0.6834	0.4622	0.3965
Precisión: SURF				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5935	0.6215	0.4622	0.4653
500	0.6235	0.6649	0.4508	0.4586
700	0.6568	0.6595	0.4422	0.4646
900	0.6774	0.6620	0.4410	0.4454
1100	0.6870	**0.6910**	0.4492	0.4404
1300	0.6804	0.6800	0.4479	0.4569

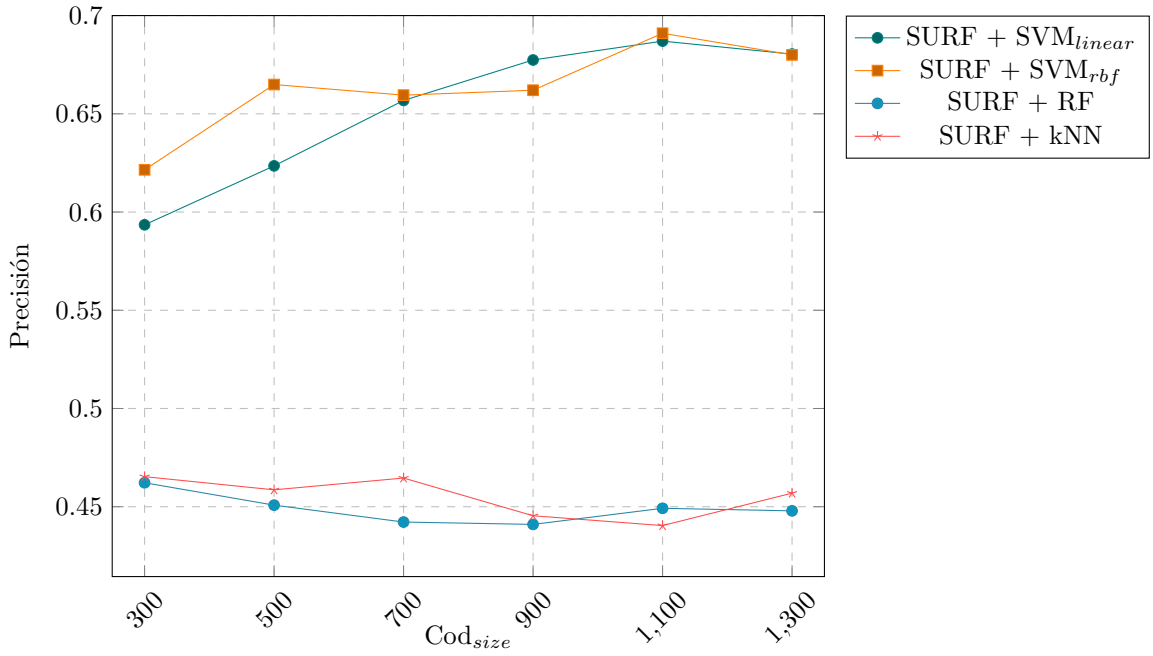
Fuente: *Elaboración propia*

Figura 86: *Resultados más elevados de SVM, RF y kNN basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT.*



Fuente: *Elaboración propia*

Figura 87: *Resultados más elevados de SVM, RF y kNN basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF.*



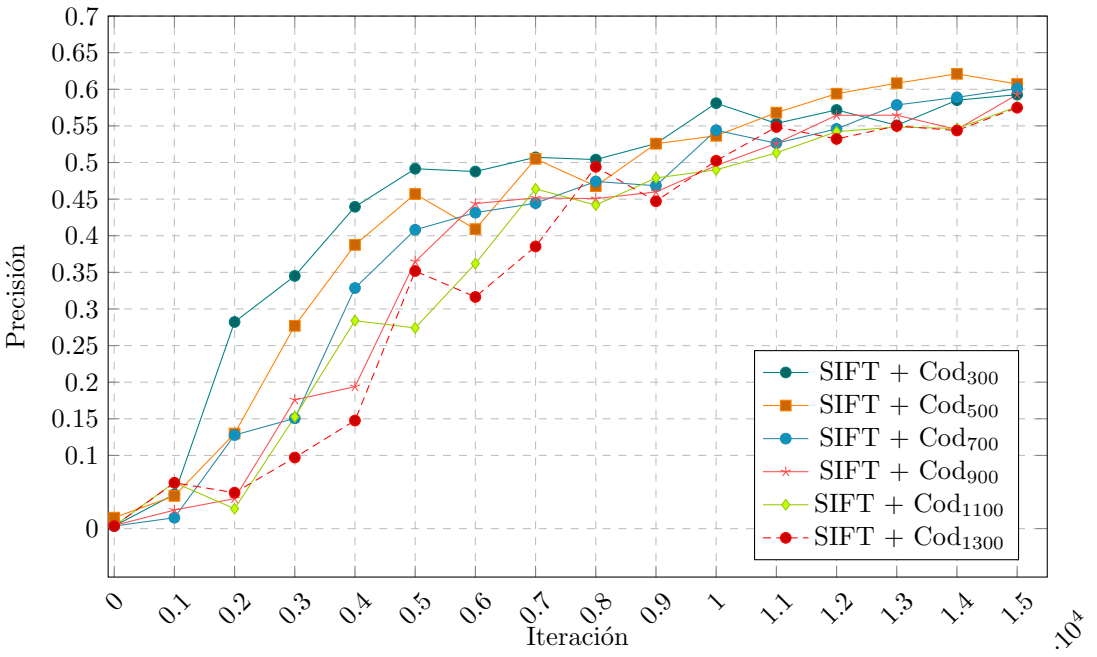
Fuente: *Elaboración propia*

Tabla 59: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT, durante 15000 iteraciones.*

Precisión: NN + SIFT						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0035	0.0150	0.0035	0.0043	0.0037	0.0036
1000	0.0469	0.0447	0.0149	0.0253	0.0630	0.0626
2000	0.2822	0.1299	0.1278	0.0410	0.0272	0.0491
3000	0.3451	0.2770	0.1506	0.1758	0.1527	0.0971
4000	0.4395	0.3875	0.3286	0.1939	0.2840	0.1476
5000	0.4915	0.4570	0.4081	0.3647	0.2741	0.3518
6000	0.4877	0.4090	0.4315	0.4439	0.3619	0.3164
7000	0.5071	0.5050	0.4444	0.4517	0.4640	0.3854
8000	0.5040	0.4677	0.4741	0.4507	0.4419	0.4941
9000	0.5259	0.5257	0.4683	0.4598	0.4789	0.4471
10000	0.5810	0.5365	0.5441	0.4958	0.4900	0.5024
11000	0.5534	0.5681	0.5263	0.5262	0.5132	0.5485
12000	0.5718	0.5940	0.5459	0.5645	0.5421	0.5322
13000	0.5508	0.6083	0.5786	0.5646	0.5485	0.5499
14000	0.5851	**0.6211**	0.5890	0.5451	0.5465	0.5437
15000	0.5927	0.6072	0.6010	0.5928	0.5762	0.5749

Fuente: *Elaboración propia*

Figura 88: *Resultados más elevados de Neural Network basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones.*



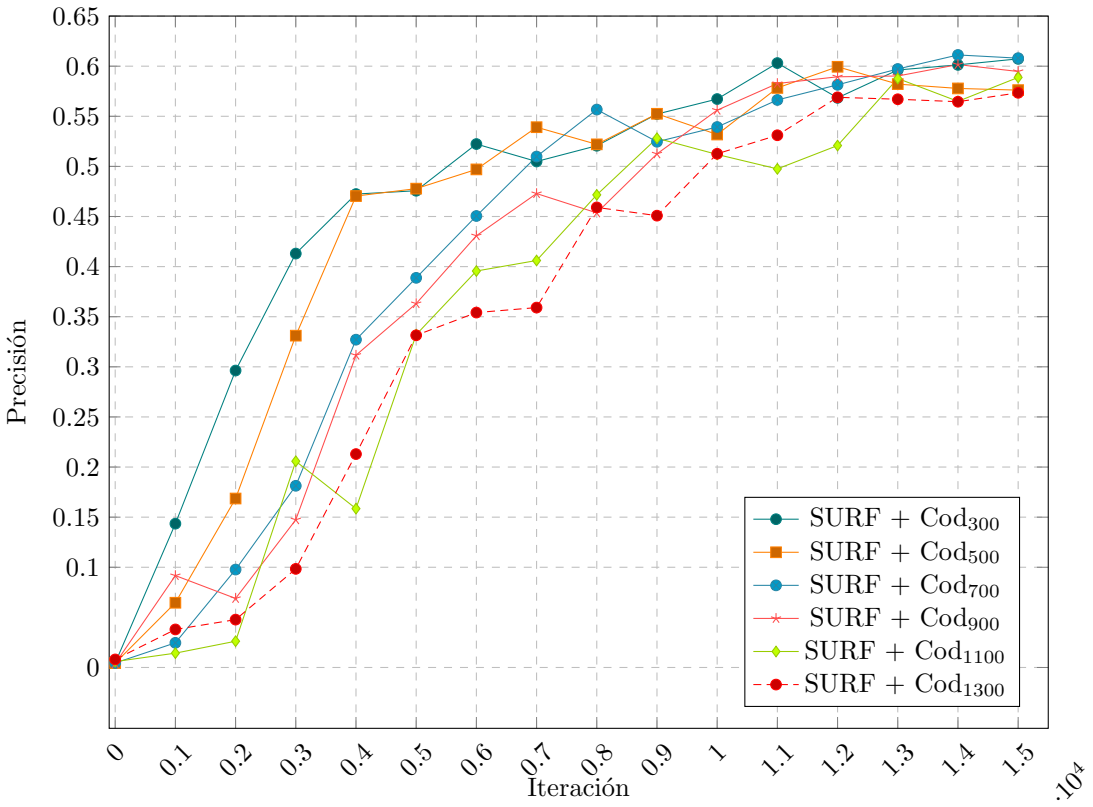
Fuente: *Elaboración propia*

Tabla 60: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF, durante 15000 iteraciones.*

Precisión: NN + SURF						
Iteración	**Cod_{300}**	**Cod_{500}**	**Cod_{700}**	**Cod_{900}**	**Cod_{1100}**	**Cod_{1300}**
0	0.0045	0.0044	0.0045	0.0039	0.0059	0.0080
1000	0.1435	0.0646	0.0246	0.0917	0.0143	0.0380
2000	0.2963	0.1686	0.0977	0.0689	0.0262	0.0478
3000	0.4130	0.3311	0.1813	0.1474	0.2058	0.0984
4000	0.4724	0.4704	0.3271	0.3116	0.1585	0.2129
5000	0.4757	0.4778	0.3888	0.3631	0.3321	0.3315
6000	0.5223	0.4970	0.4505	0.4308	0.3956	0.3541
7000	0.5049	0.5391	0.5098	0.4729	0.4061	0.3591
8000	0.5206	0.5220	0.5567	0.4536	0.4716	0.4591
9000	0.5522	0.5525	0.5246	0.5125	0.5281	0.4508
10000	0.5672	0.5318	0.5393	0.5561	0.5119	0.5126
11000	0.6031	0.5783	0.5663	0.5827	0.4975	0.5311
12000	0.5684	0.5994	0.5814	0.5894	0.5208	0.5689
13000	0.5962	0.5821	0.5972	0.5902	0.5877	0.5670
14000	0.6013	0.5778	**0.6112**	0.6016	0.5649	0.5645
15000	0.6073	0.5761	0.6080	0.5947	0.5888	0.5734

Fuente: *Elaboración propia*

Figura 89: *Resultados más elevados de Neural Network basado en la métrica Precisión, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones.*



Fuente: *Elaboración propia*

De la Figura 86, se puede abstraer que la técnica de aprendizaje de máquina SVM presenta el mejor resultado equivalente a **70.92 %** al utilizar un kernel_{linear}. En específico sobre un extractor de características SIFT, cuando el $\text{Cod}_{size} = 1300$. Mientras que en la Figura 87, cuando se analiza el extractor de características SURF, la mejor técnica de aprendizaje vuelve a ser SVM. Sin embargo, utiliza un kernel_{rbf}, siendo el valor más elevado equivalente a **69.10 %**, utilizando un $\text{Cod}_{size} = 1100$. Por otro lado, las Figuras 88 y 89 obtienen los valores más elevados durante la iteración 14000; se puede observar en las Figuras 88 y 89 que los resultados mejoran a medida que el número de iteraciones crece.

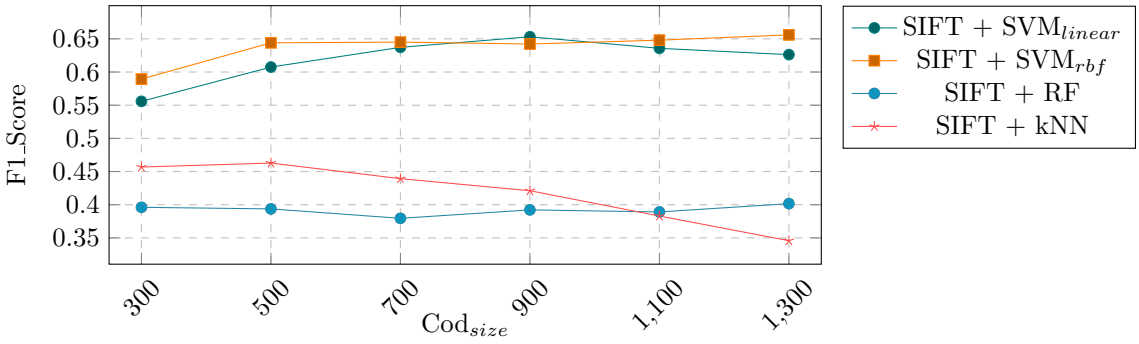
En la Tabla 61 se visualiza el resumen de los resultados de los Split Promedio para el *F1_Score* a partir del modelo *Bag-of-Words*, basado en los extractores de características SIFT y SURF, utilizando las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor*. Así mismo, en las Tablas 62 y 63 se visualizan los resultados de aplicar Neural Network sobre los extractores de características SIFT y SURF respectivamente, durante 15000 iteraciones y utilizando diferentes tamaños del *codebook*. La visualización de estos resultados se muestran en las Figuras 90, 91, 92 y 93.

Tabla 61: *Resumen de los resultados más elevados de las técnicas de aprendizaje de máquina SVM, RF y kNN basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre los extractores de características SIFT y SURF.*

F1_Score: SIFT				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5558	0.5894	0.3961	0.4568
500	0.6075	0.6440	0.3937	0.4628
700	0.6372	0.6451	0.3795	0.4393
900	0.6531	0.6422	0.3922	0.4211
1100	0.6357	0.6480	0.3891	0.3830
1300	0.6262	**0.6560**	0.4016	0.3458
F1_Score: SURF				
$\mathbf{Cod}_{size}$	SVM_{linear}	SVM_{rbf}	RF	kNN
300	0.5635	0.6004	0.4212	0.4602
500	0.6090	0.6419	0.4068	0.4473
700	0.6330	0.6415	0.3820	0.4284
900	0.6358	0.6369	0.3830	0.3778
1100	0.6287	**0.6598**	0.3871	0.3390
1300	0.5980	0.6517	0.3848	0.2897

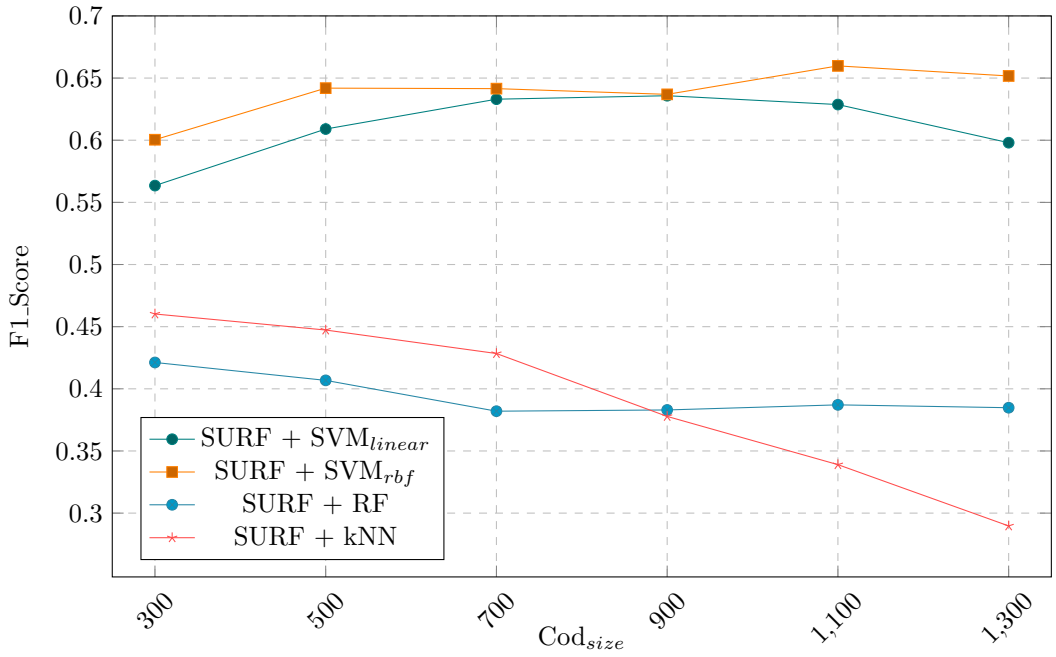
Fuente: *Elaboración propia*

Figura 90: *Resultados más elevados de SVM, RF y kNN basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT.*



Fuente: *Elaboración propia*

Figura 91: *Resultados más elevados de SVM, RF y kNN basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF.*



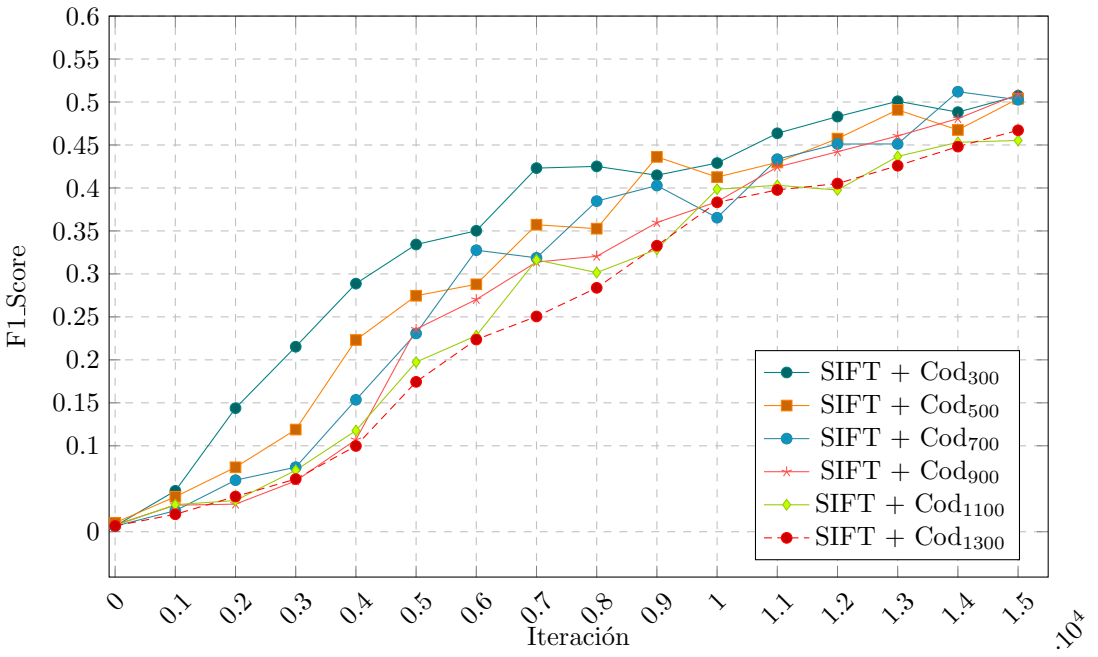
Fuente: *Elaboración propia*

Tabla 62: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook y SIFT, durante 15000 iteraciones.*

F1_Score: NN + SIFT						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0066	0.0104	0.0066	0.0081	0.0071	0.0068
1000	0.0476	0.0407	0.0245	0.0309	0.0315	0.0203
2000	0.1438	0.0750	0.0600	0.0321	0.0363	0.0410
3000	0.2153	0.1189	0.0751	0.0591	0.0715	0.0613
4000	0.2887	0.2231	0.1535	0.1071	0.1177	0.0998
5000	0.3342	0.2746	0.2307	0.2357	0.1974	0.1744
6000	0.3502	0.2879	0.3276	0.2703	0.2281	0.2236
7000	0.4231	0.3572	0.3187	0.3138	0.3163	0.2505
8000	0.4251	0.3526	0.3847	0.3205	0.3015	0.2838
9000	0.4148	0.4362	0.4028	0.3597	0.3286	0.3328
10000	0.4290	0.4126	0.3654	0.3840	0.3985	0.3833
11000	0.4636	0.4297	0.4335	0.4243	0.4031	0.3977
12000	0.4830	0.4574	0.4511	0.4421	0.3976	0.4051
13000	0.5009	0.4909	0.4512	0.4605	0.4367	0.4259
14000	0.4882	0.4675	**0.5120**	0.4807	0.4531	0.4481
15000	0.5073	0.5038	0.5026	0.5090	0.4553	0.4671

Fuente: *Elaboración propia*

Figura 92: *Resultados más elevados de Neural Network basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SIFT y un entrenamiento durante 15000 iteraciones.*



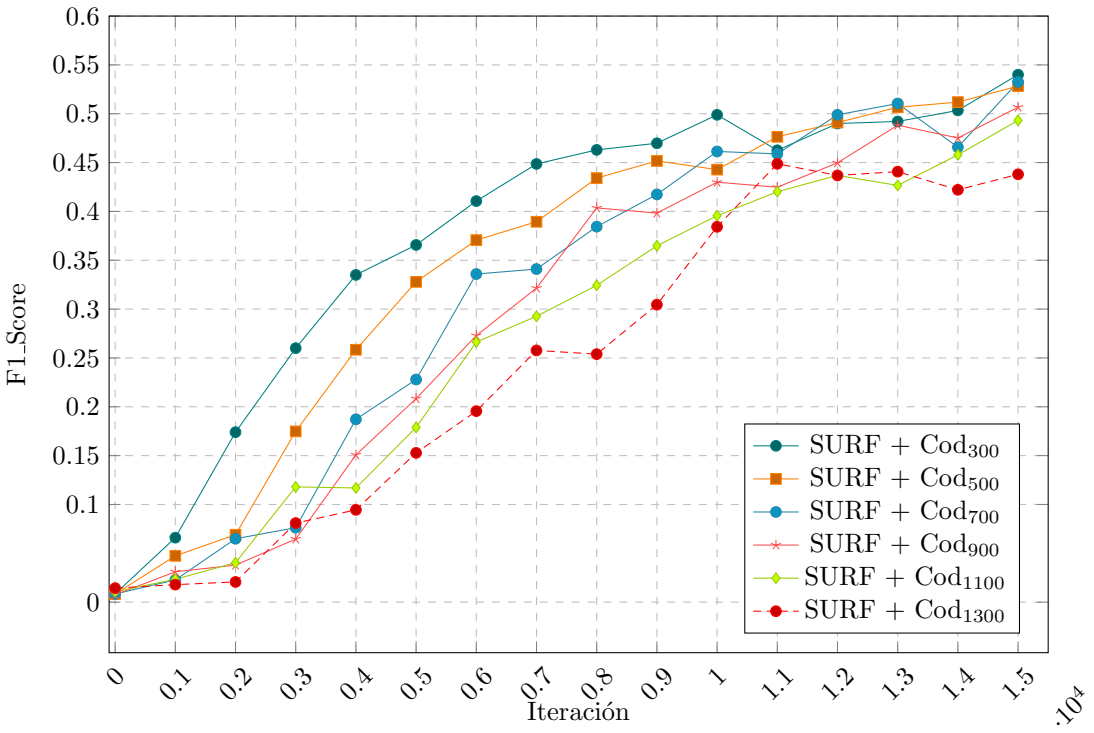
Fuente: *Elaboración propia*

Tabla 63: *Resumen de los resultados más elevados de la técnica de aprendizaje de máquina Neural Network basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook y SURF, durante 15000 iteraciones.*

F1_Score: NN + SURF						
Iteración	**Cod$_{300}$**	**Cod$_{500}$**	**Cod$_{700}$**	**Cod$_{900}$**	**Cod$_{1100}$**	**Cod$_{1300}$**
0	0.0085	0.0083	0.0084	0.0075	0.0107	0.0143
1000	0.0660	0.0474	0.0227	0.0311	0.0234	0.0179
2000	0.1740	0.0690	0.0649	0.0379	0.0404	0.0207
3000	0.2600	0.1748	0.0762	0.0646	0.1180	0.0809
4000	0.3350	0.2584	0.1872	0.1507	0.1169	0.0945
5000	0.3657	0.3279	0.2279	0.2084	0.1789	0.1528
6000	0.4106	0.3706	0.3359	0.2729	0.2663	0.1955
7000	0.4486	0.3894	0.3410	0.3215	0.2926	0.2577
8000	0.4630	0.4341	0.3844	0.4036	0.3243	0.2539
9000	0.4698	0.4517	0.4174	0.3983	0.3648	0.3045
10000	0.4989	0.4428	0.4614	0.4299	0.3956	0.3843
11000	0.4626	0.4764	0.4589	0.4248	0.4202	0.4487
12000	0.4900	0.4910	0.4989	0.4497	0.4369	0.4369
13000	0.4921	0.5066	0.5105	0.4883	0.4266	0.4407
14000	0.5034	0.5119	0.4658	0.4752	0.4579	0.4221
15000	**0.5400**	0.5283	0.5325	0.5067	0.4932	0.4380

Fuente: *Elaboración propia*

Figura 93: *Resultados más elevados de Neural Network basado en la métrica F1_Score, al utilizar Bag-of-Words con diferentes tamaños del codebook sobre el extractor de características SURF y un entrenamiento durante 15000 iteraciones.*



Fuente: *Elaboración propia*

De la Figura 90, se puede abstraer que la técnica de aprendizaje de máquina SVM presenta el mejor resultado equivalente a **65.60 %** al utilizar un kernel$_{rbf}$. En específico sobre un extractor de características SIFT, cuando el Cod$_{size}$ = 1300. Mientras que en la Figura 91, cuando se analiza el extractor de características SURF, la mejor técnica de aprendizaje vuelve a ser SVM, utilizando un kernel$_{rbf}$, siendo el valor más elevado equivalente a **65.98 %**, utilizando un Cod$_{size}$ = 1100. Por otro lado, las Figuras 92 y 93 obtienen los valores más elevados durante las últimas iteraciones.

De las Tabla 52 se puede observa que el mejor resultado es **66.25 %**, donde SIFT genera los mejores resultados de la métrica *Accuracy*, con un Cod$_{size}$ = 900. Sin embargo, al ser evaluada la métrica *Recall* se presenta un escenario diferente, como se muestra en las Tabla 55 donde el valor más elevado es **64.33 %**, se observa que SURF genera los mejores resultados utilizando un Cod$_{size}$ = 1100. Mientras tanto, al verificar la *Precisión* se contempla que SIFT posee un mejor rendimiento con un Cod$_{size}$ = 1300, como se muestra en la Tabla 58 donde **70.92 %** es el valor más elevados. Finalmente, de la Tabla 61 se observa que el mejor valor es **65.98 %** para la métrica F1_Score, se utiliza SURF y un Cod$_{size}$ = 1100.

4.2.2.2. *Redes Neuronales Convolucionales (CNN)*

Para evaluar los efectos de esta técnica, se comparan tres modelos pre-entrenados de redes neuronales convolucionales, como son: *VGG-16, VGG-19 e Inception-V3.* En específico, se utilizan los conceptos de *Transfer Learning* para extraer los *Transfer Values* de los modelos pre-entrenados. Estos se utilizan como entrada para cada uno de los cuatro técnicas de aprendizaje de máquina.

1. ***VGG-16:*** Los resultados de aplicar las técnicas de aprendizaje de máquina (ML) *Support Vector Machine, Random Forest y k Nearest Neighbor* sobre las carpetas Split 1, Split 2 y Split 3 se muestran en la Tabla 64. Donde se utiliza solo la técnica SVM con un kernel_{linear} por limitaciones computacionales, siendo los valores para C equivalentes a 100, 0.1 y 0.1 para las carpetas Split 1, Split 2 y Split 3 respectivamente. Mientras que en la Tabla 65 se visualiza el promedio de las carpetas Split.

Tabla 64: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN.*

SPLIT 1				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-16 + SVM	**0.9048**	**0.9028**	**0.9032**	**0.9026**
VGG-16 + RF	0.8280	0.8059	0.8662	0.8256
VGG-16 + kNN	0.7969	0.7929	0.8178	0.8013
SPLIT 2				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-16 + SVM	**0.9127**	**0.9189**	**0.9157**	**0.9165**
VGG-16 + RF	0.8201	0.8057	0.8708	0.8230
VGG-16 + kNN	0.7930	0.7945	0.8077	0.7956
SPLIT 3				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-16 + SVM	**0.9171**	**0.9171**	**0.9185**	**0.9177**
VGG-16 + RF	0.8298	0.8137	0.8642	0.8308
VGG-16 + kNN	0.7956	0.7974	0.8079	0.7957

Fuente: *Elaboración propia*

Tabla 65: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN.*

SPLIT PROMEDIO								
ML Technique	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
VGG-16 + SVM	**0.9115**	0.0062	**0.9129**	0.0088	**0.9125**	0.0081	**0.9123**	0.0084
VGG-16 + RF	0.8260	0.0052	0.8084	0.0046	0.8671	0.0034	0.8264	0.0040
VGG-16 + kNN	0.7952	0.0020	0.7949	0.0023	0.8111	0.0058	0.7975	0.0033

Fuente: *Elaboración propia*

De la Tabla 65 se observa que la técnica SVM presenta los mejores resultados para las métricas de *Accuracy*, *Recall*, *Precisión* y *F1_Score* equivalentes a **91.15 %, 91.29 %, 91.25 % y 91.23 %** respectivamente.

Tabla 66: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando Neural Network como técnica de aprendizaje de máquina.*

SPLIT 1: VGG-16 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0421	0.0447	0.0388	0.0340
1000	0.8675	0.8669	0.8630	0.8617
2000	0.8522	0.8503	0.8449	0.8422
3000	0.8719	0.8593	0.8770	0.8626
4000	0.8719	0.8711	0.8795	0.8733
5000	0.8807	0.8815	0.8757	0.8754
6000	0.8754	0.8752	0.8860	0.8777
7000	0.8654	0.8709	0.8749	0.8696
8000	0.8732	0.8831	0.8689	0.8726
9000	**0.8899**	0.8854	**0.8934**	0.8877
10000	0.8750	0.8701	0.8909	0.8767
11000	0.8680	0.8709	0.8668	0.8640
12000	0.8197	0.8103	0.8798	0.8298
13000	0.8627	0.8582	0.8768	0.8582
14000	0.8895	**0.8926**	0.8856	**0.8878**
15000	0.8890	0.8919	0.8852	0.8871
SPLIT 2: VGG-16 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0895	0.0687	0.0910	0.0538
1000	0.8342	0.8403	0.8422	0.8327
2000	0.8434	0.8514	0.8638	0.8518
3000	0.8412	0.8372	0.8762	0.8460
4000	0.8759	0.8749	0.8818	0.8738
5000	0.8711	0.8758	0.8733	0.8705
6000	0.8785	0.8816	0.8858	0.8816
7000	0.8715	0.8686	0.8799	0.8691
8000	0.8614	0.8705	0.8745	0.8632
9000	**0.8961**	**0.9029**	**0.9051**	**0.9008**
10000	0.8921	0.8965	0.9048	0.8981
11000	0.8921	0.8965	0.9048	0.8981
12000	0.8921	0.8965	0.9048	0.8981
13000	0.8921	0.8965	0.9048	0.8981
14000	0.8921	0.8965	0.9043	0.8979
15000	0.8921	0.8965	0.9043	0.8979
SPLIT 3: VGG-16 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0632	0.0579	0.0679	0.0497
1000	0.8509	0.8357	0.8641	0.8469
2000	0.8592	0.8571	0.8633	0.8553
3000	0.8579	0.8633	0.8631	0.8554
4000	0.8675	0.8668	0.8643	0.8608
5000	0.8158	0.8308	0.8671	0.8388
6000	0.8697	0.8582	0.8844	0.8664
7000	0.8754	0.8701	0.8838	0.8736
8000	0.8820	0.8892	0.8860	0.8857
9000	0.8636	0.8480	0.8911	0.8596
10000	0.8798	0.8730	0.8862	0.8757
11000	**0.8886**	**0.8905**	0.8857	**0.8864**
12000	0.8561	0.8543	0.8600	0.8476
13000	0.8719	0.8664	0.8829	0.8714
14000	0.8781	0.8806	0.8733	0.8728
15000	0.8868	0.8831	**0.8927**	0.8852

Fuente: *Elaboración propia*

En la Tabla 66 se visualiza que los resultados de la carpeta Split 1 equivalentes a **88.99 %, 89.26 %, 89.34 %** y **88.78 %** corresponden a la *Accuracy, Recall, Precisión y F1_Score* respectivamente. Así mismo, la carpeta Split 2 obtiene valores equivalentes a **89.61 %, 90.29 %, 90.51 %** y **90.08 %** durante la iteración 9000. También, se puede ver que los valores más elevados para la *Accuracy, Recall, Precisión y F1_Score* en la carpeta Slit 3 corresponde a **88.86 %, 89.05 %, 89.27 %** y **88.64 %** respectivamente. Por otro lado, en la Tabla 67 se visualiza el promedio y desviación estandar de las carpetas Split.

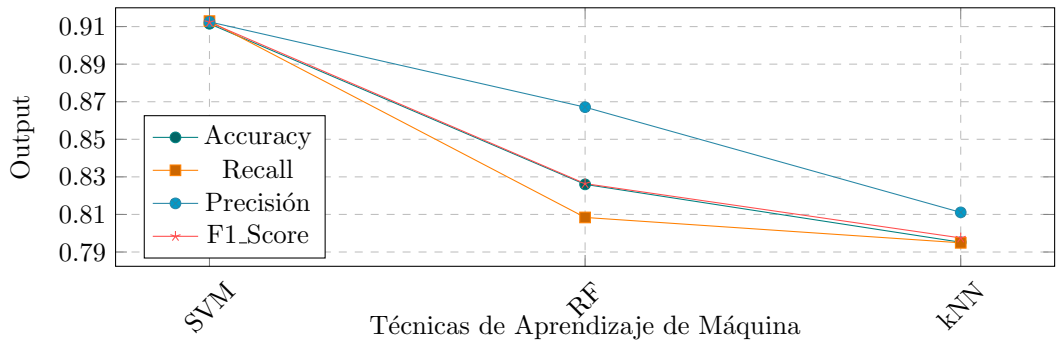
Tabla 67: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-16, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina Neural Network.*

SPLIT PROMEDIO								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0649	0.0237	0.0571	0.0120	0.0659	0.0262	0.0459	0.0104
1000	0.8509	0.0167	0.8476	0.0169	0.8564	0.0123	0.8471	0.0145
2000	0.8516	0.0079	0.8529	0.0036	0.8574	0.0108	0.8498	0.0068
3000	0.8570	0.0154	0.8533	0.0140	0.8721	0.0078	0.8547	0.0083
4000	0.8718	0.0042	0.8709	0.0041	0.8752	0.0095	0.8693	0.0073
5000	0.8558	0.0350	0.8627	0.0278	0.8720	0.0045	0.8616	0.0199
6000	0.8746	0.0045	0.8717	0.0121	0.8854	0.0009	0.8752	0.0079
7000	0.8708	0.0051	0.8699	0.0011	0.8795	0.0044	0.8708	0.0024
8000	0.8722	0.0103	0.8810	0.0096	0.8765	0.0088	0.8738	0.0113
9000	0.8832	0.0172	0.8788	0.0280	**0.8965**	0.0075	0.8827	0.0210
10000	0.8823	0.0088	0.8799	0.0145	0.8939	0.0097	0.8835	0.0127
11000	0.8829	0.0130	0.8860	0.0134	0.8858	0.0190	0.8828	0.0173
12000	0.8560	0.0362	0.8537	0.0431	0.8815	0.0224	0.8585	0.0354
13000	0.8756	0.0150	0.8737	0.0202	0.8882	0.0147	0.8759	0.0203
14000	0.8865	0.0075	0.8899	0.0083	0.8877	0.0156	0.8862	0.0126
15000	**0.8893**	0.0026	**0.8905**	0.0068	0.8941	0.0096	**0.8901**	0.0068

Fuente: *Elaboración propia*

En la Tabla 67 se visualiza que los mejores resultados para la **Accuracy, Recall, Precisión y F1_Score** corresponden a **88.93 %, 89.05 %, 89.65 % y 89.01 %** respectivamente. Por otro lado, en la Figura 94 se muestra que el mejor rendimiento corresponden a la técnica SVM.

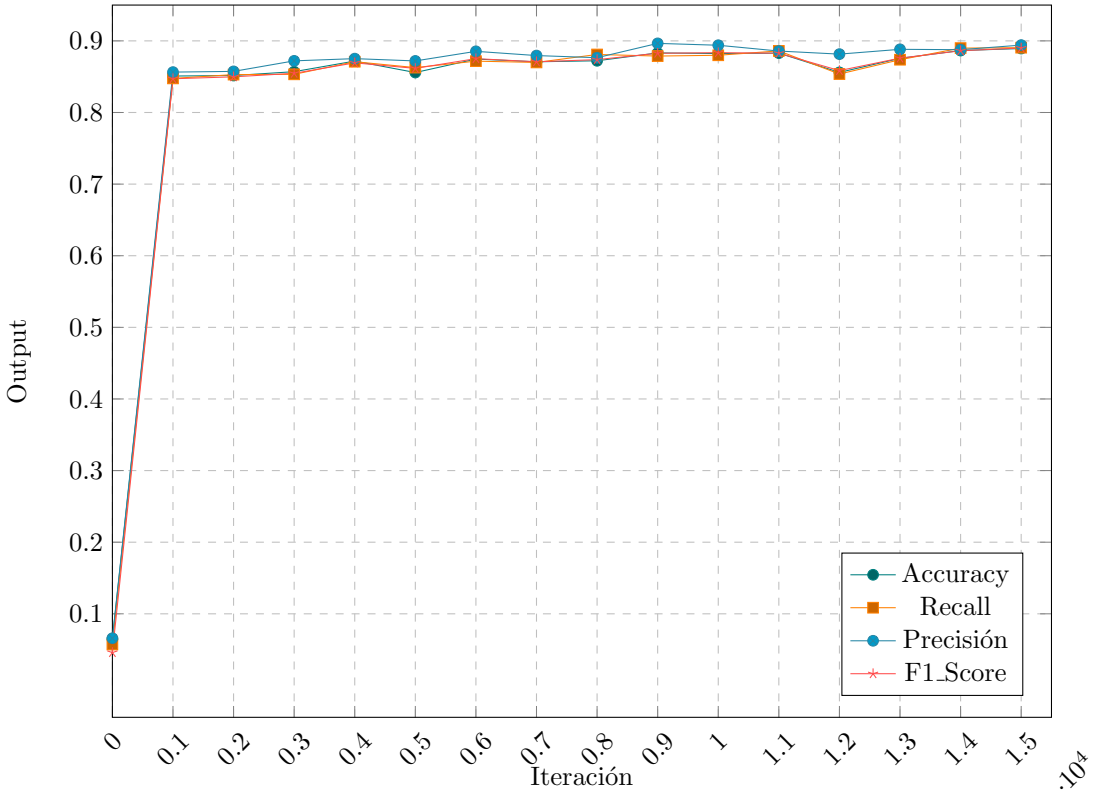
Figura 94: *Visualización de las meétricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo VGG-16. Utilizando SVM, RF y kNN.*



Fuente: *Elaboración propia*

Así mismo, en la Figura 95 se muestra que los resultados mejoran significativamente cada vez que el número de iteraciones aumenta; siendo esta la ventaja de utilizar un modelo de red neuronal pre-entrenado .

Figura 95: *Visualización de las métricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo VGG-16, basado en Neural Network.*



Fuente: *Elaboración propia*

2. ***VGG-19:*** De forma similar, los resultados de aplicar las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor* sobre las carpetas Split 1, Split 2 y Split 3 se muestran en la Tabla 68. Donde se utiliza solo la técnica SVM con un kernel$_{linear}$ por limitaciones computacionales, siendo los valores para C equivalentes a 0.1, 100 y 0.1 para las carpetas Split 1, Split 2 y Split 3 respectivamente. Mientras que en la Tabla 69 se visualiza el promedio de las carpetas Split.

   De la Tabla 69 se observa que la técnica SVM presenta los mejores resultados para las métricas *Accuracy*, *Recall*, *Precisión* y *F1_Score* que son equivalentes a **89.31 %, 89.50 %, 89.27 % y 89.34 %** respectivamente.

Tabla 68: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN*

SPLIT 1				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-19 + SVM	**0.8908**	**0.8901**	**0.8875**	**0.8883**
VGG-19 + RF	0.8105	0.7907	0.8379	0.8053
VGG-19 + kNN	0.7662	0.7588	0.7759	0.7615
SPLIT 2				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-19 + SVM	**0.8947**	**0.9014**	**0.9006**	**0.9006**
VGG-19 + RF	0.7899	0.7812	0.8408	0.7971
VGG-19 + kNN	0.7662	0.7668	0.7919	0.7719
SPLIT 3				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
VGG-19 + SVM	**0.8939**	**0.8935**	**0.8901**	**0.8912**
VGG-19 + RF	0.8118	0.7982	0.8515	0.8136
VGG-19 + kNN	0.7803	0.7821	0.7971	0.7819

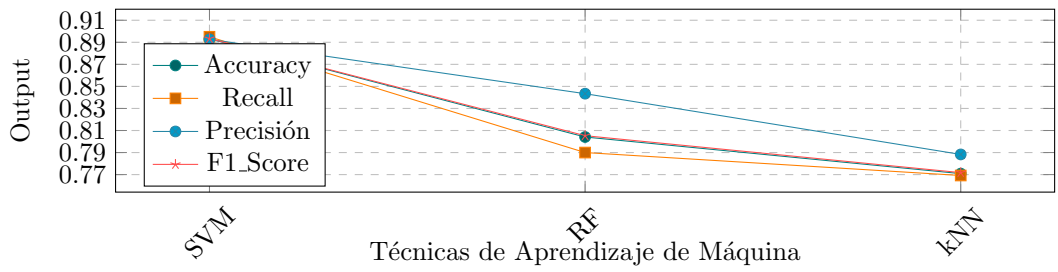
Fuente: *Elaboración propia*

Tabla 69: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN.*

SPLIT PROMEDIO								
ML Technique	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
VGG-19 + SVM	**0.8931**	0.0021	**0.8950**	0.0058	**0.8927**	0.0070	**0.8934**	0.0065
VGG-19 + RF	0.8041	0.0123	0.7900	0.0085	0.8434	0.0072	0.8053	0.0083
VGG-19 + kNN	0.7709	0.0081	0.7692	0.0118	0.7883	0.0111	0.7718	0.0102

Fuente: *Elaboración propia*

Figura 96: *Visualización de las métricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo VGG-19.*



Fuente: *Elaboración propia*

En la Figura 96 se muestra que el mejor rendimiento corresponden a la técnica SVM, para las métricas utilizadas.

Tabla 70: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando Neural Network como técnica de aprendizaje de máquina*

SPLIT 1: VGG-19 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0737	0.0755	0.1175	0.0554
1000	0.8368	0.8331	0.8531	0.8378
2000	0.8544	0.8489	0.8648	0.8515
3000	0.8404	0.8379	0.8576	0.8439
4000	0.8491	0.8383	0.8667	0.8469
5000	0.8434	0.8529	0.8425	0.8419
6000	0.8658	0.8683	0.8700	0.8661
7000	0.8592	0.8586	0.8601	0.8569
8000	0.8627	0.8618	0.8683	0.8623
9000	0.8518	0.8494	0.8591	0.8471
10000	0.8579	0.8612	0.8562	0.8555
11000	**0.8759**	**0.8792**	**0.8758**	**0.8748**
12000	0.8526	0.8583	0.8524	0.8519
13000	0.8605	0.8649	0.8639	0.8620
14000	0.8522	0.8518	0.8525	0.8468
15000	0.8649	0.8718	0.8621	0.8622
SPLIT 2: VGG-19 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1057	0.0914	0.0870	0.0720
1000	0.8294	0.8392	0.8327	0.8306
2000	0.8461	0.8514	0.8536	0.8476
3000	0.8491	0.8606	0.8548	0.8548
4000	0.8566	0.8583	0.8660	0.8608
5000	0.8373	0.8361	0.8557	0.8408
6000	**0.8711**	0.8750	**0.8763**	**0.8747**
7000	0.8504	0.8585	0.8562	0.8521
8000	0.8596	0.8616	0.8680	0.8625
9000	0.8667	0.8741	0.8696	0.8706
10000	0.8430	0.8420	0.8510	0.8369
11000	0.8500	0.8453	0.8675	0.8518
12000	0.8706	**0.8756**	0.8760	0.8736
13000	0.8548	0.8534	0.8758	0.8605
14000	0.8684	0.8728	0.8749	0.8708
15000	0.8684	0.8728	0.8749	0.8708
SPLIT 3: VGG-19 + NN				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0890	0.0709	0.0827	0.0468
1000	0.8346	0.8285	0.8497	0.8342
2000	0.8232	0.8373	0.8269	0.8248
3000	0.8404	0.8484	0.8364	0.8374
4000	0.8500	0.8626	0.8415	0.8490
5000	0.8373	0.8350	0.8558	0.8378
6000	0.8614	0.8588	0.8639	0.8585
7000	0.8596	0.8531	0.8646	0.8555
8000	0.8474	0.8404	0.8632	0.8424
9000	0.8544	0.8595	0.8475	0.8481
10000	**0.8754**	**0.8700**	**0.8802**	**0.8739**
11000	0.8588	0.8643	0.8579	0.8585
12000	0.8487	0.8515	0.8555	0.8489
13000	0.8588	0.8655	0.8653	0.8628
14000	0.8592	0.8657	0.8655	0.8631
15000	0.8596	0.8660	0.8659	0.8634

Fuente: *Elaboración propia*

En la Tabla 70 se visualiza que los resultados de la carpeta Split 1 equivalentes a **87.59 %, 87.92 %, 87.58 % y 87.48 %** corresponden a la *Accuracy, Recall, Precisión y F1_Score* respectivamente. Así mismo, la carpeta Split 2 obtiene valores equivalentes a **87.11 %, 87.50 %, 87.63 % y 87.47 %** durante la iteración 6000. También, se puede ver que los valores más elevados para la *Accuracy, Recall, Precisión y F1_Score* en la carpeta Slit 3 corresponde a **87.54 %, 87.00 %, 88.02 % y 87.39 %** respectivamente. Por otro lado, en la Tabla 71 se visualiza el promedio y desviación estandar de las carpetas Split.

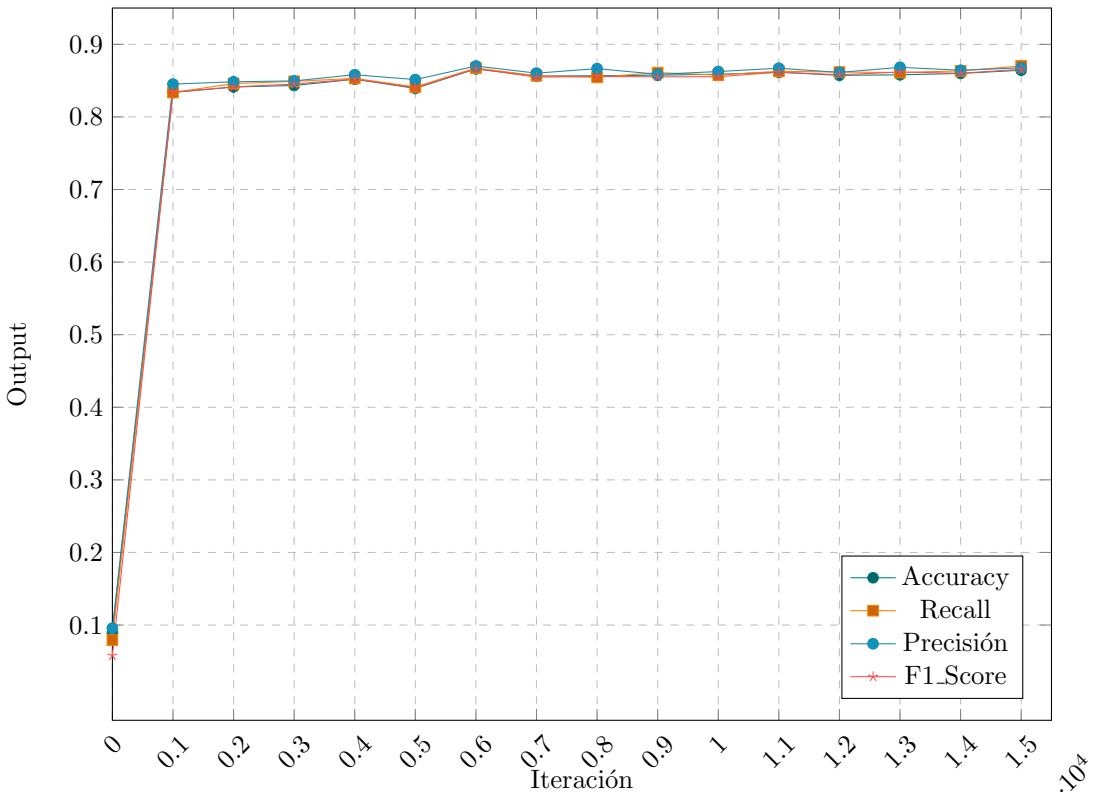
Tabla 71: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional VGG-19, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina Neural Network.*

SPLIT PROMEDIO								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0895	0.0160	0.0793	0.0108	0.0957	0.0190	0.0580	0.0128
1000	0.8336	0.0038	0.8336	0.0054	0.8452	0.0109	0.8342	0.0036
2000	0.8412	0.0161	0.8459	0.0075	0.8484	0.0195	0.8413	0.0144
3000	0.8433	0.0051	0.8490	0.0113	0.8496	0.0115	0.8453	0.0088
4000	0.8519	0.0041	0.8531	0.0129	0.8581	0.0144	0.8522	0.0075
5000	0.8393	0.0035	0.8413	0.0101	0.8514	0.0077	0.8402	0.0022
6000	**0.8661**	0.0048	0.8674	0.0081	**0.8701**	0.0062	**0.8664**	0.0081
7000	0.8564	0.0052	0.8567	0.0031	0.8603	0.0042	0.8548	0.0025
8000	0.8566	0.0081	0.8546	0.0123	0.8665	0.0029	0.8557	0.0116
9000	0.8576	0.0080	0.8610	0.0124	0.8587	0.0111	0.8553	0.0133
10000	0.8588	0.0162	0.8577	0.0143	0.8625	0.0156	0.8554	0.0185
11000	0.8615	0.0132	0.8629	0.0170	0.8671	0.0090	0.8617	0.0118
12000	0.8573	0.0117	0.8618	0.0124	0.8613	0.0128	0.8581	0.0135
13000	0.8580	0.0029	0.8612	0.0068	0.8683	0.0065	0.8618	0.0012
14000	0.8599	0.0081	0.8634	0.0107	0.8643	0.0112	0.8602	0.0123
15000	0.8643	0.0044	**0.8702**	0.0037	0.8676	0.0066	0.8655	0.0047

Fuente: *Elaboración propia*

En la Tabla 71 se visualiza que el promedio de los mejores resultados es equivalente a **86.61 %, 87.02 %, 87.01 % y 86.64 %** para las métrica de *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, en la Figura 97 se muestra que los resultados mejoran durante la primera iteración.

Figura 97: *Visualización de las métricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo VGG-19, basado en Neural Network.*



Fuente: *Elaboración propia*

3. ***Inception-V3:*** De igual forma, los resultados de aplicar las técnicas de aprendizaje de máquina *Support Vector Machine, Random Forest y k Nearest Neighbor* sobre las carpetas Split 1, Split 2 y Split 3 se muestran en la Tabla 72. Donde se utiliza solo la técnica SVM con un kernel$_{linear}$, siendo los valores para C equivalentes a 0.1, 100 y 0.1 para las carpetas Split 1, Split 2 y Split 3 respectivamente. Mientras que en la Tabla 73 se visualiza el promedio de las carpetas Split.

   De la Tabla 73 se observa que la técnica SVM presenta los mejores resultados para las métricas de *Accuracy*, *Recall*, *Precisión* y *F1_Score* equivalentes a **91.83 %, 91.79 %, 91.74 % y 91.72 %** respectivamente. Así mismo, en la Figura 98 se muestra que la técnica SVM presenta los mejores resultados.

Tabla 72: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN*

SPLIT 1				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
Inception-V3 + SVM	**0.9206**	**0.9186**	**0.9211**	**0.9192**
Inception-V3 + RF	0.8487	0.8310	0.8746	0.8436
Inception-V3 + kNN	0.8162	0.8207	0.8359	0.8257
SPLIT 2				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
Inception-V3 + SVM	**0.9140**	**0.9156**	**0.9149**	**0.9147**
Inception-V3 + RF	0.8118	0.8029	0.8525	0.8121
Inception-V3 + kNN	0.8048	0.8142	0.8283	0.8193
SPLIT 3				
ML Technique	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
Inception-V3 + SVM	**0.9202**	**0.9196**	**0.9162**	**0.9176**
Inception-V3 + RF	0.8377	0.8231	0.8671	0.8349
Inception-V3 + kNN	0.8048	0.8029	0.8215	0.8094

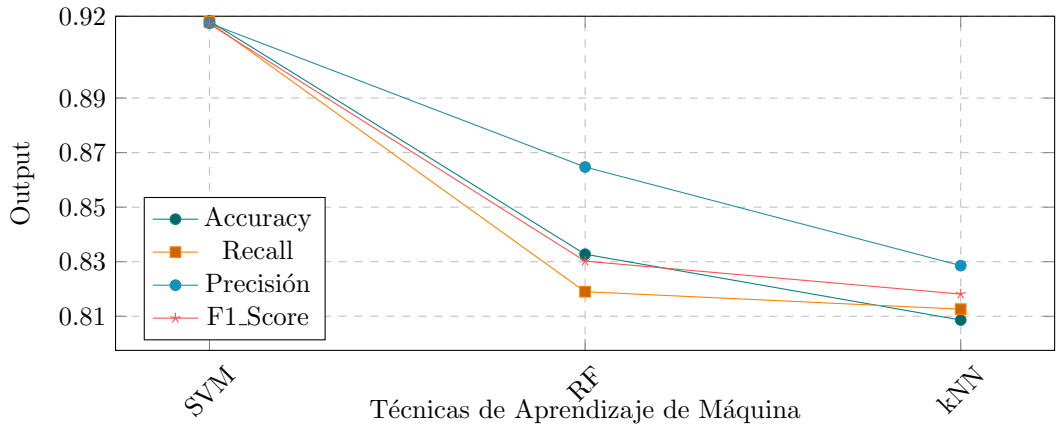
Fuente: *Elaboración propia*

Tabla 73: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando las técnicas de aprendizaje de máquina SVM, RF y kNN*

SPLIT PROMEDIO								
ML Technique	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
Inception-V3 + SVM	**0.9183**	0.0037	**0.9179**	0.0021	**0.9174**	0.0033	**0.9172**	0.0023
Inception-V3 + RF	0.8327	0.0189	0.8190	0.0145	0.8647	0.0112	0.8302	0.0163
Inception-V3 + kNN	0.8086	0.0066	0.8126	0.0090	0.8286	0.0072	0.8181	0.0082

Fuente: *Elaboración propia*

Figura 98: *Visualización de las métricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo Inception-V3.*



Fuente: *Elaboración propia*

Tabla 74: *Resumen de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando Neural Network como técnica de aprendizaje de máquina.*

SPLIT 1				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0390	0.0714	0.0028	0.0054
1000	0.8939	0.8913	0.8937	0.8891
2000	0.8991	0.8951	0.9076	0.8996
3000	0.9075	0.9042	0.9077	0.9045
4000	0.9057	0.9016	0.9094	0.9041
5000	0.9105	0.9098	0.9095	0.9087
6000	0.9096	0.9097	0.9067	0.9069
7000	0.9070	0.9045	0.9063	0.9042
8000	0.9079	0.9091	0.9022	0.9049
9000	0.9092	0.9109	0.9061	0.9077
10000	0.8890	0.8884	0.8964	0.8873
11000	0.9110	**0.9123**	0.9113	**0.9109**
12000	**0.9114**	0.9091	**0.9126**	0.9099
13000	0.9066	0.9032	0.9068	0.9039
14000	0.9070	0.9038	0.9071	0.9045
15000	0.8974	0.9025	0.8956	0.8970
SPLIT 2				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.0561	0.0433	0.1815	0.0307
1000	0.8706	0.8725	0.8967	0.8763
2000	0.8855	0.8858	0.8965	0.8888
3000	0.9022	0.9053	0.9053	0.9045
4000	0.9044	0.9082	0.9045	0.9054
5000	0.9057	0.9099	0.9036	0.9060
6000	0.9053	0.9089	0.9058	0.9064
7000	0.9053	0.9104	0.9050	0.9066
8000	0.9092	0.9126	0.9083	0.9096
9000	0.9061	0.9103	0.9037	0.9062
10000	0.9057	0.9095	0.9058	0.9069
11000	0.9070	0.9102	0.9063	0.9075
12000	0.9088	0.9124	**0.9103**	**0.9107**
13000	0.9088	0.9121	0.9084	0.9094
14000	**0.9105**	**0.9131**	0.9098	0.9105
15000	0.9083	0.9125	0.9086	0.9096
SPLIT 3				
Iteración	**Accuracy**	**Recall**	**Precisión**	**F1_Score**
0	0.1667	0.1296	0.0333	0.0519
1000	0.8868	0.8845	0.8899	0.8860
2000	0.8947	0.8967	0.8904	0.8920
3000	0.8706	0.8834	0.8842	0.8774
4000	0.8969	0.8953	0.8956	0.8949
5000	0.9026	0.9003	0.9016	0.9002
6000	0.9013	0.8984	0.9004	0.8991
7000	0.9048	0.9027	0.9001	0.9007
8000	0.9018	0.8995	**0.9035**	0.9000
9000	0.9061	**0.9043**	0.9025	**0.9024**
10000	0.9039	0.9002	0.9032	0.9000
11000	0.9044	0.9018	0.8992	0.8996
12000	0.9044	0.9001	0.8993	0.8991
13000	**0.9061**	0.9019	0.9034	0.9018
14000	0.8654	0.8678	0.8987	0.8744
15000	0.8895	0.8898	0.8883	0.8831

Fuente: *Elaboración propia*

En la Tabla 74 se visualiza que los resultados de la carpeta Split 1 equivalentes a **91.14 %, 91.23 %, 91.26 % y 91.09 %** corresponden a la *Accuracy, Recall, Precisión y F1_Score* respectivamente. Así mismo, la carpeta Split 2 obtiene valores equivalentes a **91.05 %, 91.31 %, 91.03 % y 91.07 %** durante las iteracioesn 14000, 14000, 12000 y 12000 respectivamente. También, se puede ver que los valores más elevados para la *Accuracy, Recall, Precisión y F1_Score* en la carpeta Slit 3 corresponde a **90.61 %, 90.43 %, 90.35 % y 90.24 %** respectivamente. Por otro lado, en la Tabla 75 se visualiza el promedio y desviación estandar de las carpetas Split.

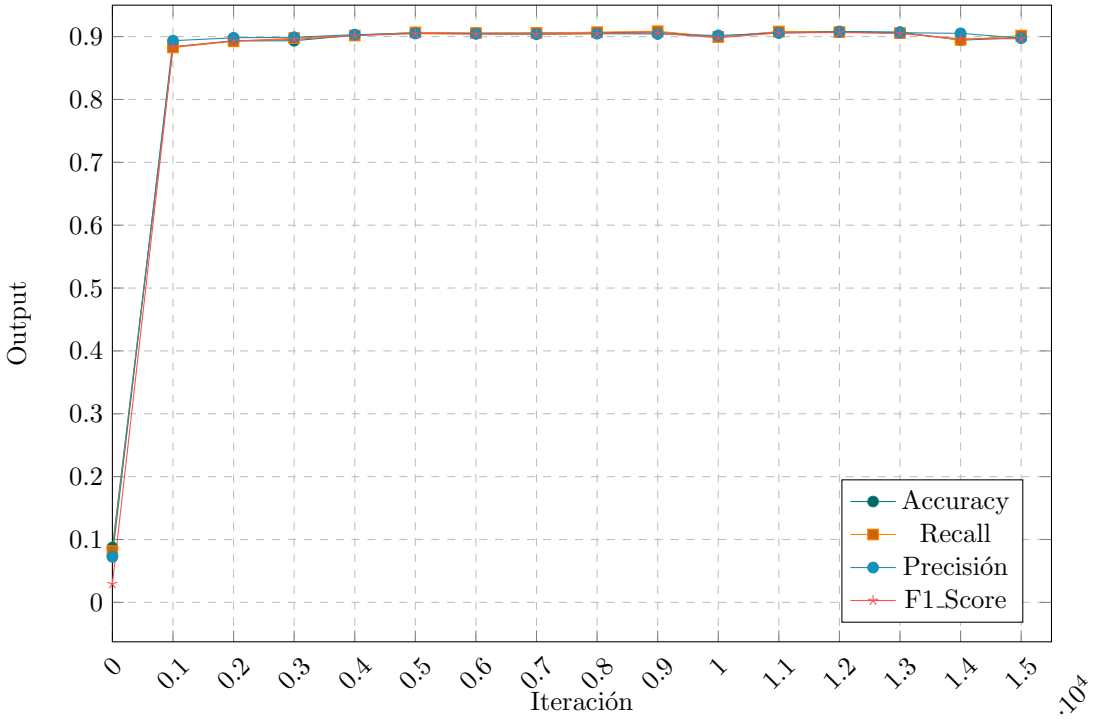
Tabla 75: *Promedio y desviación estandar de los resultados de la aplicación del modelo de red neuronal convolucional Inception-V3, sobre las carpetas Split. Utilizando la técnica de aprendizaje de máquina Neural Network.*

SPLIT PROMEDIO								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0873	0.0693	0.0814	0.0440	0.0725	0.0956	0.0293	0.0233
1000	0.8838	0.0119	0.8827	0.0095	0.8935	0.0034	0.8838	0.0066
2000	0.8931	0.0069	0.8925	0.0059	0.8981	0.0087	0.8934	0.0056
3000	0.8934	0.0199	0.8976	0.0124	0.8990	0.0129	0.8955	0.0156
4000	0.9023	0.0047	0.9017	0.0064	0.9031	0.0070	0.9014	0.0057
5000	0.9063	0.0040	0.9067	0.0055	0.9049	0.0041	0.9049	0.0043
6000	0.9054	0.0042	0.9057	0.0063	0.9043	0.0034	0.9041	0.0043
7000	0.9057	0.0012	0.9059	0.0040	0.9038	0.0033	0.9038	0.0030
8000	0.9063	0.0040	0.9070	0.0068	0.9046	0.0032	0.9048	0.0048
9000	0.9072	0.0018	0.9085	0.0037	0.9041	0.0018	0.9054	0.0027
10000	0.8996	0.0092	0.8994	0.0106	0.9018	0.0049	0.8981	0.0099
11000	0.9075	0.0033	**0.9081**	0.0055	0.9056	0.0061	0.9060	0.0058
12000	**0.9082**	0.0035	0.9072	0.0064	**0.9074**	0.0071	**0.9065**	0.0065
13000	0.9072	0.0014	0.9057	0.0056	0.9062	0.0026	0.9050	0.0039
14000	0.8943	0.0251	0.8949	0.0239	0.9052	0.0058	0.8965	0.0194
15000	0.8984	0.0095	0.9016	0.0114	0.8975	0.0103	0.8966	0.0133

Fuente: *Elaboración propia*

En la Tabla 75 se visualiza el promedio de las carpetas Split, donde los mejores resultados equivalen a **90.82 %, 90.81 %, 90.74 % y 90.65 %** para las métrica de aprendizaje *Accuracy*, *Recall*, *Precisión* y *F1_Score* respectivamente. Así mismo, en la Figura 99 se muestra que los resultados mejoran durante la primera iteración. Sin embargo, el modelo aprende lentamente durante cada iteración posterior.

Figura 99: *Visualización de las métricas de aprendizaje Accuracy, Recall, Precisión y F1_Score, obtenidas utilizando el modelo Inception-V3, basado en la técnica de aprendizaje de máquina Neural Network.*



Fuente: *Elaboración propia*

De la Tabla 73 se puede observar que los mejores resultados obtenidos durante la evaluación de las técnicas de aprendizaje de máquina son **91.83 %, 91.79 %, 91.74 % y 91.72 %** que corresponde a las métricas *Accuracy, Recall, Precisión y F1_Score* respectivamente. Estos resultados se logran a partir de la técnica de aprendizaje de máquina Support Vector Machine y el modelo de red neuronal convolucional pre-entrenado *Inception-V3*.

Así mismo, es necesario especificar que los modelos basados en *redes neuronales convolucionales* presentan una mejor extracción de características frente a los modelos basado en *Bag-of-Words*.

Por otro lado, se realizaron experimentos complementarios utilizando *data augmentation*, con el objetivo de visualizar las mejoras que ofrece esta técnica. Sin embargo, para simplificar los resultados solo se visualiza el promedio de las carpetas Split. En la Tabla 76 se visualizan los efectos de aplicar *data augmentation*, este proceso se aplica sobre cada imagen de entrada. En particular, se realizan transformaciones aleatorias de tinte, contraste, brillo y saturación, como se muestra en la Figura 100. Por lo tanto, el total de imágenes de entrenamiento sera igual a 11400. Es decir, el 50 % de la muestra para el entrenamiento = 2280 × 5 (4 transformaciones y una imagen original).

Figura 100: *Data augmentation aplicado sobre una imagen de entrenamiento.*



Fuente: *Elaboración propia*

Tabla 76: *Promedio y desviación estandar de los resultados de aplicar data augmentation sobre las carpetas Split. Utilizando el modelo pre-entrenado Inception-V3.*

SPLIT PROMEDIO: Support Vector Machine, Random Forest and k Nearest Neighbor								
ML Technique	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
SVM	**0.9235**	0.0063	**0.9252**	0.0028	**0.9232**	0.0079	**0.9239**	0.0054
RF	0.8405	0.0054	0.8252	0.0031	0.8751	0.0050	0.8389	0.0047
kNN	0.8142	0.0026	0.8177	0.0024	0.8306	0.0012	0.8219	0.0024
SPLIT PROMEDIO: Neural Network								
Iteración	U_{Acc}	σ_{Acc}	U_{Recall}	σ_{Recall}	U_{Pre}	σ_{Pre}	U_{F1}	σ_{F1}
0	0.0639	0.0522	0.0553	0.0194	0.0327	0.0070	0.0230	0.0052
1000	0.8461	0.0249	0.8439	0.0256	0.8766	0.0123	0.8427	0.0286
2000	0.8912	0.0086	0.8856	0.0147	0.9005	0.0058	0.8893	0.0118
3000	0.8968	0.0113	0.8952	0.0116	0.8973	0.0099	0.8932	0.0126
4000	**0.9042**	0.0071	0.9061	0.0049	**0.9097**	0.0061	**0.9054**	0.0065
5000	0.9015	0.0022	0.9055	0.0015	0.9001	0.0027	0.8999	0.0001
6000	0.8769	0.0114	0.8844	0.0102	0.8821	0.0108	0.8771	0.0115
7000	0.8969	0.0042	0.8978	0.0056	0.9019	0.0106	0.8964	0.0030
8000	0.8860	0.0102	0.8824	0.0088	0.8994	0.0055	0.8872	0.0074
9000	0.8918	0.0149	0.8911	0.0094	0.8971	0.0167	0.8901	0.0124
10000	0.9039	0.0047	**0.9063**	0.0030	0.9002	0.0080	0.9008	0.0063
11000	0.9022	0.0101	0.8977	0.0134	0.9063	0.0047	0.8997	0.0083
12000	0.8997	0.0051	0.8988	0.0080	0.8991	0.0064	0.8964	0.0062
13000	0.8972	0.0052	0.8961	0.0051	0.9042	0.0040	0.8972	0.0045
14000	0.8999	0.0058	0.8997	0.0144	0.9026	0.0053	0.8978	0.0073
15000	0.8918	0.0133	0.8974	0.0060	0.8939	0.0134	0.8919	0.0104

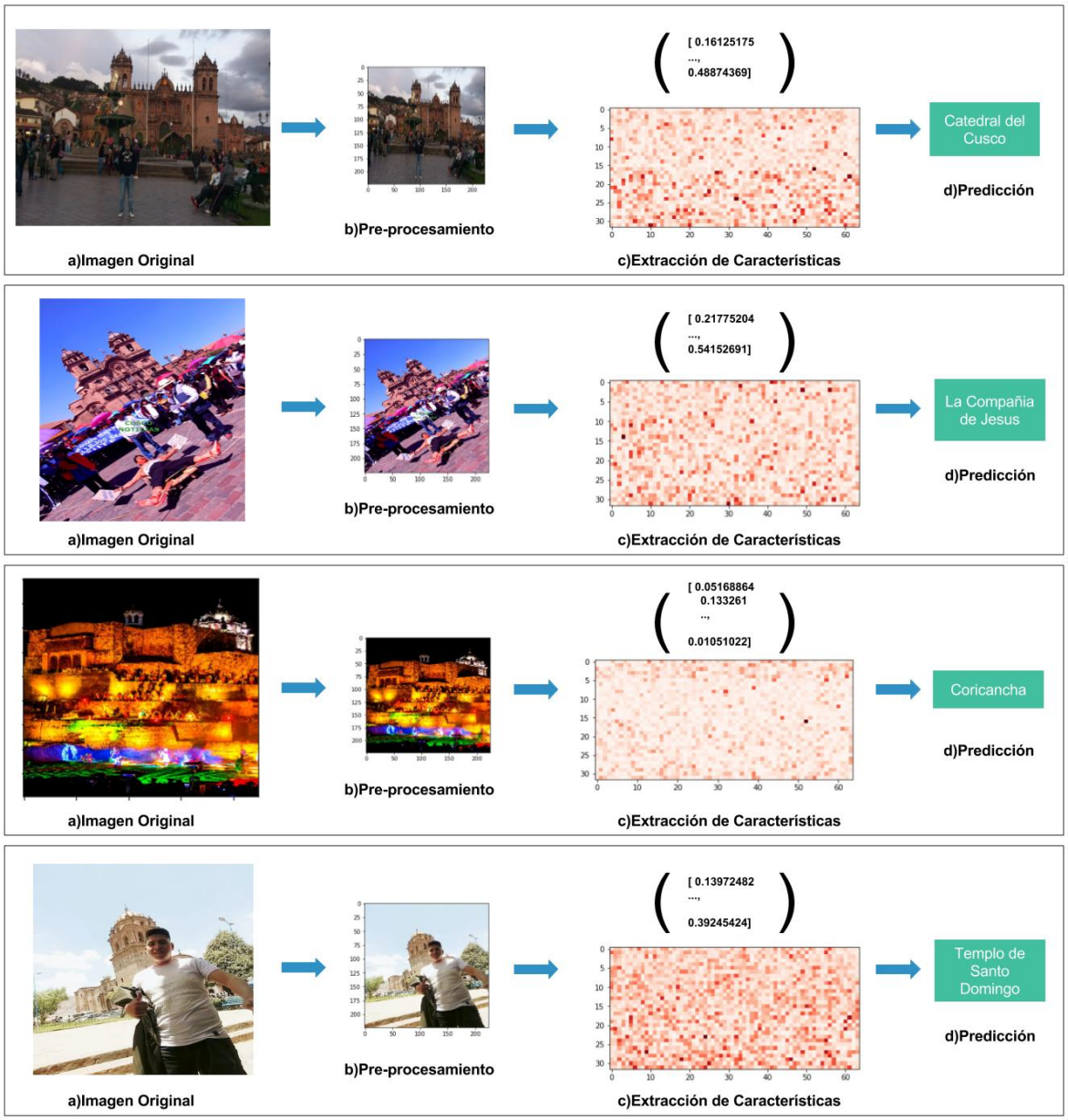
Fuente: *Elaboración propia*

4.2.3. Predicción

De la fase anterior (entrenamiento y construcción del modelo) se determino que el modelo de *red neuronal convolucional* basado en *Inception-V3* con *data augmentation* presenta los resultados más elevados a partir de las métricas utilizadas sobre la técnica de aprendizaje de máquina Support Vector Machine.

Se recolectaron 30 imágenes de *Instagram* para validar el modelo propuesto, donde la mayoría de imágenes pertenecen a la categoría Çatedral del Cusco". A continuación, en las Figura 101 y 102 se visualizan los casos de éxito y error del modelo propuesto.

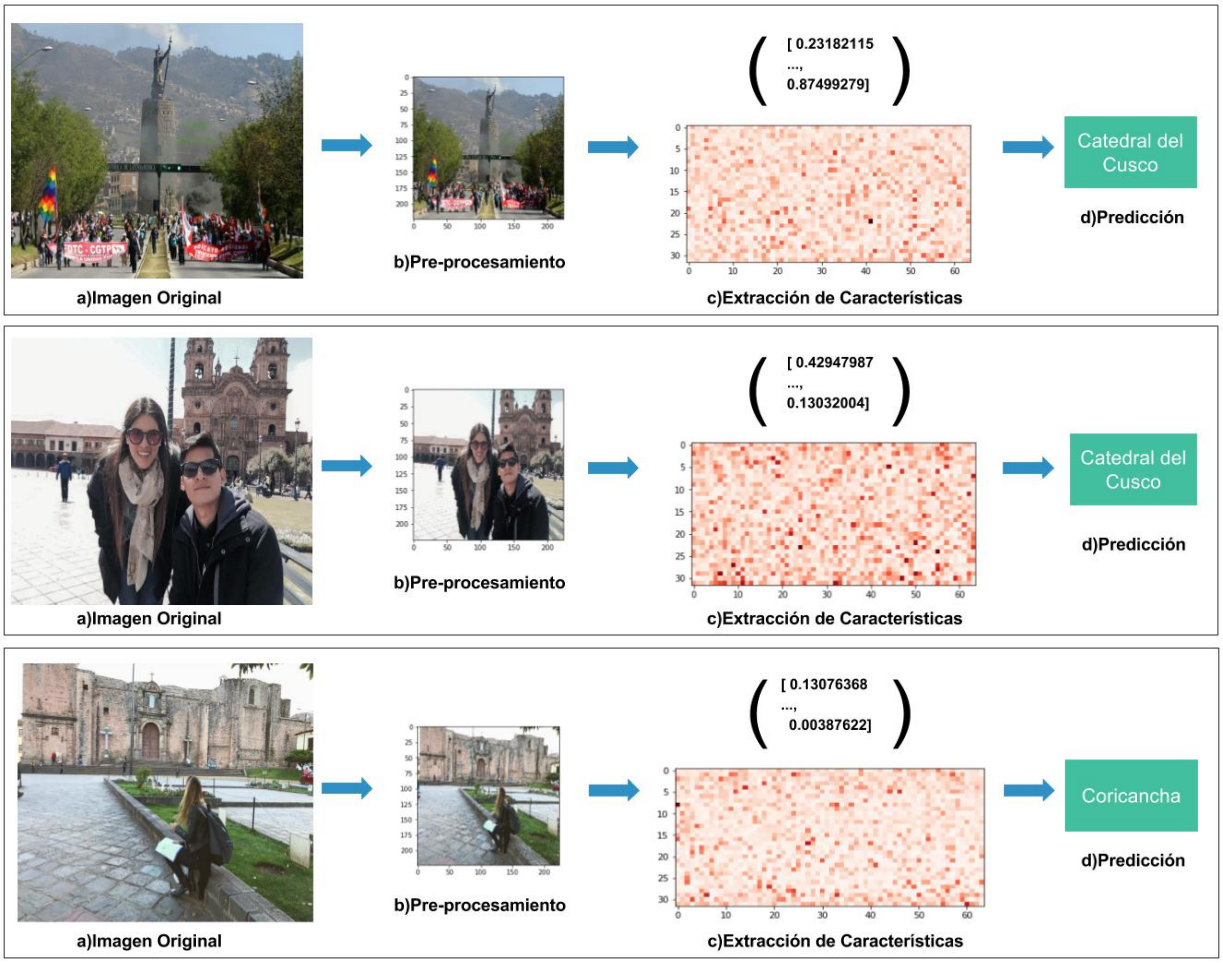
Figura 101: *Éxito de la predicción de un edificio histórico de la ciudad del Cusco.*



Fuente: *Elaboración propia*

El error durante la predicción de una imagen de un edificio histórico de la ciudad del Cusco se presenta a causa de; los *transfer values* generados por el modulo *Inception-V3* sobre la imagen de consulta, están próximos al hiperplano de separación de una clase que no corresponde. Debido a que, la imagen de consulta presenta ciertas características de dos edificio históricos de categorías diferentes.

Figura 102: *Error de la predicción de un edificio histórico de la ciudad del Cusco.*



Fuente: *Elaboración propia*

Del total de imágenes evaluadas, se identificaron 27 imágenes de forma correcta. Por otro lado, los casos de error se visualizan en la Figura 102. De este conjunto de datos, la *Accuracy* es equivalente a **90.00 %**.

Parte V

Resultados

Capítulo 5

Resultados e interpretaciones

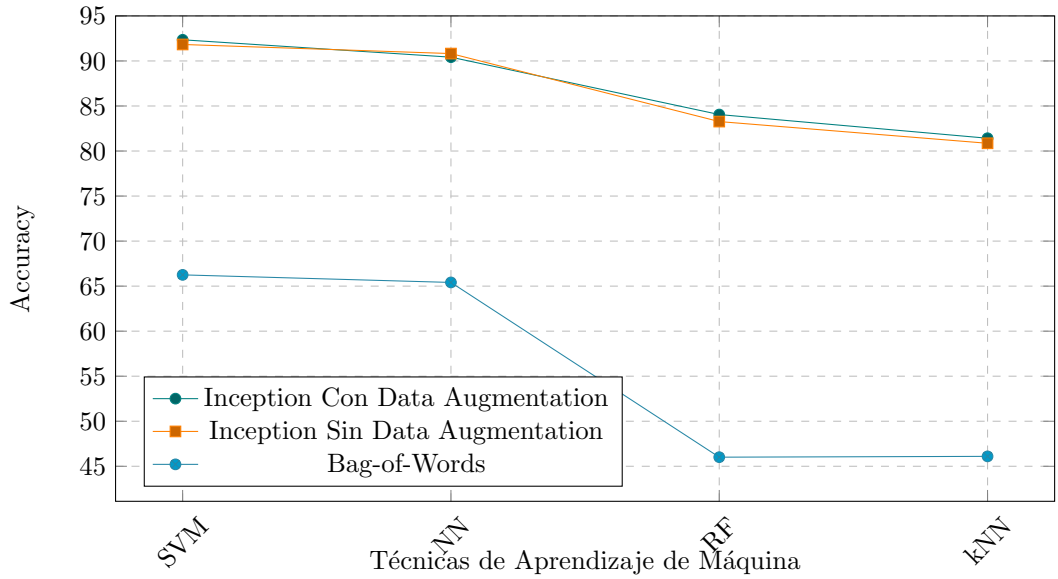
En la Tabla 77 se muestra la comparación de los mejores resultados del modelo de *red neuronal convolucional Inception-V3* (con y sin *data augmentation*) y *Bag-of-Words*. Así mismo, en las Figuras 103, 104, 105 y 106 se visualizan los resultados de las métricas *Accuracy, Recall, Precisión y F1_Score* respectivamente.

Tabla 77: *Comparación de los mejores resultados generados por las técnicas de aprendizaje de máquina Support Vector Machine, Random Forest, Neural Network y k Nearest Neighbor. Utilizando Inception-V3 (con y sin data augmentation) y Bag-of-Words.*

ACCURACY			
ML Technique	**Inception Con DA**	**Inception Sin DA**	**Bag-of-Words**
SVM	**92.35**	91.83	66.25
NN	90.42	90.82	65.41
RF	84.05	83.27	46.01
KNN	81.42	80.86	46.10
RECALL			
ML Technique	**Inception Con DA**	**Inception Sin DA**	**Bag-of-Words**
SVM	**92.52**	91.79	64.33
NN	90.63	90.65	53.82
RF	82.52	81.90	40.19
KNN	81.77	81.26	47.94
PRECISIÓN			
ML Technique	**Inception Con DA**	**Inception Sin DA**	**Bag-of-Words**
SVM	**92.32**	91.74	70.92
NN	90.97	90.81	62.11
RF	87.51	86.47	46.22
KNN	83.06	82.86	46.46
F1_SCORE			
ML Technique	**Inception Con DA**	**Inception Sin DA**	**Bag-of-Words**
SVM	**92.39**	91.72	65.98
NN	90.54	90.74	54.00
RF	83.89	83.02	42.12
KNN	82.19	81.81	46.28

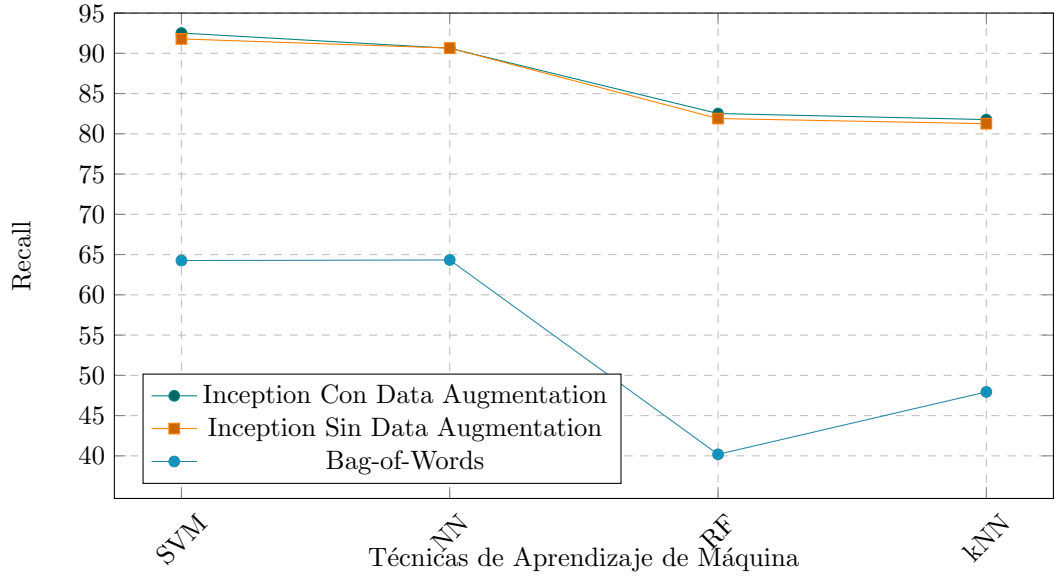
Fuente: *Elaboración propia*

Figura 103: *Comparación de los mejores resultados de la métrica Accuracy, Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words.*



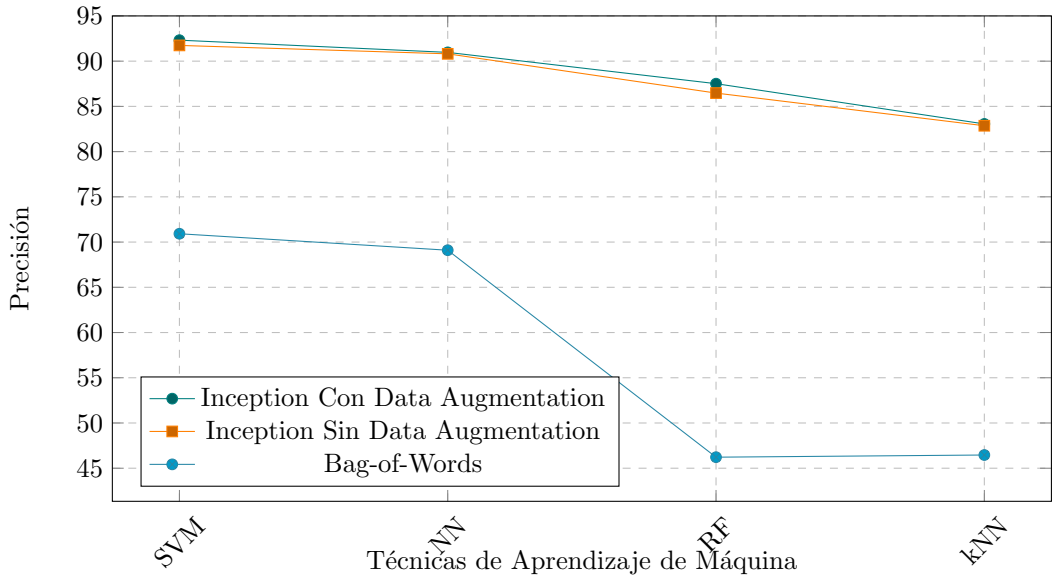
Fuente: *Elaboración propia*

Figura 104: *Comparación de mejores resultados de métrica Recall. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words.*



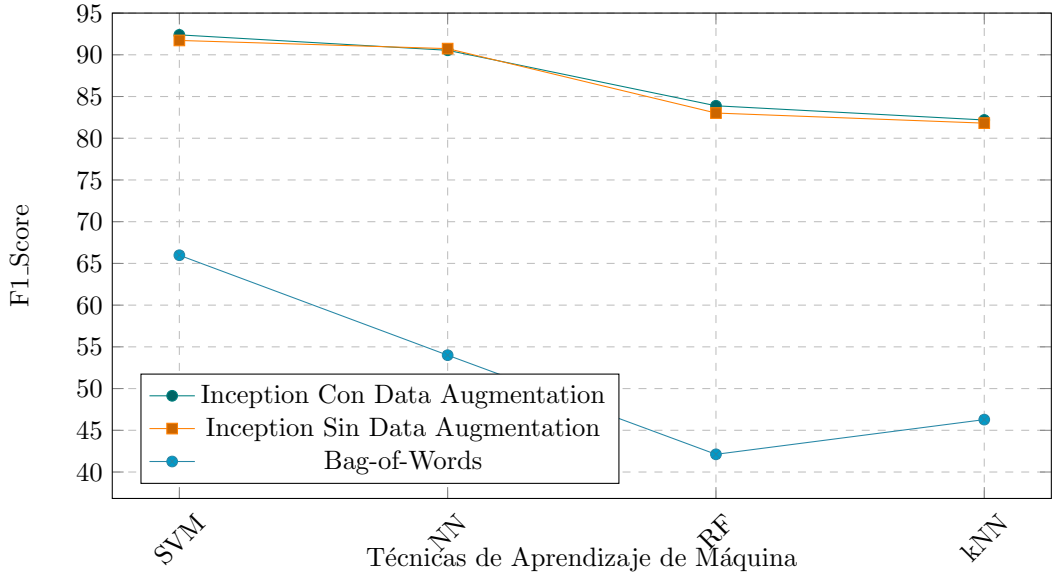
Fuente: *Elaboración propia*

Figura 105: *Comparación de mejores resultados de métrica Precisión. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words.*



Fuente: *Elaboración propia*

Figura 106: *Comparación de mejores resultados de métrica F1_Score. Utilizando el modelo pre-entrenado Inception-V3(con y sin data augmentation) y Bag-of-Words.*



Fuente: *Elaboración propia*

A partir de las Figuras 103, 104, 105 y 106 se puede observar que la técnica de aprendizaje de máquina *Support Vector Machine* obtiene los resultados más elevados en un escenario de identificación de edificios históricos de la ciudad del Cusco (ya sea utilizando *Bag-of-Words o redes neuronales convolucionales*), debido a que los parámetros del kernel se calculan de forma eficiente a partir de un proceso de búsqueda *(Grid Search)*. Así mismo, los hiperplanos de separación demuestran ser los más efectivos a la hora de separar categorías (clases). Por otro lado, la técnica *Neural Networks* presenta un número elevado de parámetros, como el número de capas, el número de neuronas por capa, la función de coste y otros; diversos trabajos como [Bashiri and Geranmayeh, 2011], hacen mención sobre el hecho de que no hay un método definido y explícito para seleccionar los parámetros óptimos para esta técnica. Mientras tanto, los parámetros de la técnica *Random Forest* se establecieron a partir de los valores seleccionados del trabajo previo [Biau, 2012], en este se consiguen buenos resultados utilizando un número de árboles equivalentes a 500 y una profundidad máxima de 50. Sin embargo, en este trabajo los resultados son pobres comparados con la técnica SVM. Finalmente, knn presenta los resultados más limitados. A pesar de que, la teoría indica que el parámetro más óptimo para k es equivalente a 1. Por lo tanto, knn es la técnica menos recomendable a la hora de identificar imágenes de edificios hitóricos. También, se espera que trabajos futuros realicen un *benchmarking* modificando el valor de estos parámetros, para elevar los resultados.

De la Tabla 77 los resultados más elevados de utilizar *Bag-of-Words* son generados a partir de la técnica de aprendizaje de máquina Support Vector Machine. Los resultados de las métricas *Accuracy y Precisión* son generados a partir de utilizar el extractor de características SIFT y un SVM con Kernel_{linear}. Por otro lado, los resultados de las métricas *Recall y F1_Score* se obtienen a partir de aplicar el extractor de características SURF y un SVM con Kernel_{rbf}, esto se debe a los siguientes factores:

- **Parámetros:** *Support Vector Machine* Por otro lado, el resto de técnicas de aprendizaje de máquina evaluados en este trabajo no presentan esa peculiaridad. Por ejemplo: *Neural Network (NN)*

- **Extractores de Características:** A partir de utilizar SIFT y SURF se desarrollo esta tarea, donde trabajos previos como [Panchal et al., 2013] y [Srivastava, 2015] identifican a SIFT como el extractor de características más invariante durante los cambios de escala, rotación e intensidad. Mientras tanto, SURF es identificado como el más robusto durante los cambios de iluminación. Sin embargo, según las métricas (*Accuracy*, *Recall*, *Precisión* y *F1_Score*) se visualiza que existe un equilibrio entre SIFT y SURF. Particularmente, estos resultados se dan por la variabilidad que presentan las imágenes del conjunto de datos analizado.

- ***Codebook*:** La variación del tamaño del *codebook* permite a la técnica SVM identificar los resultados más elevados en diferentes escenarios. Como se visualiza en los experimentos, donde existe la necesidad de variar el tamaño del *codebook* (Cod_{size}), debido a que en algunas métricas un Cod_{size} con valores elevado presenta resultados pobres. Mientras que en otros, un Cod_{size} pequeño presenta mejores resultados.

Por otro lado, los resultados más elevados de utilizar *redes neuronales convolucionales* (utilizando *data augmentation*) como extractor de características son a partir de la técnica Support Vector Machine, esto se debe a los siguientes factores:

- **Modelo pre-entrenado:** Actualmente, construir una arquitectura de red neuronal convolucional desde cero es un problema, debido a la inicialización de los parámetros y el alto costo computacional. Estos problemas son mitigados al utilizar modelos pre-entrenados. En particular, el modelo *Inception-V3* extrae características de manera adecuada, como se muestra en los resultados.

- **Profundidad de la CNN:** No basta con utilizar un modelo pre-entrenado, la teoría de *Deep Learning* indica que a mayor profundidad (Número de capas) el modelo extrae características más relevantes. En específico utilizar *Inception-V3* y sus 48 capas permite una mejor entrada para la técnica SVM, es necesario indicar que el número de imágenes utilizadas para el entrenamiento y prueba del clasificador fueron 2280.

- **Data augmentation:** Esta técnica eleva los resultados del modelo *Inception-V3*. Debido a que el número de imágenes de entrenamiento se incrementa. Es decir, el número total de muestra de entrenamiento es equivalente a $2280 \times 5 = 11400$ (4 transformaciones y una imagen original) y de esta forma se obtiene una fase de entrenamiento más óptima.

Finalmente, de los antecedentes (ver Tabla 1, Tabla 2 y Tabla 3) se puede visualizar que la precisión más elevada corresponde a 96 %. Mientras tanto, el resultado más elevado durante la fase de *Entrenamiento y construcción del modelo* para la métrica *Accuracy* corresponde a 92.35 %. Sin embargo, la diferencia de los resultados es debido a que, la base de datos evaluada no es la misma (*CuscoBID* presenta desafíos mayores). También, en este trabajo se analizan 14 categorías, mientras que en el otro trabajo se analizan 3. Por lo tanto, bajo estas consideraciones es aceptable decir que el modelo propuesto supera al antecedente con mejores resultados.

5.1. Detalles Técnicos

Para desarrollar el presente proyecto se utilizo el siguiente hardware y software.

5.1.1. Hardware

- **Cámara:** LUMIX DMC-SZ8.

- **Cámara:** Samsung Grand Neo Prime.

- **Notebook:** HP Pavilion, Procesador AMD A10-9600P RADEON R5, 16 Gb RAM.

5.1.2. Software

- **Sistema Operativo:** Ubuntu 16.04.1 LTS x86_64.
- **Lenguaje de Programación:** Python 2.7[37].
- **Librerías:**
  - Tensorflow 1.0.0: Se utilizo para realizar el entrenamiento de la *Neural Networks*. Es decir, se utilizo este framework para hacer uso de la función de coste de entropía cruzada y el aprendizaje basado en *Adam Optimization*. Por otro lado, también se utilizo durante la generación de imágenes durante la técnica *data augmentation*. Por último, se empleo el modelo pre-entrenado *inception-v3*[38].
  - SciKitLearn 0.18.1: Se utilizo para realizar el entrenamiento de las técnicas *Support Vector Machine, Random Forest* y *K Nearest Neighbor*. Además, para medir estas técnicas se emplearon las métricas de aprendizaje *Accuracy, Recall, Precisión* y *F1_Score*.
  - Keras 2.0.2: Se utilizo para obtener los modelos pre-entrenados VGG-16 y VGG-19[39].
  - Opencv 2.4.11: Framework utilizado durante la fase de detección y descripción de los puntos de interés (SIFT y SURF).
  - NumPy 1.12.0: Librería matemática utilizada para realizar operaciones entre matrices (imágenes y filtros).
  - SciPy 0.18.1: Librería utilizada para convertir una imagen a matriz. También, para realizar el redimensionamiento de la imagen.
  - Matplotlib 2.0.0: Se utilizo para la visualización de los datos, tales como el vector de características generado por *Inception-V3*

[37]El código fuente y la base de datos CuscoBID se encuentran en: `https://github.com/jeanfranc0/CuscoBID`

[38]El modelo pre-entrenado *Inception-V3* fue desarrollado en los laboratorios del MIT por [Hvass Pedersen, 2016]

[39]Los modelos pre-entrenados VGG-16 y VGG-19 fueron proporcionados por la librería especializada en Deep Learning KERAS [Chollet et al., 2015]

Conclusiones

1. Las conclusiones en base al objetivo general son:

   a) Los mejores resultados de la fase de *"Entrenamiento y construcción del modelo"* corresponden a la técnica de aprendizaje de máquina *Support Vector Machine*, donde se logra un tasa promedio de *Accuracy* de 91.83 %, un *Recall* de 91.79 %, una *Precisión* de 91.74 % y un *F1_Score* de 91.72 % al utilizar *redes neuronales convolucionales*. Así mismo, *Bag-of-Words* obtiene una *Accuracy, Recall, Precisión y F1_Score* equivalentes a 66.25 %, 64.26 %, 70.92 % y 65.98 % respectivamente. Por lo tanto, la técnica de aprendizaje de máquina *Support Vector Machine* basada en el modelo de *red neuronal convolucional* (en específico *Inception-V3*) es la más óptima al abordar el problema del reconocimiento de imágenes de edificios históricos de la ciudad del Cusco.

   b) Por otro lado, durante la fase de *"Predicción"* se valida la técnica de aprendizaje de máquina *Support Vector Machine* y el modelo de *red neuronal convolucional* (Inception-V3) sobre 30 imágenes recolectadas de *Instagram*. De estos experimentos se obtuvo una *Accuracy* equivalente a 90.00 %. Finalmente, sobre el modelo de *red neuronal convolucional Inception-V3* se aplico la técnica *data augmentation*, ésta demuestra tener efectos positivos debido a que en la fase de *"Entrenamiento y construcción del modelo"* obtiene los mejores resultados al utilizar la técnica de aprendizaje de máquina *Support Vector Machine*.

2. Las conclusiones en base al primer objetivo específico son:

   a) Se construyó la base de datos denominada *Cusco Building Database (CuscoBID)*, cuya última versión consta de 4560 imágenes de 14 diferentes edificios históricos de a ciudad del Cusco. Donde el 60 % corresponde a imágenes de internet para obtener diferentes condiciones fotométricas y el 40 % restante a imágenes con capturas personalizas que presentan problemas de rotación, ruido, escalamiento y otros.

3. Las conclusiones en base al segundo objetivo específico son:

   a) El resumen de evaluar un conjunto de técnicas de aprendizaje de máquina a partir de métodos basados en Bag-of-Words se muestra en las Tablas 52, 53 y 54 donde 66.25 %, 54.77 % y 55.94 % son los valores más elevados respectivamente, estos corresponden a la métrica *Accuracy*. Así mismo, los resultados generados para la métrica *Recall* se muestran en la Tablas 55, 56 y 57 con los valores más elevados equivalentes a

64.33 %, 51.01 % y 53.82 % respectivamente. De forma similar, los resultados generados para la métrica *Precisión* se muestran en las Tablas 58, 59 y 60 donde 70.92 %, 62.11 % y 61.12 % son los valores más elevados respectivamente. Finalmente, con respecto a la métrica *F1_Score* se visualiza en las Tablas 61, 62 y 63 que los valores más son equivalentes a 65.98 %, 51.20 % y 54.00 % respectivamente.

4. Las conclusiones en base al tercer objetivo específico son:

   a) El resumen de evaluar un conjunto de técnicas de aprendizaje de máquina sobre la arquitectura CNN *VGG-16* se muestra en las Tablas 65 y 67. Donde la técnica SVM presenta los mejores resultados(Tabla 65) para las métricas *Accuracy, Recall, Precisión y F1_Score* son equivalentes a 91.15 %, 91.29 %, 91.25 % y 91.23 % respectivamente. Así mismo, el resumen de los experimentos sobre la arquitectura *VGG-19* se muestra en las Tablas 69 y 71. Donde la técnica SVM presenta los mejores resultados(Tabla 69) para las métricas *Accuracy, Recall, Precisión y F1_Score* son equivalentes a 89.31 %, 89.50 %, 89.27 % y 89.34 % respectivamente. De forma similar, el resumen de los experimentos sobre la arquitectura *Inception-V3* se muestra en las Tablas 73 y 75. Donde la técnica SVM vuelve a presenta los mejores resultados(Tabla 73) para las métricas *Accuracy, Recall, Precisión y F1_Score* son equivalentes a 91.83 %, 91.79 %, 91.74 % y 91.72 % respectivamente. Finalmente, utilizando *data augmentation* sobre la arquitectura *Inception-V3* se observa (Tabla 76) que los mejores resultados para las métricas *Accuracy, Recall, Precisión y F1_Score* son equivalentes a 92.35 %, 92.52 %, 92.32 % y 92.39 % respectivamente.

5. Las conclusiones en base al cuarto objetivo específico son:

   a) De las Figuras 103, 104, 105 y 106 se comparan las técnicas de aprendizaje de máquina, a partir de los parámetros seleccionados para cada técnica. Donde SVM selecciona los parámetros para su kernel a partir de un proceso de búsqueda (*Grid Search*), este selecciona los parámetros de forma óptima, debido a que los hiperplanos de separación demuestran ser los más efectivos a la hora de separar categorías. Así mismo, la técnica *Neural Networks* utiliza 1024 neuronas en la capa de entrada, 1024 neuronas en la capa oculta y 14 neuronas en la capa de salida. A continuación, se utiliza un función de activación, en específico *Softmax* para generar la probabilidad de pertenencia de la imagen a todas las clases. Sin embargo, trabajos previos como [Bashiri and Geranmayeh, 2011], indican que no hay un método definido y explícito para seleccionar los parámetros óptimos para esta técnica. De forma similar, los parámetros seleccionados para *Random Forest* se obtiene a partir de [Biau, 2012]. Donde se utiliza un número de árboles equivalentes a 500 y una profundidad máxima de 50. Finalmente, knn presenta los resultados más limitados. A pesar de que, la teoría indica que el parámetro más óptimo para k es equivalente a 1. Por consiguiente, a partir de obtener parámetros óptimos, SVM presenta los resultados más elevados frente a las otras técnicas.

b) En la Tabla 77 se comparan los resultados más elevados de experimentar un conjunto de técnicas de aprendizaje de máquina a partir de métodos basados en *Bag-of-Words* y *redes neuronales convolucionales*; donde la arquitectura *Inception-V3* presenta los mejores resultados equivalentes a 91.83 %, 91.79 %, 91.74 % y 91.72 % para las métricas *Accuracy, Recall, Precisión y F1_Score* respectivamente. Por otro lado, *Bag-of-Words* obtiene los resultados más elevados para las métricas *Accuracy, Recall, Precisión y F1_Score* equivalentes a 66.25 %, 64.33 %, 70.92 % y 65.98 % respectivamente.

6. La conclusión en base al quinto objetivo específico es:

   a) De la Tabla 77 se observa que la técnica de aprendizaje de máquina más óptima es Support Vector Machine a partir de utilizar una arquitectura *Inception-V3* basada en la técnica *data augmentation*. Debido a que obtiene los resultados más elevados durante la evaluación de las métricas de aprendizaje.

Trabajos Futuros

- Se deja como trabajo futuro analizar otros escenarios de rechazo. Es decir, en un escenario real, las imágenes no siempre pertenecen a una de las categorías de la base de datos. Por lo tanto, se debe proponer métodos capaces de identificar que una imagen dada no pertenece a ninguna de las categorías definidas, en otras palabras, debe rechazar la imagen.

- Se deja como trabajo futuro evaluar otros modelos pre-entrenados como *ResNet, Xception, Inception-V4 o Inception-Resnet-v2* sobre un conjunto de datos con un mayor número de imágenes y categorías.

- El campo del *Deep Learning* esta en constate avance. Por lo tanto, se recomienda realizar experimentos utilizando arquitecturas basadas en *Densely Connected Convolutional Networks (DenseNet)*, este es un tipo de *red neuronal convolucional* muy similar a *ResNet*; en *ResNet* la entrada a la capa L_i se obtiene mediante la suma de las salidas de las capas anteriores. Mientras tanto, en *DenseNet* la entrada a la capa L_i se obtiene mediante la concatenación de las salidas de las capas anteriores, como lo describe [Huang et al., 2017]. Así mismo, la razón a utilizar esta arquitectura *deep learning* basada en *DenseNet*, es el número reducido de parámetros y el rendimiento elevado durante el desafío *ImageNet*.

- De forma simiar, otra forma de generar datos (*data augmentation*) es utilizar un enfoque que utilice *deep learning*. En particular, *Generative Adversarial Networks (GAN)* es una técnica con buenos resultados. Debido a que a partir de una cantidad limitada de imágenes, puede generar un número elevado de datos; para luego realizar un proceso de clasificación. Se recomienda utilizar esta técnica por el número limitado de datos a entrenar.

- Finalmente, En noviembre del 2017 se publico una nueva arquitectura *deep learning*. En particular, esta es denominada *Capsule Networks (CapsNet)* [Sabour et al., 2017]. El autor indica que "La operación de *pooling* utilizada en las *redes neuronales convolucionales* es un gran error y el hecho de que funcione tan bien es un desastre". Por lo tanto, existe la necesidad de realizar aplicaciones utilizando *CapsNet*, para luego efectuar comparaciones con los resultados obtenidos por las CNNs.

Apéndice A

Publicación

Towards accurate building recognition using convolutional neural networks

Jeanfranco D. Farfan-Escobedo*, Lauro Enciso-Rodas*, John E. Vargas-Muñoz†‡
*Escuela Profesional de Ingeniería Informática y de Sistemas
Universidad Nacional de San Antonio Abad del Cusco, Perú
Email: jeanfrancodfe@gmail.com, lauro.enciso@unsaac.edu.pe
†Institute of Computing - University of Campinas, Campinas, Brazil
Email: john.vargas@ic.unicamp.br
‡Asociación de Investigadores en Tecnologia del Cusco

***Abstract*—Building recognition from images is a challenging task since pictures can be taken from different angles and under different illumination conditions. Most of the building recognition methods use local and global handcrafted image features and do not consider the rejection scenario, where the method have to be capable of identifying if a given image does not belong to any of the classes of interest. We propose a method based on convolutional neural networks that obtain effective feature vectors to perform accurate classification of buildings. Additionally, we analyze and propose methods for the problem of classification with rejection.**

***Keywords*—building recognition, convolutional neural networks, classification with rejection**

I. Introduction

Building recognition methods can be used in various kinds of applications like architectural design identification and mobile device navigation [1]. The recognition of buildings from image data is a complex task since the pictures can be taken from different perspectives and under diverse illumination conditions. An additional challenge is to differentiate buildings with a similar architectural design.

Several methods in the literature [2], [3] use global and local image features, for example color, texture and SIFT (Scale-invariant feature transform [4]) features. Other methods use the bag-of-words technique (BOW) to obtain a middle-level representation of the image and afterward perform classification [5], [3]. The method proposed in [6] use a deep learning method for building recognition. However, the author uses a simple LeNet [7] neural network architecture since it requires fewer computational resources to train the model.

In this work, we propose a transfer learning approach using a recent state-of-the-art convolutional neural network (CNN), namely Inception-V3 [8], to accurately recognize buildings. We compare the proposed method with bag-of-words techniques as baselines. There exist two publicly available data sets for building recognition [9], [10]. However, the images in these data sets contain low variance of illumination conditions and the data set presented in [10] has a few images per category. Therefore, we use for validation a new data set containing pictures from historical buildings in the city of Cusco. This data set contains images taken from different perspectives and diverse photometric conditions, and several pictures contain occlusions.

In a real scenario, the images do not always belong to one of the building categories in a data set. So, the method must be capable of identifying that a given image does not belong to any of the defined categories, in other words, reject to classify the image. We analyze and propose different methods to solve this problem, commonly called classification with rejection.

The experimental results show that using features extracted from convolutional neural networks, we can obtain accurate recognition performance in a standard classification scenario, without considering rejection. However, the results considering the rejection scenario are not that accurate, pointing out that further research needs to be done on this important topic.

II. Related Work

The method proposed in [1] use points of interest detected by the Harris corner detector and described by the SIFT feature descriptor. The method matches the points of interest of a test image with images in the dataset to identify the category of a given test image. In [2] the authors propose a fast building recognition system using a local invariant region algorithm. However, the method is sensitive to illumination changes. The method Steerable Filter-based Building Recognition was proposed in [11]. This method selects oriented features that can deal with edge information in different angles and for classification it applies LDA followed by a classification performed by the SVM classifier. More recently, the authors in [5] proposed a bag-of-words based method that selects only the points of interest inside buildings to create the BOW representation. Other related works were proposed for the problem of architecture style identification [12], [13], [14]. These methods use the Scale Invariant Feature Transform (SIFT) method to detect and describe points of interest in the images. Then, bag-of-words features are computed using the SIFT features.

The method proposed in [6] use the simple convolutional neural network model LeNet [7] to identify buildings. The authors use this model since they train the neural network from scratch and LeNet requires relatively low computational resources. However, there exist other alternatives, like transfer

978-1-5090-6363-5/17/$31.00 ©2017 IEEE

learning, that can be used to use more complex CNN models without training the CNN model from scratch.

Since AlexNet [15] obtained the highest performance in the competition ILSVRC 2012 (ImageNet Large-Scale Visual Recognition Challenge) more complex neural network models have been proposed. In the competition ILSVRC 2014 the GoogleNet [16] CNN architecture obtained the best performance. The success of GoogleNet shows that the components of the CNN do not necessarily need to be sequentially arranged. In the ILSVRC 2015, the model ResNet [17] attained the best results using a neural network with 152 layers. A modified version of GoogleNet, called Inception-V3, was proposed in [8]. This model improves the results obtained by GoogleNet maximizing the information flow into the network. This is done by constructing networks that balance depth and width.

III. Methodology

In this section, we describe the baseline method Bag-of-Words, the proposed CNN-based method and some techniques designed to perform the task of classification with rejection.

A. Bag-of-Words

The Bag-of-Words (BOW) methodology create a middle-level representation of the image. The process of feature extraction using BOW generally consists of five stages, as shown Figure 1. First, the system takes a set of color images as input and transform them to their grayscale representation. Second, several points of interest are selected inside the image, the SIFT algorithm is the most widely used for this task [18], but other alternatives like SURF [19] are also equally effective. Third, compute local descriptors over those points of interest, SIFT can also be used for feature extraction. Fourth, we select a set of representative descriptors to conform a visual vocabulary (set of visual words called codebook). Finally, the BOW representation is obtained by creating a histogram of word frequencies.

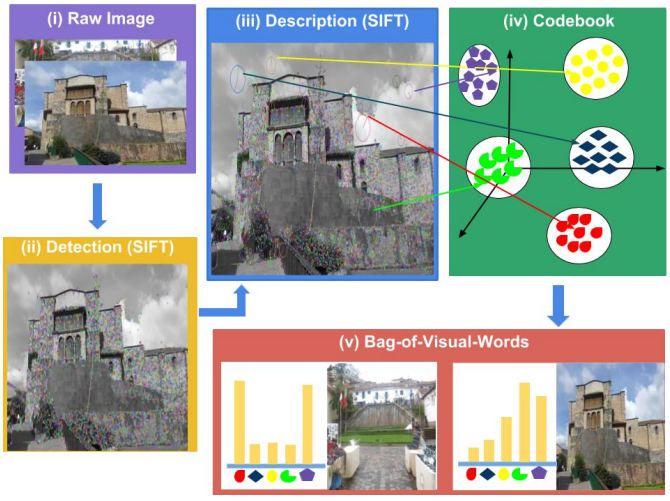


Fig. 1: Bag-of-Words model

B. Proposed method

We propose a transfer learning approach using the Inception-V3 model. The method consists of three stages: preprocessing, compute transfer values and classification. The methodology using transfer learning and a neural network with softmax for classification is depicted in Figure 2.

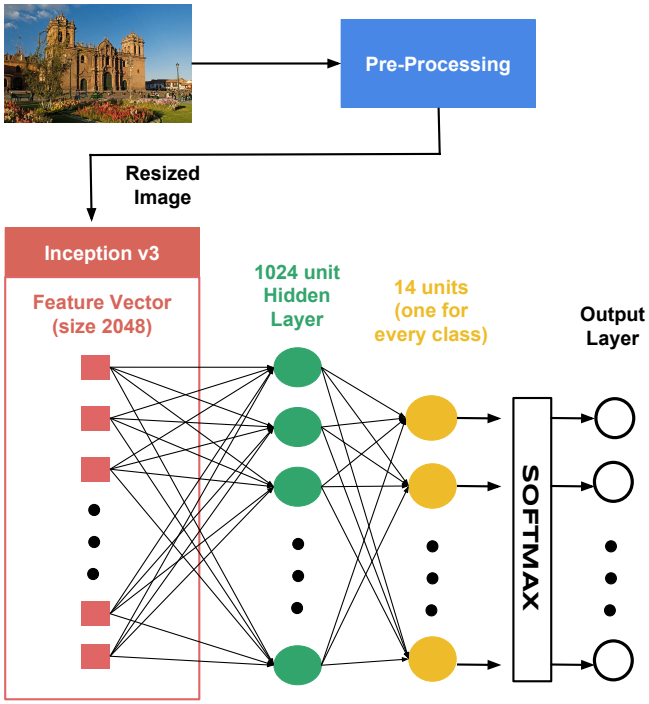


Fig. 2: Transfer learning approach using the Inception-V3 model and a neural network with softmax for classification.

1) Preprocessing: The images are first resized to the size of 500 × 500 so that it can be fed to the Inception-V3 model. In order to improve the accuracy of the method a very well known technique is data augmentation, that consist in performing transformations in the original image, for examples rotation, cropping, contrast and brightness changes, to create a larger set of images to train the model. In particular, we applied rotation, brightness, contrast and saturation transformations. Data augmentation is used successfully in [20].

2) Compute transfer values: The Inception-V3 model is composed of several inception modules connected in a sequential way. Each inception module consists of several convolution operations and one pooling operation. The results of these operations are concatenated to conform the feature map for the next layer. These operations can be performed in parallel making Inception-V3 more efficient than other CNN architectures. Different from the GoogleNet model Inception-V3 limits the filter size for convolutions to a maximum of 3 × 3 sizes. The representation vector of the input image is obtained at different hierarchical levels using the different layers of convolutional neural networks. The 2048 dimensional representation vector

obtained at the next-to-last layer of the Inception-V3 model is used as a feature vector for the images.

3) Classification: These image descriptors obtained by using Inception-V3 can be used to train any classifier like for example a neural network with softmax function as depicted in Figure 2. In this figure the transfer values are the input for a neural network that contains 1024 units in the hidden layer and 14 units (number of classes in the dataset) in the second layer. Then, a softmax layer is used to output the probabilities of the image belonging to each of the classes of interest. In order to evaluate the performance of this classifier, we use the Adam optimizer and cross-entropy loss for updating the weights. The system was trained for 10000 iterations to obtain weights for the classification model.

C. Classification with Rejection

As mentioned before, in this work we also analyze the scenario of classification with rejection. We evaluate the methods one-class SVM, like in [21], and the method Nearest Neighbor Distance Ratio (NNDR) proposed in [22]. In our context the method one-class SVM is trained with all the samples that belong to the classes of interest, *known samples*. Given a test sample, this method outputs a positive score when it estimates that the sample belongs to the *known samples*. In the case of a negative output the sample is rejected, it does not belong to any of the classes of interest and is considered as an *unknown sample*. Given a test samples s the method NNDR obtains its nearest neighbor t. Then, NNDR gets the nearest neighbor u of s, such that u and t have different labels. Afterwards, the method computes the ratio

$$R = d(s,t)/d(s,u), \tag{1}$$

where $d(x,x^{'})$ computes the euclidean distance between x and $x^{'}$ in the feature space. If R is less than a determined threshold the test sample s is considered as a *known sample*. Otherwise, it is classified as an *unknown sample*.

We propose a rejection method based on the uncertainty metric *Breaking Ties* (BT) [23]. First, for a test sample s we compute a classification confidence score $C_i(s)$, for every class of interest i in the data set of m classes. Then, we compute the difference between the best confidence score and second best confidence score

$$\max_1 = \underset{i\in\{1,2,\ldots,m\}}{\arg\max} \{C_i(s)\} \tag{2}$$

$$\max_2 = \underset{i\in\{1,2,\ldots,m\}, i\neq \max_1}{\arg\max} \{C_i(s)\} \tag{3}$$

$$U(s) = C_{\max_1}(s) - C_{\max_2}(s), \tag{4}$$

where $U(s)$ represents a degree of classification uncertainty. High value of $U(s)$ corresponds to most confident classification, while smaller values represent less confident classification. If the classification uncertainty value is greater than a determined threshold we consider s as a *known sample*. Otherwise, the sample is classified as an *unknown sample*. An optimal threshold can be computed using a validation set that contains *known* and *unknown* samples. Note that any method that outputs a classification confidence can be used to compute the uncertainty value $U(s)$.

TABLE I: Classification results obtained by using different BOW and CNN features with different classifiers.

Method	Accuracy	
	Mean	std. dev.
BOW-SIFT + RF	68.90	1.64
BOW-SIFT + SVM	81.14	1.40
BOW-SURF + RF	71.26	1.59
BOW-SURF + SVM	80.84	1.26
Inception-V3 + RF	88.64	1.13
Inception-V3 + SVM	94.58	0.80
Inception-V3 + NN	94.36	1.05

IV. Experimental results

A. Dataset

For the evaluation of the proposed method, we use a new dataset that we collect from 14 historical buildings in the City of Cusco. The dataset contains 2000 images from many websites and several pictures that were taken in place to obtain images with more diverse photometric conditions. Figure 3 illustrates some images from different categories in the data set.

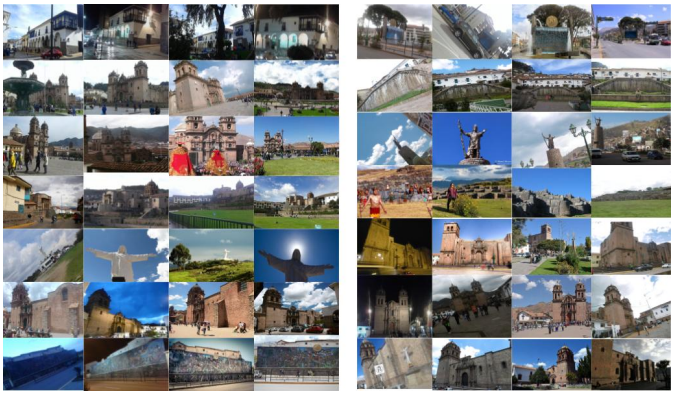


Fig. 3: Some images of the data set that contains images from historical buildings of the city of Cusco.

B. Building recognition

In order to evaluate the performance of the methods, we randomly divide the dataset five times into 50% of samples for training the classifier and 50% for testing it. After performing the process of feature extraction using the BOW method and the transfer learning approach with Inception-V3, we use different models for the classification process, namely Support Vector Machines (SVM), Radom Forest (RF), and the neural network (NN) described in Section III-B3. Table I presents the classification results using different combinations of feature extraction method and classifier models. As can be observed in Table I the methods that use CNN features obtained with the Inception-V3 model attain the best results as compared with the methods that use BOW features. The classifier SVM and NN using CNN features obtain the best results.

TABLE II: Classification results considering the rejection scenario.

Method	Normalized Accuracy	
	Mean	std. dev.
One-class SVM	60.60	0.62
NNDR	78.37	3.33
BT-SVM distances	82.88	1.16
BT-SVM probabilities	81.54	1.08
BT-NN	81.53	1.98

C. Classification with rejection

In order to evaluate the methods that consider the rejection scenario, we use an additional data set that contains 927 images from buildings of several cities around the world [24]. The pictures of this new data set should be rejected by the methods since they do not belong to any of our classes of interest. All the methods presented in this section use the transfer learning features to train the classifier. For the methods based on *Breaking Ties* (BT) we use several techniques that output classification confidence scores. First, we use the distances to the SVM hyperplanes and SVM probabilistic outputs, we call these methods BT-SVM distances and BT-SVM probabilities, respectively. Second, we use the neural network activations from the last layer before the softmax classification layer, we call this method BT-NN.

We use the normalized accuracy metric to measure the performance of the rejection methods. This metric is computed by taking the average of the accuracy of *known samples* and *unknown samples*. Table II presents the results for the analyzed rejection methods. As can be observed the proposed methods based on the *Breaking Ties* strategy obtain the best results.

V. Conclusion

We presented a method for building recognition that obtains accurate results in a challenging dataset that contains images with different photometric conditions and occlusions. We showed that features extracted by CNNs are more adequate than BOW features for the task of building recognition. We also analyzed the rejection scenario, in which the methods have to be capable of indicating that a given test image does not belong to any of the classes of interest. We found that the accuracy obtained in the rejection scenario is much lower than in the standard scenario. Our future work will concern to propose better methods for the rejection scenario and the evaluation other convolutional neural network models. Additionally, we will validate our method in a dataset with a larger number of classes.

VI. Acknowledgment

John E. Vargas-Muñoz thanks the financial support of FAPESP (grant 2016/14760-5).

References

[1] R. Hutchings and W. Mayol-Cuevas, "Building recognition for mobile devices: incorporating positional information with visual features," University of Bristol, Tech. Rep., 2005.

[2] N. J. C. Groeneweg, B. de Groot, A. H. R. Halma, B. R. Quiroga, M. Tromp, and F. C. A. Groen, "A fast offline building recognition application on a mobile telephone," in *Proceedings of the 8th International Conference on Advanced Concepts For Intelligent Vision Systems*, ser. ACIVS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 1122–1132.

[3] S. Lowry, "Visual place recognition: A survey," in *IEEE Transactions on Robotics, vol. 32, no. 1*, 2016, pp. 1–19.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] N. Hascot and T. Zaharia, "Building recognition with adaptive interest point selection," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2017, pp. 29–32.

[6] B. Pavol, "Building recognition system based on deep learning," in *2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR) 19-21 Sept. 2016*, 2016, pp. 1–5.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] S. Christian, V. Vincent, I. Sergey, and S. Jonathon, "Rethinking the inception architecture for computer vision," in *arXiv preprint arXiv:1512.00567v3*, 2015, pp. 1–10.

[9] J. Li and N. M. Allinson, "Subspace learning-based dimensionality reduction in building recognition," *Neurocomputing*, vol. 73, no. 1, pp. 324–330, 2009.

[10] H. Shao, T. Svoboda, and L. Van Gool, "Zubud-zurich buildings database for image based recognition," Tech. Rep., 2003.

[11] J. Li and N. Allinson, "Building recognition using local oriented features," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1697–1704, 2013.

[12] G. Shalunts, Y. Haxhimusa, and R. Sablatnig, "Architectural style classification of building facade windows," in *7th International Symposium, ISVC 2011 Las Vegas, NV, USA, September 26-28, 2011 Proceedings, Part II*, 2011, pp. 280–289.

[13] R. Sablatnig, G. Shalunts, and Y. Haxhimusa, "Classification of gothic and baroque architectural elements," in *19th International Conference on Systems, Signals and Image Processing (IWSSIP) took place 11-13 April 2012 in Vienna, Austria*, 2012, pp. 1–4.

[14] S. Gayane, "Architectural style classification of building facade towers," in *11th International Symposium, ISVC 2015 Las Vegas, NV, USA, December 1416, 2015 Proceedings, Part I*, 2012, pp. 285–294.

[15] K. Alex, S. Ilya, and H. Geoffrey, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems, pp. 10971105 (2012)*, 2012, pp. 1–9.

[16] S. Christian, L. Wei, J. Yangqing, S. Pierre, and R. Scott, "Imagenet classification with deep convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[17] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[18] S. Siddharth, "Sift vs surf: Quantifying the variation in transformations," in *arXiv preprint arXiv:1504.06740*, 2015, pp. 1–4.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[20] B. Lokesh, K. Srinivas, and B. R. Venkatesh, "Crowdnet: A deep convolutional network for dense crowd counting," in *arXiv preprint arXiv:1608.06197v1*, 2016, pp. 1–5.

[21] W. Homenda, M. Luckner, and W. Pedrycz, "Classification with rejection based on various svm techniques," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 3480–3487.

[22] P. R. Mendes Júnior, R. M. de Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. B. Penatti, R. d. S. Torres, and A. Rocha, "Nearest neighbors distance ratio open-set classifier," *Machine Learning*, vol. 106, no. 3, pp. 359–386, 2017.

[23] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, and T. Hopkins, "Active learning to recognize multiple types of plankton," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 589–613, 2005.

[24] G. Tolias and Y. Avrithis, "Speeded-up, relaxed spatial matching," in *in Proceedings of International Conference on Computer Vision (ICCV 2011)*, Barcelona, Spain, November 2011.

Bibliografía

[Akilan et al., 2017] Akilan, T., Wu, Q. J., Yang, Y., and Safaei, A. (2017). Fusion of transfer learning features and its application in image classification. In *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*, pages 1–5. IEEE.

[Bashiri and Geranmayeh, 2011] Bashiri, M. and Geranmayeh, A. F. (2011). Tuning the parameters of an artificial neural network using central composite design and genetic algorithm. *Scientia Iranica*, 18(6):1600–1608.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417.

[Betancourt, 2005] Betancourt, G. A. (2005). Las máquinas de soporte vectorial (svms). *Scientia et technica*, 1(27).

[Bezak, 2016] Bezak, P. (2016). Building recognition system based on deep learning. In *Artificial Intelligence and Pattern Recognition (AIPR), International Conference on*, pages 1–5. IEEE.

[Biau, 2012] Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr):1063–1095.

[Boominathan et al., 2016] Boominathan, L., Kruthiventi, S. S., and Babu, R. V. (2016). Crowdnet: a deep convolutional network for dense crowd counting. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 640–644. ACM.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Buitinck et al., 2013] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

[Canziani et al., 2016] Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

[Chen et al., 2014] Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209.

[Cho and Kim, 2016] Cho, B. and Kim, Y. C. (2016). Recognition of urban buildings by sift matching with google street view images. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pages 46–50. ACM.

[Chollet, 2016] Chollet, F. (2016). Building powerful image classification models using very little data. url https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

[Dasgupta and Nath, 2014] Dasgupta, A. and Nath, A. (2014). Classification of machine learning algorithms.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J. an Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Dumoulin and Visin, 2016] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

[Esposito and Tse, 2017] Esposito, M. and Tse, T. (2017). You've heard about it, but do you understand? everything you need to know about machine learning. urlhttps://www.weforum.org/agenda/2017/05/what-is-machine-learning?utm$_c$*ontent* = $buffer c5529 utm_m edium$ = *socialutm*$_s$*ource* = $facebook.com utm_c ampaign = buffer$.

[Farfan Escobedo, 2017] Farfan Escobedo, Jeanfranco D. Enciso Rodas, L. . V. M. n. J. E. (2017). Towards accurate building recognition using convolutional neural networks. In *IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4. IEEE.

[Fletcher, 2009] Fletcher, T. (2009). Support vector machines explained. *University College London, London*.

[Fridman et al., 2017] Fridman, L., Jenik, B., Angell, W., Dodd, S., and Brow, D. (2017). 6.s094: Deep learning for self-driving cars lecture 1: Introduction to deep learning and self-driving cars. *cars.mit.edu*, 1.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Gouron, 2016] Gouron, R. (2016). Random forests teorÃa y ejemplos. *Conferencia 9, GLAM*.

[Groeneweg et al., 2006] Groeneweg, N. J., de Groot, B., Halma, A. H., Quiroga, B. R., Tromp, M., and Groen, F. C. (2006). A fast offline building recognition application on a mobile telephone. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 1122–1132. Springer.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Huang et al., 2017] Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3.

[Hvass Pedersen, 2016] Hvass Pedersen, M. E. (2016). The mit license (mit). `https://github.com/Hvass-Labs/TensorFlow-Tutorials`.

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

[Jain et al., 1996] Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3):31–44.

[Karpathy, 2016] Karpathy, A. (2016). Cs231n: Convolutional neural networks for visual recognition. *Neural networks*, 1.

[Khan, 2013] Khan, W. (2013). Image segmentation techniques: A survey. *Journal of Image and Graphics*, 1(4):166–170.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li and Karpathy, 2015] Li, F.-F. and Karpathy, A. (2015). Convolutional neural networks for visual recognition.

[Li, 2017] Li, Fei-Fei Johnson, J. . Y. S. (2017). Lecture 2: Image classification pipeline. url http://cs231n.stanford.edu/slides/2017/cs231n$_2$017$_l$ecture2.*pdf*.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Lowry et al., 2016] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19.

[McAfee et al., 2012] McAfee, A., Brynjolfsson, E., et al. (2012). Big data: the management revolution. *Harvard business review*, 90(10):60–68.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[McGonagle and Williams, 2017] McGonagle, J. and Williams, C. (2017). Backpropagation. urlhttps://brilliant.org/wiki/backpropagation/.

[Mejía, 2005] Mejía, E. M. (2005). Metodología de la investigación científica. Master's thesis, *Universidad Nacional Mayor de San Marcos*.

[Nielsen, 2015] Nielsen, M. A. (2015). Neural networks and deep learning.

[Ortega et al., 2010] Ortega, J. P., Del, M., Rojas, R. B., and Somodevilla, M. J. (2010). Research issues on k-means algorithm: An experimental trial using matlab. In *CEUR Workshop Proceedings: Semantic Web and New Technologies*.

[Ozaki, 2014] Ozaki, T. J. (2014). Comparing machine learning classifiers based on their hyperplanes or decision boundaries. urlhttp://tjoen.hatenablog.com/entry/2014/01/06/234155.

[Panchal et al., 2013] Panchal, P., Panchal, S., and Shah, S. (2013). A comparison of sift and surf. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2):323–327.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

[Peterson, 2009] Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883.

[Renuka, 2016] Renuka, J. (2016). Accuracy, precision, recall f1 score: Interpretation of performance measures. url http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[Rere et al., 2015] Rere, L. R., Fanany, M. I., and Arymurthy, A. M. (2015). Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72:137–144.

[Sabour et al., 2017] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

[Sampieri, 2015] Sampieri, R. H. (2015). *Metodología de la investigación 5ta ed.* México: The McGraw-hill Companies.

[Sechidis et al., 2011] Sechidis, K., Tsoumakas, G., and Vlahavas, I. (2011). On the stratification of multi-label data. *Machine Learning and Knowledge Discovery in Databases*, pages 145–158.

[Shalunts, 2015a] Shalunts, G. (2015a). Architectural style classification of building facade towers. In *International Symposium on Visual Computing*, pages 285–294. Springer.

[Shalunts, 2015b] Shalunts, G. (2015b). Segmentation of building facade towers. In *International Symposium on Visual Computing*, pages 185–194. Springer.

[Shalunts et al., 2011] Shalunts, G., Haxhimusa, Y., and Sablatnig, R. (2011). Architectural style classification of building facade windows. In *International Symposium on Visual Computing*, pages 280–289. Springer.

[Shalunts et al., 2012a] Shalunts, G., Haxhimusa, Y., and Sablatnig, R. (2012a). Architectural style classification of domes. *Advances in Visual Computing*, pages 420–429.

[Shalunts et al., 2012b] Shalunts, G., Haxhimusa, Y., and Sablatnig, R. (2012b). Classification of gothic and baroque architectural elements. In *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, pages 316–319. IEEE.

[Shalunts et al., 2012c] Shalunts, G., Haxhimusa, Y., and Sablatnig, R. (2012c). Segmentation of building facade domes. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 324–331.

[Shcherbatov et al., 2017] Shcherbatov, I., Dung, N. T., Glazkov, V., and Protalinskiy, O. (2017). Control movement of mobile robots inside building based on pattern recognition algorithm. In *Control and Communications (SIBCON), 2017 International Siberian Conference on*, pages 1–5. IEEE.

[Shiffman, 2012] Shiffman, D. (2012). The nature of code: Chapter 10. *Neural Networks*.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Solem, 2012] Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. .º'Reilly Media, Inc.".

[Sonka et al., 1993] Sonka, M., Hlavac, V., and Boyle, R. (1993). Image preprocessing. In *Image Processing, Analysis and Machine Vision*, pages 56–111. Springer.

[Srivastava, 2015] Srivastava, S. (2015). Sift vs surf: quantifying the variation in transformations. *arXiv preprint arXiv:1504.06740*.

[Suárez, 2014] Suárez, E. J. C. (2014). Tutorial sobre máquinas de vectores soporte (svm). *Tutorial sobre Máquinas de Vectores Soporte (SVM)*.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

[Thoma, 2015] Thoma, M. (2015). On-line recognition of handwritten mathematical symbols. *arXiv preprint arXiv:1511.09030*.

[Torii et al., 2013] Torii, A., Sivic, J., Pajdla, T., and Okutomi, M. (2013). Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 883–890.

[Triangles, 2017] Triangles (2017). The gradient descent function. url https://www.internalpointers.com/post/gradient-descent-function.

[Tripathy, 2016] Tripathy, A. (2016). Googlenet insights. url https://www.slideshare.net/aurot/googlenet-insights.

[Ujjwalkarn, 2016a] Ujjwalkarn (2016a). An intuitive explanation of convolutional neural networks. url https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[Ujjwalkarn, 2016b] Ujjwalkarn (2016b). A quick introduction to neural networks. url https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

[Van Veen, 2014] Van Veen, F. (2014). The neural network zoo.

[Vega and Palomino, 2003] Vega, J. J. and Palomino, L. G. (2003). El inti raymi inkaico, la verdadera historia de la gran fiesta del sol. *Boletín. Museo de Arqueología y Antropología*, 6:37–71.

[Viñuales, 2004] Viñuales, G. (2004). *El espacio urbano en el Cusco colonial: uso y organización de las estructuras simbólicas*. Epígrafe Editores, S.A.

[Xu et al., 2015] Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

[Zhang, 2000] Zhang, G. P. (2000). Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462.

[Zhang et al., 2011] Zhang, J., Wu, G., Hu, X., Li, S., and Hao, S. (2011). A parallel k-means clustering algorithm with mpi. In *Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on*, pages 60–64. IEEE.