UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

Maestría en Explotación de Datos y Descubrimiento de Conocimiento

Variantes de Vectores de Fisher para la clasificación de imágenes de lesiones de piel mediante redes neuronales profundas residuales

Tesis Presentada para Optar al Título de Magíster de la Universidad de Buenos Aires en Explotación de Datos y Descubrimiento de Conocimiento

Tesista: Ing. Cristian Luciano Salto

Director: Dr. Daniel Acevedo

Buenos Aires, 29 de agosto de 2025

Resumen

El melanoma es un cáncer que surge de los melanocitos, células que se encuentran predominantemente en la piel. En los últimos 20 años el número de casos de cáncer de melanomas se ha duplicado en el mundo, siendo muy importante la detección temprana para su tratamiento. El presente trabajo investiga alternativas al problema de detección de melanomas mediante la clasificación de imágenes dermatoscópicas utilizando redes neuronales residuales (ResNet) y codificando los descriptores de la misma con Vectores de Fisher. Inicialmente, se desarrollaron modelos basados en clasificadores SVM, los cuales emplearon Vectores de Fisher generados a partir de descriptores de imágenes como GLCM, LBP y HOG. Luego se realizó el reentrenamiento de un clasificador ResNet-50. A continuación se aplicaron Vectores de Fisher a los descriptores de distintas muestras de una imagen. Otra alternativa investigada fue generar Vectores de Fisher sobre la base de los descriptores obtenidos como salida del quinto bloque convolucional de la red ResNet-50. Finalmente se realizó un ensamble de las aplicaciones de Vectores de Fisher generados a partir de los descriptores de la red convolucional. Los resultados alcanzados se encuentran en el orden de magnitud de trabajos realizados anteriormente, siendo el ensamble de Vectores de Fisher el que mejor resultados alcanzó según las métricas de evaluación.

Fisher Vector Variants for Skin Lesion Image Classification Using Residual Deep Neural Networks

Abstract

Melanoma is a cancer that arises from melanocytes, cells predominantly found in the skin. In the last 20 years the number of cases of melanoma cancer has doubled in the world, being early detection for its treatment very important. The present work investigates alternatives to the problem of melanoma detection through classification of dermoscopic images using residual neural networks (ResNet) and encoding its descriptors with Fisher Vectors. Initially, models based on SVM classifiers were developed, employing Fisher Vectors generated from image descriptors such as GLCM, LBP, and HOG. Subsequently, a ResNet-50 classifier was retrained. Fisher Vectors were then applied to the descriptors extracted from different regions of an image. Another approach investigated involved generating Fisher Vectors based on the descriptors obtained from the output of the fifth convolutional block of the ResNet-50. Finally, an ensemble of Fisher Vector applications generated from the descriptors of the convolutional network was implemented. The results obtained are comparable to those reported in previous studies, with the Fisher Vector ensemble achieving the best performance according to the evaluation metrics.

Agradecimientos

A mis padres, amigos y compañeros que me apoyaron a lo largo del cursado de la maestría.

Al Dr. Daniel Acevedo, por la generosidad de aceptarme como tesista, y por la infinita paciencia que tuvo para guiarme a lo largo de este trabajo.

A la Dra. María Elena Buemi por su coordinación y vincularme con mi director de tesis.

Índice

Lista de abreviaturas

ANN *Artificial Neural Network*

AUC *Area Under the ROC Curve*

CNN *Convolutional Neural Network*

FV *Fisher Vectors*

GLCM *Gray Level Co-occurrence Matrix*

GMM *Gaussian Mixture Model*

HOG *Histogram of Oriented Gradients*

LBP *Local Binary Patterns*

mAP *Mean Average Precision*

PCA *Principal Component Analysis*

RBF *Radial Basis Function*

ReLu *Rectified Linear Unit*

ResNet *Residual Neural Network*

SGD *Stochastic Gradient Descent*

SVM *Support Vector Machines*

1 Introducción

Con el fin de contextualizar el trabajo desarrollado, en esta primer sección se introduce la problemática de la detección de melanomas a partir de imágenes de lesiones cutáneas. Luego se mencionan diversas aplicaciones desarrolladas para el diagnóstico de melanomas. A continuación, se delimita el alcance de este trabajo, así como los objetivos propuestos. Finalmente, se presenta la estructura organizativa del trabajo.

1.1 Contexto general

El melanoma es un cáncer que surge de los melanocitos, células pigmentadas especializadas que se encuentran predominantemente en la piel. La incidencia de melanoma está aumentando constantemente en las poblaciones occidentales; el número de casos en todo el mundo se ha duplicado en los últimos 20 años. En sus primeras etapas, el melanoma maligno se puede curar mediante resección quirúrgica, pero una vez que ha progresado a la etapa metastásica, es extremadamente difícil de tratar y no responde a las terapias actuales. Los descubrimientos recientes en la señalización celular han proporcionado una mayor comprensión de la biología que subyace al melanoma, y estos avances se están aprovechando para proporcionar fármacos dirigidos y nuevos enfoques terapéuticos [1].

La detección temprana es fundamental para disminuir la mortalidad. La incidencia de detección de melanoma ha aumentado con el tiempo debido en parte a una mayor conciencia y vigilancia de las lesiones cutáneas anormales [2]. En el último tiempo, el dermatoscopio ha ganado una enorme popularidad entre los dermatólogos como una herramienta complementaria para visualizar mejor las estructuras cutáneas e identificar patrones que pueden mejorar el diagnóstico de una amplia gama de enfermedades de la piel. Inicialmente, los expertos en lesiones pigmentadas que fueron los primeros en adoptarlo promovieron el uso del dermatoscopio para aumentar la precisión diagnóstica de los melanomas tempranos y disminuir la recolección de lesiones benignas. Con la adopción actual casi universal de la técnica de diagnóstico por parte de los dermatólogos, el dermatoscopio ahora se emplea para ayudar a identificar una amplia variedad de afecciones inflamatorias, infecciosas y vasculares de la piel, el cabello y las uñas, lo que resulta en la aparición de varias ramas de la dermatoscopia inflamamoscopia, tricoscopía, onicoscopía y entodermoscopía. Por esta razón se están desarrollando muchas aplicaciones utilizando imágenes dermatoscópicas para el diagnóstico de melanomas haciendo que el proceso de evaluación sea cada vez más objetivo, más preciso

y universalmente disponible [3].

Los avances en inteligencia artificial ofrecen soluciones prometedoras para el diagnóstico de melanomas. En particular, las CNN (*Convolutional Neural Network*), sistemas computacionales inspirados en el procesamiento visual humano, destacan por su capacidad para analizar imágenes médicas. Estas redes están compuestas por bloques convolucionales, capas especializadas que detectan progresivamente patrones visuales. Las redes convolucionales pueden identificar patrones complejos en imagénes de lesiones cutáneas, detectando características que podrían indicar la presencia de melanoma. Para enriquecer el análisis se pueden combinar las redes neuronales convolucionales con FV (*Fisher Vectors*), lo cuales codifican atributos visuales como textura y forma, siendo útiles para clasificación de imágenes médicas. Los Vectores de Fisher suelen emplearse como descriptores de una imagen para entrenar un clasificador SVM, un algoritmo de aprendizaje automático que utiliza transformaciones no lineales para diferenciar las clases del problema en cuestión, en este caso lesiones benignas y melanomas. Tanto los conceptos de redes convolucionales, Vectores de Fisher y SVM se abordarán en la Sección 2.

Considerando la importancia crítica de la detección temprana del melanoma y con el objetivo de avanzar el estado del arte en diagnóstico dermatológico, este proyecto propone estudiar alternativas basadas en redes neuronales convolucionales y Vectores de Fisher para el análisis automatizado de imágenes dermatoscópicas.

1.2 Estado del arte

A continuación se presentan una serie de aplicaciones relacionadas al diagnóstico de melanomas que utilizan redes neuronales profundas para extraer descriptores de las imágenes y codificarlos como Vectores de Fisher. Estos métodos varían en función de las redes utilizadas, la capa de la cual extraen los descriptores y el procesamiento de las imágenes. También se presentan otros trabajos en los que se aplican redes neuronales convolucionales con Vectores de Fisher para clasificar otros tipos de imágenes.

En aplicaciones dermatológicas, Yu et al. [4] implementaron redes neuronales residuales para resolver los problemas de degradación y sobreajuste en redes profundas. Luego, construyen una FCRN (*Fully Convolutional Residual Network*) para una segmentación de las lesiones cutáneas. Finalmente, integran la FCRN propuesta (para segmentación) y otras redes residuales muy profundas (para clasificación) formando un *framework* de dos etapas. Yu et al. [5] presentaron un *framework* de clasificación híbrido para la evaluación de imágenes dermatoscópicas mediante la combinación de una red neuronal

convolucional profunda, Vectores de Fisher y máquinas de vectores de soporte. Específicamente, primero se extraen las representaciones profundas de subimágenes de varias ubicaciones de una imagen dermatoscópica reescalada a partir de una CNN previamente entrenada. Luego adoptan métodos de codificación de Vectores de Fisher basados en estadísticas visuales sin orden para generar estas características y construir representaciones más invariantes. Finalmente, las representaciones codificadas en Vectores de Fisher se clasifican para el diagnóstico utilizando un clasificador SVM lineal.

Enfoques similares se implementaron en estudios previos. En el primero de ellos, Yu et al. [6] destacan los méritos del método de aprendizaje profundo y la estrategia de codificación de descriptores locales. Específicamente, las representaciones profundas de una imagen de dermatoscopia se extraen primero utilizando una red neuronal residual muy profunda previamente entrenada en ImageNet. Luego, estos descriptores profundos locales se codifican en Vectores de Fisher para construir una representación agregada de la imagen. Finalmente, las representaciones codificadas se clasifican usando SVM. En el segundo enfoque, Yu et al. [7] emplean representaciones profundas de imágenes dermatoscópicas reescaladas mediante una red neuronal residual (ResNet) preentrenada en un amplio conjunto de datos de imágenes naturales. Luego, estos descriptores profundos locales se acumulan mediante características basadas en la codificación del Vector de Fisher para construir una representación de imagen global. Finalmente, las representaciones codificadas en FV se utilizan para clasificar las imágenes de melanoma utilizando una máquina de vectores de soporte con un núcleo de chi-cuadrado.

Por su parte, Zhang et al. [8] proponen un modelo de SDL (*Synergic Deep Learning*) para abordar el problema de clasificación de imágenes mediante el uso de múltiples DCNN (*Deep Convolutional Neural Networks*) simultáneamente y permitiéndoles aprender mutuamente entre sí. Cada par de DCNN tiene su representación de imagen aprendida concatenada como la entrada de una red sinérgica, que tiene una estructura completamente conectada que predice si el par de imágenes de entrada pertenece a la misma clase. Liberman et al. [9] presentaron distintas alternativas al problema de clasificación de imágenes, donde entrenaron tres clasificadores de imágenes de melanomas. El primero corresponde a una red convolucional VGG-16, los otros dos corresponden a dos modelos híbridos. Cada modelo híbrido está compuesto por una red de entrada VGG-16 y un SVM como clasificador. Estos modelos se entrenan con Vectores de Fisher calculados con los descriptores que son la salida de la red convolucional antes mencionada. La diferencia entre estos dos últimos clasificadores radica en que uno tiene imágenes segmentadas como entrada de la red VGG-16, mientras que el otro utiliza imágenes no segmentadas. La segmentación se realiza mediante una modificación de una

red Unet basada en VGG. Finalmente, Yu et al. [10] proponen un *framework* para el reconocimiento automático de lesiones cutáneas mediante la codificación basada en redes cruzadas de múltiples redes convolucionales. Los mapas de activación de salida de cada red se extraen como mapas de indicadores para seleccionar los descriptores convolucionales profundos locales (es decir, patrones locales y color) en imágenes de dermatoscopia. Además, este mapa de una capa convolucional captura las regiones semánticas de la imagen de entrada y localiza el objeto de destino. Estas características seleccionadas se acumulan en un mapa de características informativo, que potencialmente se conservan mejor en los mapas de características convolucionales. Finalmente, usan Vectores de Fisher para codificar las características seleccionadas.

Para finalizar se presentan algunos trabajos donde implementaron Vectores de Fisher en otros conjuntos de datos. Estas aplicaciones resultan importante mencionarlas ya que se pueden replicar las arquitecturas propuestas para el problema de diagnóstico de melanomas. En et al. [11], dada una CNN previamente entrenada, reemplazaron las dos primeras capas completamente conectadas con dos capas convolucionales equivalentes para obtener una gran cantidad de activaciones densas de múltiples escalas. Las activaciones son seguidas por una capa MPP (*Multi-scale Pyramid Pooling*) que acumulan los Vectores de Fisher de cada escala. Estas representaciones se combinan con un SVM para realizar la tarea de clasificación. Xie et al. [12] utilizando distintos conjuntos de datos y arquitecturas de redes convolucionales para clasificación de imágenes plantean dos representaciones basadas en diccionarios: MLR (*Mid-level Local Representation*) y CFV (*Convolutional Fisher Vector Representation*). En MLR, se utiliza un método de agrupación en dos etapas, es decir, agrupación ponderada en las partes de una sola imagen seguida de una agrupación de todas las partes representativas de todas las imágenes, para generar una mezcla de clases o un diccionario de partes específicas de clase. Después de eso, el diccionario de piezas se usa para operar con las entradas de imagen de múltiples escalas para generar una representación de nivel medio. En CFV, se utiliza una estrategia de entrenamiento de GMM (*Gaussian Mixture Model*) de escala múltiple y proporcional a la escala para generar Vectores de Fisher basados en la última capa convolucional de CNN. Finalmente integra la información complementaria de MLR, CFV y las características de CNN de la capa completamente conectada para entrenar un clasificador de SVM. Li et al. [13] realizaron clasificación de imágenes satelitales a través de una arquitectura híbrida llamada codificación de ADFF (*Aggregated Deep Fisher Feature*). El método consiste en utilizar una red convolucional preentrenada para obtener los descriptores del primer bloque no convolucional y los descriptores de Fisher de la capa convolucional que mejor codifica las imágenes. Ambos descriptores son acumulados y su dimen-

sionalidad es reducida mediante PCA (*Principal Component Analysis*) para finalmente entrenar un clasificador SVM.

1.3 Alcance y objetivos

En el presente trabajo se proponen distintas alternativas al problema de clasificación de melanomas. En primer lugar, se desarrollan modelos basados en Vectores de Fisher generados a partir de descriptores de imágenes tradicionales.

En una segunda instancia, se exponen tres alternativas al problema de clasificación de melanomas con redes convolucionales. El primer enfoque involucra el entrenamiento de una red convolucional ResNet-50. El segundo implica la extracción de descriptores a partir de muestras de la imagen original para luego codificarlos con Vectores de Fisher. Mientras que en el tercer enfoque, la imagen completa es procesada para obtener los descriptores a partir del quinto bloque convolucional de la red y luego se realiza la codificación con Vectores de Fisher.

El objetivo del presente trabajo es desarrollar un modelo de clasificación de lesiones de piel que permita distinguir las lesiones benignas de los melanomas. Para alcanzar este objetivo se experimentarán distintas alternativas utilizando Vectores de Fisher. Cada una de estas alternativas tendrán los siguientes objetivos específicos:

- Lograr capacidad de generalización en un conjunto de imágenes no utilizadas para el desarrollo del modelo.
- Alcanzar resultados comparables o superadores al de otras aplicaciones desarrolladas sobre este conjunto de datos. Para ello se tendrán en cuenta las aplicaciones mencionadas en la Sección 1.2.

1.4 Organización del trabajo

El resto de este documento está organizado de la siguiente manera. En la Sección 2 se exponen los principales conceptos utilizados en el trabajo, ellos son las redes neuronales convolucionales y los Vectores de Fisher. La Sección 3 describe los detalles de los experimentos realizados. La Sección 4 presenta los resultados experimentales. Luego en la Sección 5 se realiza un análisis de los desafíos encontrados al realizar el trabajo y finalmente en la Sección 6 las conclusiones del trabajo.

2 Marco teórico

En esta sección se desarrollan los principales conceptos utilizados a lo largo del trabajo. En primer lugar, se mencionan diferentes formas de generar descriptores a partir de una imagen, específicamente se detallan la matriz de co-ocurrencia de nivel de gris, los patrones locales binarios y los histogramas de gradientes orientados. A continuación se presentan las redes neuronales convolucionales, describiendo sus componentes esenciales y el proceso de entrenamiento. Finalmente, se abordan los conceptos claves para el cálculo de Vectores de Fisher, incluyendo el modelo de mezcla gaussiana, el análisis de componentes principales y la maquina de vectores de soporte.

2.1 Descriptores de imágenes

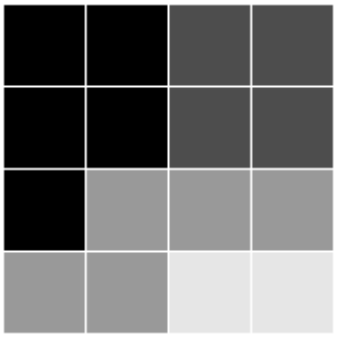
Los descriptores de imágenes que se presentan dentro de este apartado son la matriz de co-ocurrencia de nivel de gris, los patrones locales binarios y los histogramas de gradientes orientados. Estas son técnicas utilizadas para la extracción de características en imágenes que luego puedan ser utilizadas para la clasificación o el reconocimiento de patrones. En todas las técnicas, el análisis se realiza considerando la disposición espacial de los píxeles en la imagen. A diferencia de las redes neuronales convolucionales estas técnicas no requieren de un gran conjunto de datos ni altos recursos de computación.

2.1.1 Matriz de co-ocurrencia de nivel de gris

GLCM (*Gray Level Co-occurrence Matrix*) es una tabulación de la frecuencia con la que ocurren diferentes combinaciones de valores de brillo de píxeles (niveles de gris) en una imagen [14]. La mayoría de los cálculos de textura GLCM utilizados actualmente fueron sistematizados en una serie de artículos de Robert Haralick y coautores en la década de 1970 [15] y [16]. Las principales mejoras en GLCM desde esa década hasta la actualidad han residido en algoritmos de cálculo más rápidos, en lugar de cambios en las estadísticas en sí.

A continuación se presenta un ejemplo para el cálculo de GLCM de una imagen de 4 niveles de grises. Las imágenes de 8-bit tienen 256 niveles de grises pero para mostrar un ejemplo sencillo se utilizarán solamente 4 niveles. La Fig. 1 contiene la imagen de ejemplo con sus valores en escala de grises.

La textura GLCM considera la relación entre dos píxeles a la vez, llamados píxel de referencia y píxel vecino. Para el cálculo de la matriz de co-ocurencia de nivel de gris, además de definir los niveles de grises, es necesario especificar la distancia o ventana entre los píxeles y el ángulo o dirección a considerar



(a) Imagen de ejemplo en escala de grises.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

(b) Valores en escala de grises de la imagen de ejemplo.

Figura 1: Ejemplo de imagen para el cálculo de la matriz de co-ocurrencia de nivel de gris.

entre el píxel de referencia y el píxel vecino. Para este ejemplo se utilizaron los píxeles que son vecinos inmediatos, es decir, con una distancia equivalente a 1. En cuanto a la dirección, se consideró el píxel vecino que esté a la derecha del píxel de referencia, es decir, con un ángulo equivalente a 0°entre ellos. Así de esta manera, podemos marcar en la Fig. 2 distintas relaciones entre píxeles marcando en rojo aquellos píxeles de referencia y en azul sus vecinos correspondientes.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Figura 2: Relaciones entre píxeles de referencia, en color rojo, y píxeles vecinos, en color azul.

A partir de estas relaciones, se puede generar la matriz de co-ocurrencia de nivel de gris. Esta tiene un tamaño de $n \times n$ donde n esta dado por el nivel de grises, en nuestro ejemplo $n = 4$. Las filas corresponden a los niveles de grises de los píxeles de referencia ordenados de forma creciente, así la primera fila corresponde al valor 0 en este caso y la última fila al valor 3. En el caso de las columnas, estas representan los niveles de grises de los píxeles

vecinos respetando el orden de los valores de los píxeles de manera creciente de izquierda a derecha. De esta manera en la celda de la fila 1 y columna 2, tendremos la cantidad de veces que un píxel de referencia con valor 0 tiene como vecino a su derecha a un píxel con valor 1, en este caso 2. Si repetimos este proceso para todas las celdas obtendremos la matriz de co-ocurrencia de nivel de gris. En la Fig. 3 se representa cómo quedaría esta matriz para nuestro ejemplo.

0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1
	0	1	2	3

Figura 3: Matriz de co-ocurrencia de nivel de gris para el ejemplo presentado.

A partir de la matriz de co-ocurrencia de nivel de gris se pueden calcular diferentes características, o *features*. A continuación se mencionan algunas de estas:

- El contraste que también se denomina "varianza de suma de cuadrados" y, ocasionalmente, "inercia". Este concepto parte del principio que para enfatizar una gran cantidad de contraste, es necesario crear ponderaciones de modo que el cálculo dé como resultado una cifra mayor cuando haya un gran contraste entre píxeles adyacentes. Los valores en la diagonal de la matriz de co-ocurrencia de nivel de gris no muestran contraste y el contraste aumenta a medida que se aleja de la diagonal. Por lo tanto, se requiere una ponderación que aumente a medida que aumenta la distancia desde la diagonal. El cálculo del contraste se define en la próxima ecuación donde N representa la cantidad de niveles de gris, i y j las filas y columnas de la matriz, mientras que $P_{i,j}$ es la frecuencia relativa de la celda de la matriz correspondiente a la fila i y columna j.

$$Contraste = \sum_{i,j=0}^{N-1} P_{i,j}(i-j)^2 \tag{1}$$

- En lugar de que los pesos aumenten exponencialmente a medida que uno se aleja de la diagonal como lo hace el contraste, en disimilitud los pesos aumentan linealmente.

$$Disimilitud = \sum_{i,j=0}^{N-1} P_{i,j}|i-j| \tag{2}$$

- La disimilitud y el contraste dan como resultado números mayores para más ventanas que muestran más contraste. Si los pesos disminuyen alejándose de la diagonal, la medida de textura calculada será mayor para las ventanas con poco contraste. La homogeneidad pondera los valores por el inverso del peso del contraste, con pesos que disminuyen exponencialmente alejándose de la diagonal.

$$Homogeneidad = \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1+(i-j)^2} \tag{3}$$

- El ASM (*Angular Second Moment*) utiliza cada valor de la matriz como peso propio. Los valores altos de ASM se dan cuando la ventana está muy ordenada.

$$ASM = \sum_{i,j=0}^{N-1} P_{i,j}^2 \tag{4}$$

- La raíz cuadrada del ASM a veces se utiliza como medida de textura y se denomina Energía.

$$Energía = \sqrt{ASM} \tag{5}$$

- La textura de correlación mide la dependencia lineal de los niveles de gris con respecto a los de los píxeles vecinos.

$$Correlación = \sum_{i,j=0}^{N-1} P_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \tag{6}$$

donde μ_i y μ_j son las medias dadas por:

$$\mu_i = \sum_{i,j=0}^{N-1} i(P_{i,j}) \tag{7}$$

y

$$\mu_j = \sum_{i,j=0}^{N-1} j(P_{i,j}) \tag{8}$$

Mientras que σ_i^2 y σ_j^2 representan los desvíos estándar:

$$\sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j}(i - \mu_i)^2 \tag{9}$$

y

$$\sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j}(j - \mu_j)^2 \tag{10}$$

Los gráficos incluidos en esta sección se basan en Hall-Beyer [14].

2.1.2 Patrones binarios locales

El operador LBP (*Local Binary Patterns*) es uno de los descriptores de textura con mejor rendimiento y se ha utilizado ampliamente en diversas aplicaciones. Ha demostrado ser altamente discriminante y sus principales ventajas, a saber, su invariancia a los cambios monótonos de nivel de gris y la eficiencia computacional, lo hacen adecuado para tareas de análisis de imágenes exigentes [17].

La idea básica de este método se presenta a continuación. Si consideramos un vecindario de 3×3 alrededor de cada píxel, ver Fig. 4. Todos los vecinos que tengan valores más altos que el valor del píxel central toman valor 1 y todos aquellos que tienen valores menores o iguales al valor del píxel central toman valor 0 [18]. La Fig. 5, adaptada de Petrou et al. [18], muestra como se aplica el umbral sobre la diferencia entre el valor del píxel central y sus vecinos.

Para poder trabajar con texturas a diferentes escalas, el operador LBP se amplió posteriormente para utilizar vecindarios de diferentes tamaños [19]. La definición del vecindario local como un conjunto de puntos de muestreo espaciados uniformemente en un círculo centrado en el píxel que se va a etiquetar permite cualquier radio y número de puntos de muestreo. La interpolación bilineal se utiliza cuando un punto de muestreo no cae en el centro de un píxel. En lo sucesivo, se utilizará la notación (P, R) para los vecindarios de píxeles, lo que significa P puntos de muestreo en un círculo de radio R.

El operador LBP produce 2^P diferentes valores, correspondientes a los 2^P diferentes patrones binarios que pueden formarse por los P píxeles vecinos.

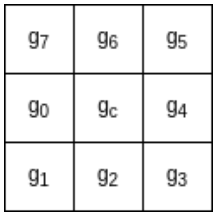


Figura 4: Píxel central g_c y píxeles vecinos $g_0, ..., g_7$.

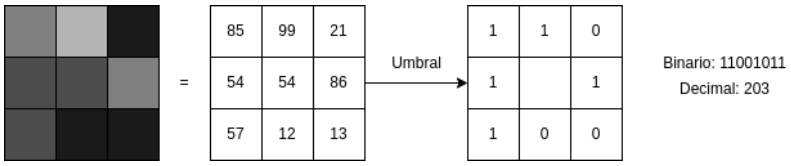


Figura 5: Ejemplo de operación de LBP.

Cuando la imagen es rotada, los valores de grises g_i se moverá correspondientemente a lo largo del perímetro del círculo alrededor de g_c. Esto generará que los valores de LBP sean diferente al aplicar rotaciones. Esto no aplica a patrones comprendidos por solamente ceros (o unos) donde el resultado se mantiene constante a todas las rotaciones. Para eliminar el efecto de la rotación, esto es, para asignar un identificador único a cada patrón binario local invariante de rotación, definimos:

$$LBP^{ri}_{P,R} = min\{ROR(LBP_{P,R}, i)|i = 0, 1, ..., P - 1\} \tag{11}$$

donde $ROR(x, i)$ realiza un desplazamiento circular bit a bit a la derecha del bit P número x i veces. En términos de píxeles de imagen, simplemente corresponde a rotar el conjunto vecino en el sentido de las agujas del reloj tantas veces que un número máximo de bits más significativos sea 0 comenzando desde g_{P-1}.

Otra extensión del operador original es la definición llamada patrones uniformes. Un patrón local binario es llamado uniforme si los patrones binarios contienen como máximo 2 transiciones de 0 a 1 o viceversa cuando el patrón binario es considerado circular. Por ejemplo, los patrones 00000000 (0 transiciones), 01110000 (2 transiciones) y 11001111 (2 transiciones) son uniformes mientras que los patrones 11001001 (4 transiciones) y 01010011 (6 transiciones) no lo son. En el cálculo del histograma LBP, se utilizan patrones uniformes de modo que el histograma tiene un contenedor separado para cada patrón uniforme y todos los patrones no uniformes se asignan a un solo contenedor.

2.1.3 Histograma de gradientes

El método HOG (*Histogram of Oriented Gradients*) se basa en la evaluación de histogramas locales normalizados de las orientaciones de los gradientes de imágenes en una cuadrícula densa. La idea básica es que la apariencia y la forma de los objetos locales a menudo se pueden caracterizar bastante bien mediante la distribución de gradientes de intensidad locales o direcciones de los bordes, incluso sin un conocimiento preciso de las posiciones del gradiente o borde correspondiente. En la práctica, esto se implementa dividiendo la imagen en pequeñas regiones espaciales denominadas celdas. Para cada celda se acumula un histograma unidimensional local de direcciones de gradiente u orientaciones de borde sobre los píxeles de la celda. Las entradas del histograma combinadas forman la representación [20]. A continuación se mencionan las distintas etapas para el cálculo de los descriptores HOG.

En primer lugar, se puede aplicar opcionalmente una corrección *gamma* de los píxeles de la imagen para reducir la influencia de los efectos de iluminación. Esta corrección generalmente implica la aplicación de una transformación no lineal a cada píxel, la cual consiste en elevar el valor de intensidad a una potencia menor que 1 y luego multiplicar el resultado por un escalar para alcanzar el efecto de corrección deseado.

Luego, se calculan los gradientes de primer orden de cada píxel para las distintas bandas de colores. Los gradientes se computan por columna $\nabla c_{i,j}$ y por fila $\nabla f_{i,j}$. Para un píxel específico en una determinada banda de color, el cálculo del gradiente en dirección vertical se realiza restando el valor del píxel a la izquierda del valor del píxel a la derecha. Por otro lado, para calcular el gradiente en dirección horizontal, se resta el valor del píxel superior del valor del píxel inferior. Si tomamos como ejemplo los valores de la Fig. 6 donde (i, j) refiere a la posición del pixel central (es decir, $P_{i,j} = 9$), el cálculo de los gradientes sería de la siguiente manera:

$$\nabla f_{i,j} = P_{i+1,j} - P_{i-1,j} = 11 - 7 = 4 \tag{12}$$

$$\nabla c_{i,j} = P_{i,j+1} - P_{i,j-1} = 13 - 5 = 8 \tag{13}$$

Con los gradientes por columna y fila, se puede obtener la magnitud y orientación del gradiente. Siguiendo con el ejemplo tenemos:

$$\nabla = \sqrt{\nabla f_{i,j}^2 + \nabla c_{i,j}^2} = \sqrt{4^2 + 8^2} \approx 8{,}94 \tag{14}$$

$$\theta = \arctan \frac{\nabla c_{i,j}}{\nabla f_{i,j}} = \arctan \frac{8}{4} \approx 26^\circ \tag{15}$$

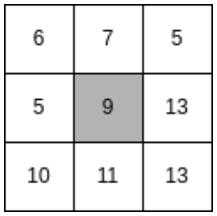


Figura 6: Ejemplo de imagen para el cálculo del gradiente del píxel central $P_{i,j} = 9$.

Este procedimiento se realiza para las distintas bandas de colores de la imagen. En cada píxel se selecciona la magnitud y orientación de la banda de color que tenga mayor predominancia, es decir, mayor magnitud del gradiente.

Seguidamente, las imágenes se dividen en regiones llamadas celdas. Las celdas son subimágenes cuadradas que, de forma contigua y no solapada, cubren toda la imagen. Para cada celda se obtiene un histograma de orientaciones. La cantidad de rangos de valores del histograma va a depender de un parámetro denominado orientaciones. Estos grupos están separados uniformemente entre 0°y 180°. De esta manera, si tomamos 4 orientaciones, los rangos serían $[0; 45)$, $[45; 90)$, $[90; 135)$ y $[135; 180)$. Para cada uno de estos grupos, se suman las magnitudes de los gradientes de cada píxel siempre y cuando la orientación del píxel este comprendido por el rango en cuestión. Luego esta sumatoria es divida por la cantidad de píxeles de la celda. De esta forma, obtenemos un histograma de orientaciones para cada una de las celdas.

Finalmente, se realiza una normalización sobre los histogramas obtenidos. Las intensidades de los gradientes varían en un amplio rango debido a las variaciones locales en la iluminación y el contraste entre el primer plano y el fondo, por lo que una normalización eficaz del contraste local resulta esencial [20]. Esta normalización se realiza sobre grupos de celdas llamados bloques. Normalmente, cada celda individual se comparte entre varios bloques, pero sus normalizaciones dependen del bloque y, por lo tanto, son diferentes. En la Fig. 7 se presenta un ejemplo de un desplazamiento de un bloque de celdas de 2×1 sobre celdas de 2×2 píxeles. A continuación se presentan algunas alternativas para la normalización.

Sea v el vector no normalizado que representa al histograma de un bloque, $||v||_k$ su k-norma para $k = 1, 2$ y ϵ sea una constante pequeña. Algunos procedimientos de normalización son:

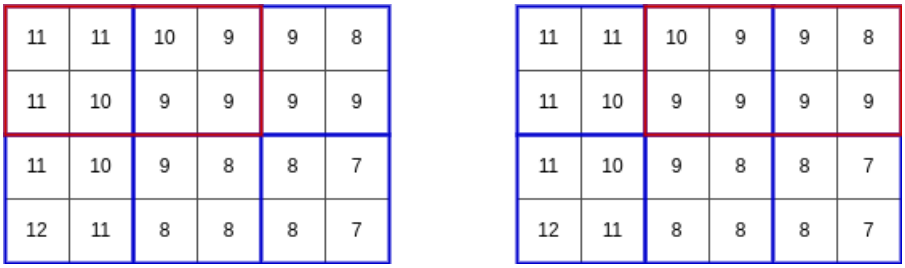
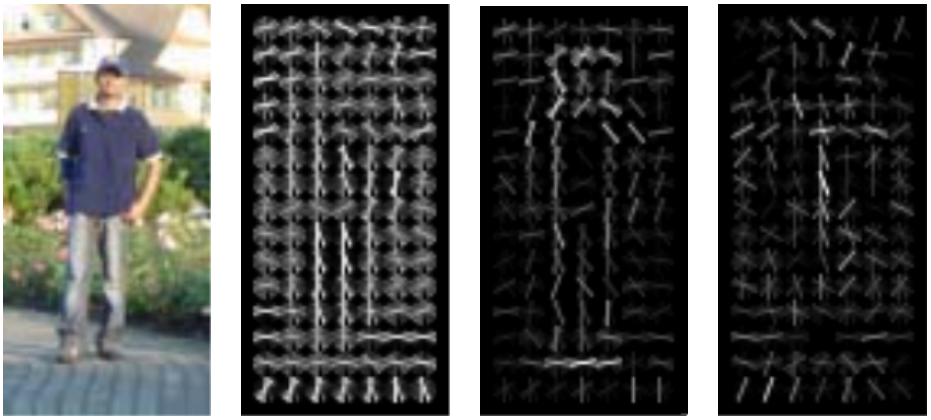


Figura 7: Desplazamiento de un bloque de celdas de 2×1 (en rojo) sobre celdas de 2×2 píxeles.

- *L2-norm*:
$$v \to \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}} \tag{16}$$
- *L2-Hys*: *L2-norm* seguida por un recorte (limitar los valores de v a 0.2) y normalizado nuevamente.
- *L1-norm*:
$$v \to \frac{v}{||v||_1 + \epsilon} \tag{17}$$
- *L1-sqrt*: *L1-norm* seguido por la raíz cuadrada.
$$v \to \sqrt{\frac{v}{||v||_1 + \epsilon}} \tag{18}$$

N. Dalal et al. [20] utilizaron los descriptores de HOG combinados con un clasificador SVM (*Support Vector Machines*) para detectar la presencia de humanos en imágenes. Los coeficientes del SVM lineal entrenado dan una medida de cuánto peso puede tener cada celda de cada bloque en la decisión de clasificación final. Un examen minucioso de la Fig. 8c muestra que las celdas más importantes son las que típicamente contienen los contornos humanos principales (especialmente la cabeza, los hombros y los pies) normalizados con respecto a los bloques que se encuentran fuera del contorno. En otras palabras, a pesar de los fondos complejos y desordenados, el detector se basa principalmente en el contraste de los contornos de las siluetas contra el fondo, no en los bordes internos o en los contornos de las siluetas contra el primer plano. La ropa estampada y las variaciones de pose pueden hacer que las regiones internas no sean confiables como señales, o las transiciones del primer plano al contorno pueden confundirse con sombreados suaves y efectos de sombras. De manera similar, la Fig. 8d ilustra que los gradientes dentro de la persona (especialmente los verticales) generalmente cuentan como señales



(a) Imagen de ejemplo.

(b) Descriptores de HOG.

(c) Descriptor HOG ponderado por los coeficientes positivos de SVM.

(d) Descriptor HOG ponderado por los coeficientes negativos de SVM.

Figura 8: Ejemplo de descriptores de HOG y ponderación de pesos de un clasificador SVM para detectar humanos. Imágenes tomadas de N. Dalal et al. [20].

negativas, presumiblemente porque esto suprime los falsos positivos en los que las líneas verticales largas activan las células verticales de la cabeza y las piernas.

2.2 Redes Neuronales

Una ANN (*Artificial Neural Network*) es un sistema de procesamiento computacional inspirado en gran medida en la forma en que funcionan los sistemas nerviosos biológicos como el cerebro humano. Las ANNs se componen principalmente de una gran cantidad de nodos computacionales interconectados denominados neuronas, cuyo trabajo se entrelaza de manera distribuida para aprender colectivamente de la entrada con el fin de optimizar su salida final [21].

La estructura básica de una ANN se puede modelar como se muestra en la Fig. 9. La capa de entrada puede recibir un vector multidimensional como datos de entrada, que luego los distribuirá a la primera capa oculta. Las capas ocultas procesarán los datos recibidos de la capa anterior y generarán como salida valores que serán a su vez la entrada de la siguiente capa. Tener múltiples capas ocultas apiladas unas sobre otras comúnmente se denomina arquitectura profunda de red neuronal.

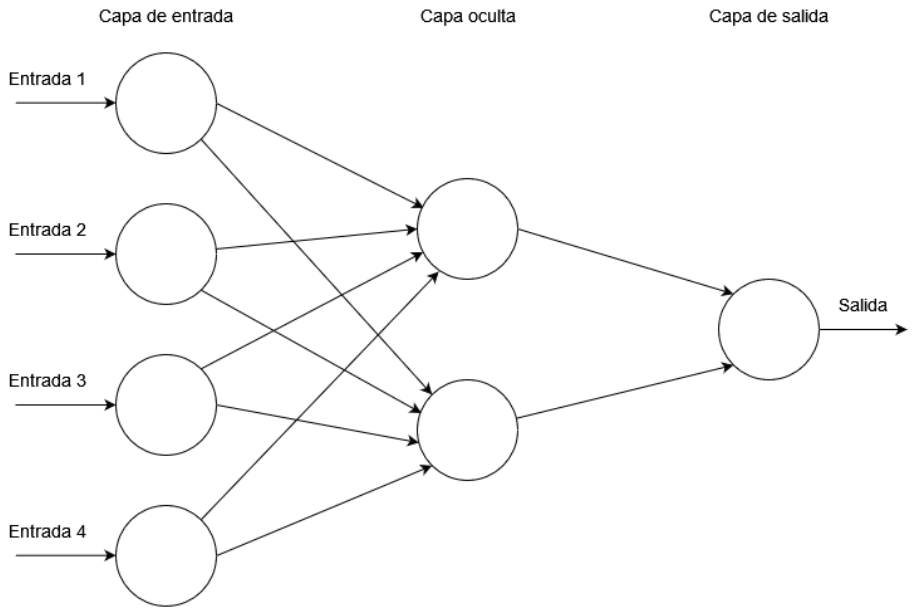


Figura 9: Una FNN (*Feedforward Neural Network*) de tres capas, compuesta por una capa de entrada, una capa oculta y una capa de salida. Esta estructura es la base de una serie de arquitecturas ANN comunes, incluidas, entre otras, las RBM (*Restricted Boltzmann Machine*) y las RNN (*Recurrent Neural Network*).

2.3 Redes Neuronales Convolucionales

Una de las mayores limitaciones de las formas tradicionales de ANN es que tienden a tener problemas con la complejidad computacional requerida para el entrenamiento de modelos que utilizan datos de imágenes. Los conjuntos de datos de evaluación comparativa de aprendizaje automático, como la base de datos MNIST de dígitos escritos a mano [22], son adecuados para la mayoría de las formas de ANN, debido a su dimensionalidad de imagen relativamente pequeña de 28×28. Con este conjunto de datos, una sola neurona en la primera capa oculta contendrá 784 pesos ($28 \times 28 \times 1$ debido a que MNIST está normalizado a solo valores en blanco y negro), lo cual es manejable para la mayoría de las formas de ANN. En cambio, si se considera una entrada de imagen con un tamaño mayor, por ejemplo de 64×64, el número de pesos de una sola neurona de la primera capa aumenta sustancialmente a 12288. Si esta imagen tiene color, deberá ser mucho más grande la escala de entrada, por lo que se volverá muy costoso trabajar con ANN este tipo de datos.

Para procesar estructuras de datos más complejas se utilizan las redes neuronales convolucionales (CNN). Estas permiten codificar las característi-

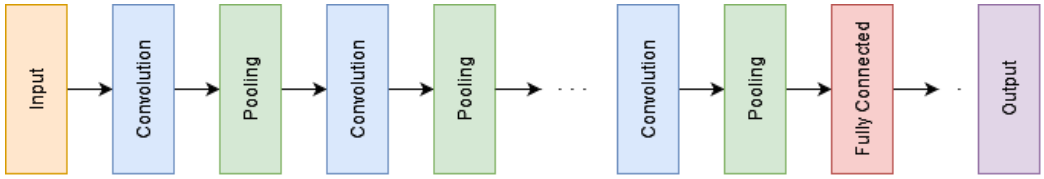


Figura 10: Bloque de construcción de una CNN típica.

cas principales de las imágenes en la arquitectura de la red reduciendo la cantidad de parámetros del modelo [21].

Las CNN son un tipo de redes neuronales artificiales diseñadas para el procesamiento de datos que tienen un patrón de cuadrícula, como las imágenes. Estas redes están inspiradas en la organización de la corteza visual animal [23, 24] y diseñadas para aprender jerarquías espaciales de características de forma automática y adaptativa, desde patrones de bajo nivel hasta patrones de alto nivel [25].

2.4 Elementos de una red convolucional

Una CNN típica, en el contexto de problemas de clasificación, se compone de uno o varios bloques de convolución. Como se muestra en la Fig. 10, las CNN también pueden incluir capas de submuestreos, algunas capas totalmente conectadas y una capa de salida [26]. Las figuras de esta sección son adaptaciones de Sultana et al. [26] y Albawi et al. [27].

2.4.1 Capa convolucional

La capa convolucional es la parte central de una CNN. La capa convolucional calcula la convolución de las imágenes de entrada utilizando filtros kernel para extraer características fundamentales. La convolución es una operación lineal entre dos funciones x y w, que obtiene como resultado una función que representa la magnitud en la que se superponen x y una versión trasladada e invertida de w. En la terminología de redes convolucionales, el primer argumento (en este ejemplo, la función x) de la convolución se suele denominar entrada y el segundo argumento (en este ejemplo, la función w) como *kernel* [28]. La salida a veces se denomina mapa de características (*feature maps*). La Fig. 11 muestra la operación convolucional típica en 2D [26].

Formalmente, la convolución se puede definir de la siguiente manera:

$$y(t) = (x * w)(t) = \int x(a)w(t-a)\,d(a) \tag{19}$$

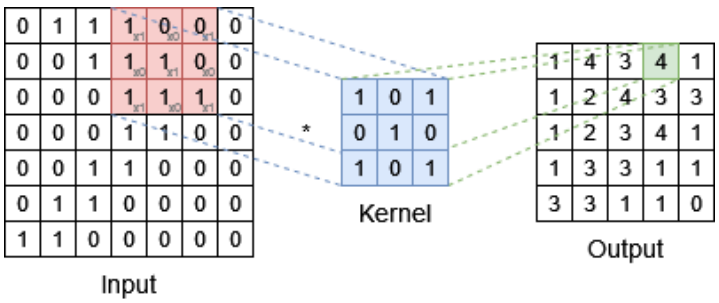


Figura 11: Capa convolucional.

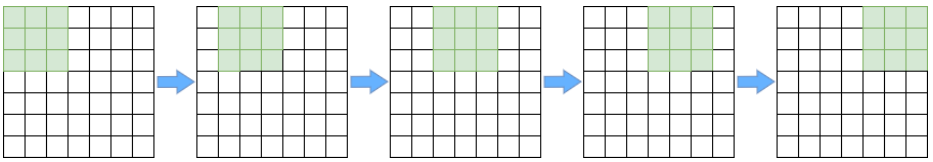


Figura 12: Paso 1. La ventana de filtro se mueve una vez por cada conexión.

donde $*$ es la operación de convolución.

Una de las herramientas que tienen las redes convolucionales para reducir la dimensión del mapa de características es el paso (*stride*). En el ejemplo mencionado anteriormente, se asume que la siguiente característica de la capa tiene solapamiento con sus vecinos al mirar las regiones. Este solapamiento es posible manipularlo controlando los pasos. La Fig. 12 muestra una imagen de 7×7. Si movemos el filtro de a 1 elemento, obtenemos una salida de 5×5. Observamos que las salidas de las tres matrices de la izquierda tienen un solapamiento (esto también ocurre con las tres matrices de salida de la derecha). Sin embargo, si nos movemos sobre la entrada haciendo pasos de 2 elementos, la salida será de 3×3. En pocas palabras, con los pasos reducimos el tamaño de la salida [27, 21].

La ecuación 20 formaliza esto, dada una imagen de tamaño $N \times N$ y un filtro de tamaño $F \times F$ la salida, mostrada en la Fig. 13, es O.

$$O = 1 + \frac{N - F}{S} \tag{20}$$

Donde N es el tamaño de entrada, F es el tamaño del filtro y S es la cantidad de pasos.

Uno de los inconvenientes de la etapa de convolución es la pérdida de información que puede existir en el borde de la imagen. Debido a que solo se capturan cuando los filtros se deslizan, nunca tienen la oportunidad de ser vistos. Un método simple y eficiente para resolver este problema es usar

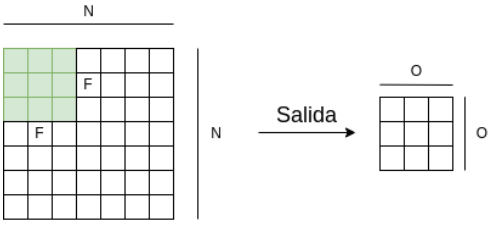


Figura 13: Efecto de la cantidad de pasos en la salida. Ejemplo para $N = 7$, $S = 2$, $F = 3$ y $O = 3$.

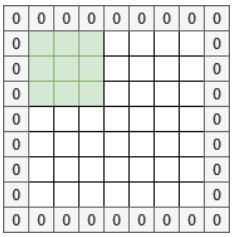


Figura 14: Relleno de ceros (*zero padding*).

rellenos de ceros. El otro beneficio del relleno es manejar el tamaño de salida. Por ejemplo en la Fig. 12, con $N = 7$ y $F = 3$ y 1 cantidad de pasos, la salida será 5×5 (que se contrae de una entrada de 7×7).

Sin embargo, agregando un relleno de ceros (*zero padding*), la salida será de 7×7, que es exactamente la misma que la entrada. La ecuación 20 modificada incluyendo el relleno de ceros es 21.

$$O = 1 + \frac{N + 2P - F}{S} \tag{21}$$

Donde P es la cantidad de capas de relleno de ceros (por ejemplo, $P = 1$ en Fig. 14). Esta idea de relleno nos ayuda a evitar que el tamaño de salida de la red se reduzca con la profundidad. Por lo tanto, es posible tener cualquier número de redes convolucionales profundas [27, 21].

En términos matemáticos, la operación de convolución se puede expresar a través de la ecuación 22, la cual detalla cómo se calcula una posición de la convolución entre dos señales o funciones discretas x y w.

$$net(t, j) = (x * w)[t, j] = \sum_n \sum_m x[n, m]w[t - n, j - m] \tag{22}$$

Donde $net(t, j)$ es la salida de la siguiente capa, x es la imagen de entrada y w es el *kernel* o matriz filtro. La Fig. 15 muestra cómo funciona la capa

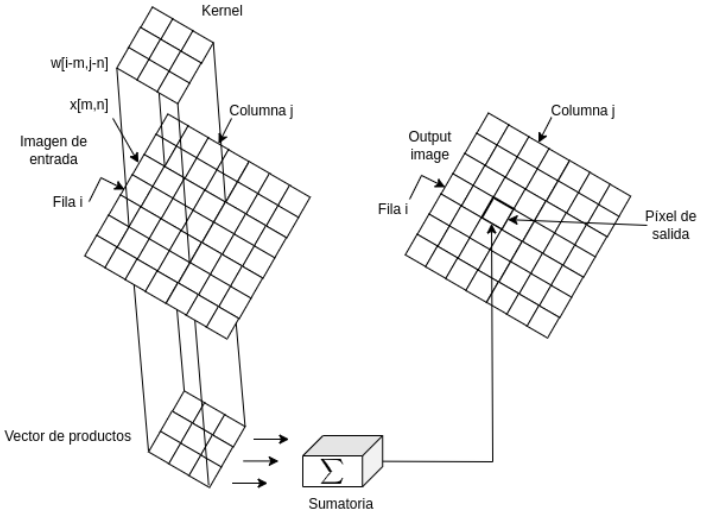


Figura 15: Detalles de la capa convolucional.

convolucional. Como puede verse, se realiza el producto punto a punto entre la imagen de entrada y el *kernel*, y el resultado se suma [27].

2.4.2 No linealidad

La siguiente capa, luego de la convolución, es de no linealidad. La no linealidad puede ser usada para ajustar o cortar las salidas generadas. La justificación fundamental para el uso de funciones de activación no lineales en redes neuronales reside en su capacidad para modelar relaciones no lineales complejas presentes en los datos de entrada. La Fig. 16 muestra los tipos comunes de no linealidad. Por muchos años las funciones sigmoide y tangente hiperbólica fueron las más populares entre las no lineales. Sin embargo, recientemente, la ReLu (*Rectified Linear Unit*) ha sido usada con mayor frecuencia debido a las siguientes ventajas:

1. ReLu tiene definiciones simples en su función y gradiente.

$$ReLu(x) = max(0, x) \tag{23}$$

$$\frac{d}{dx}ReLu(x) = \{1 \ si \ x > 0, \ 0 \text{ de otra manera}\} \tag{24}$$

2. Las funciones de saturación tales como sigmoide y tangente hiperbólica generan problemas en la retropropagación. Mientas la red neuronal diseñada sea más profunda, la señal del gradiente comienza a desaparecer,

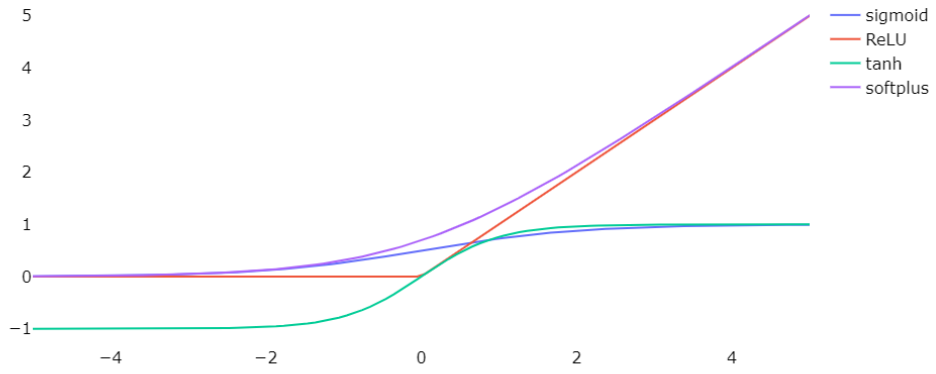


Figura 16: Tipos comunes de no linealidad.

lo que es conocido como problema de desvanecimiento del gradiente. Esto ocurre cuando el gradiente de estas funciones es cercano a cero. Sin embargo, ReLu tiene un gradiente constante para entradas positivas.

3. ReLu crea una representación más dispersa, es decir, puede generar como salida un valor cero verdadero. En cambio las funciones sigmoide y tangente hiperbólica siempre tienen resultados distintos de cero en su gradiente, que puede no ser favorable para el entrenamiento [27].

2.4.3 Submuestreo (Pooling)

La idea principal del submuestreo es reducir la complejidad de las capas adicionales. En el dominio del procesamiento de imágenes, puede considerarse similar a reducir la resolución. Esta reducción en la complejidad no afecta el número de filtros. *Max-pooling* es uno de los tipos más comunes de métodos de submuestreo. Divide la imagen en subregiones y solo devuelve el valor máximo del interior de cada subregión. Uno de los tamaños más comunes utilizados en el submuestreo es 2×2 con el objetivo de reducir a la mitad la cantidad de filas y columnas. Como se puede ver en la Fig. 17, cuando la agrupación se realiza en los bloques 2×2 de la parte superior izquierda (área rosa), se mueve 2 y se enfoca en la parte superior derecha. Esto significa que el paso 2 se usa en la agrupación. Para evitar el submuestreo, se puede usar el paso 1, lo cual no es común. Se debe considerar que el submuestreo no preserva la posición de la información. Por lo tanto, debe aplicarse cuando la

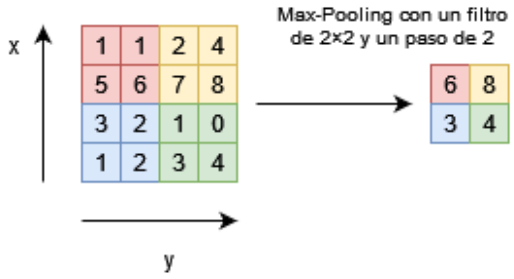


Figura 17: Demostración del submuestreo. El submuestreo con un filtro de 2×2 y un paso de 2 lleva a bajar la muestra de cada bloque de 2×2 mapeando a un bloque de 1.

presencia de información es importante (en lugar de información espacial). Además, la agrupación se puede utilizar con filtros y pasos diferentes para mejorar la eficiencia. Por ejemplo, una agrupación máxima (*max-pooling*) de 3×3 con 2 pasos mantiene algunas superposiciones entre las áreas [29, 27].

2.4.4 Capa totalmente conectada

Los mapas de características de salida de la convolución final o capa de agrupación generalmente se aplanan, es decir, se transforman en una matriz unidimensional (1D) de números (o vector) y se conectan a una o más capas completamente conectadas, también conocidas como capas densas, en el que cada entrada está conectada a cada salida por un peso aprendible, ver Fig. 18. Una vez que se crean las características extraídas por las capas de convolución y submuestreadas por las capas de agrupación, un subconjunto de capas completamente conectadas las asigna a los resultados finales de la red, como las probabilidades para cada clase en las tareas de clasificación. La capa final totalmente conectada normalmente tiene el mismo número de nodos de salida que el número de clases. A cada capa completamente conectada le sigue una función no lineal, por ejemplo la ReLu, como se describió anteriormente [25].

2.5 Entrenamiento de una red convolucional

En este apartado, se desarrollan los conceptos relacionados con el entrenamiento de una red neuronal convolucional que se aplicarán en secciones posteriores.

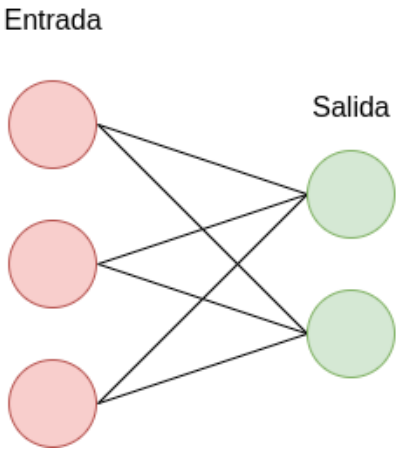


Figura 18: Capa totalmente conectada.

2.5.1 Aprendizaje supervisado

Sutskever, I [30] define el problema del aprendizaje supervisado de la siguiente manera, sea X un espacio de entrada, Y un espacio de salida y D la distribución de datos sobre $X \times Y$ que describe los datos que tendemos a observar. Por cada punto (x, y) de D, la variable x es una entrada típica, mientras que y es la salida deseada correspondiente. El objetivo del aprendizaje supervisado es usar un conjunto de entrenamiento que consta de n muestras independientes e idénticamente distribuidas, $S = \{(x_i, y_i)\}_{i=1}^{n} \sim D^n$ para encontrar la función $f : X \to Y$ cuyo error de prueba sea lo más bajo posible.

$$Prueba_D(f) \equiv E_{(x,y)\sim D}[L(f(x); y)] \tag{25}$$

Aquí $L(z; y)$ es una función de pérdida que mide la diferencia cada vez que predecimos y como z. Una vez que encontramos una función cuyo error de prueba es lo suficientemente pequeño para nuestras necesidades, el problema del aprendizaje está resuelto.

$$f^* = \underset{f\,es\,una\,función}{argmin}\ Prueba_D(f) \tag{26}$$

Aunque lo ideal sería encontrar el mínimo global del error de prueba hacerlo es fundamentalmente imposible.

$$Entrenamiento_S(f) \equiv E_{(x,y)\sim S}[L(f(x); y)] \approx Prueba_D(f) \tag{27}$$

Podemos aproximar el error de prueba con el error de entrenamiento (donde definimos S como la distribución uniforme sobre casos de entrenamiento

contando casos duplicados múltiples veces) y encontrar una función f con un error de entrenamiento bajo, pero es trivial minimizar el error de entrenamiento memorizando los casos de entrenamiento. Asegurarse de que el buen desempeño en el conjunto de entrenamiento se traduzca en un buen desempeño en el conjunto de prueba se conoce como el problema de generalización, que resulta ser conceptualmente fácil de resolver restringiendo las funciones permitidas f a una clase relativamente pequeña de funciones F:

$$f^* = \underset{f \in F}{argmin}\, Entrenamiento_S(f) \tag{28}$$

Restringir f a F esencialmente resuelve el problema de generalización, porque se puede demostrar que cuando $\log |F|$ es pequeño en relación con el tamaño del conjunto de entrenamiento, el error de entrenamiento es cercano a el error de prueba para todas las funciones $f \in F$ simultáneamente. Esto nos permite centrarnos en el problema algorítmico de minimizar el error de entrenamiento mientras se está razonablemente seguro de que el error de prueba también se minimizará aproximadamente. El costo de restringir f a F es que el mejor error de prueba alcanzable puede ser inadecuadamente alto para nuestras necesidades. Dado que el tamaño necesario del conjunto de entrenamiento crece con F, queremos que F sea lo más pequeño posible. Al mismo tiempo, queremos que F sea lo más grande posible para mejorar el desempeño de su mejor función. En la práctica, es sensato elegir el conjunto de funciones F más grande posible que pueda soportar el tamaño del conjunto de entrenamiento y el cálculo disponible. Desafortunadamente, no existe una receta general para elegir un buen conjunto F para un problema de aprendizaje automático dado. El mejor F teóricamente consiste en una sola función que logra el mejor error de prueba entre todas las funciones posibles, pero nuestra ignorancia de esta función es la razón por la que estamos interesados en aprender en primer lugar. Cuanto más sepamos sobre el problema y sobre sus funciones de alto rendimiento, más podremos restringir F estando razonablemente seguros de que contiene al menos una buena función. En la práctica, es mejor experimentar con clases de funciones que sean similares a las que son exitosas para problemas relacionados.

2.5.2 Optimización

Una vez que hemos elegido una F adecuada y recopilado un conjunto de entrenamiento lo suficientemente grande, nos enfrentamos al problema de encontrar una función $f \in F$ que tenga un error de entrenamiento bajo [30]. La tarea de de minimizar o maximizar alguna función $f(x)$ alterando x se conoce como optimización.

Los algoritmos de optimización que usan todo el conjunto de entrenamiento se denominan métodos de gradiente determinista, porque procesan todos los ejemplos de entrenamiento simultáneamente en un lote grande. Mientras que los algoritmos que utilizan lotes de datos para realizar el entrenamiento se conocen como métodos por lotes. Es muy común usar el término tamaño de lote (*batch size*) para describir el tamaño de un mini-lote.

En este apartado se presentan algunos optimizadores desarrollados para el entrenamiento de redes convolucionales.

2.5.2.1 Stochastic Gradient Descent

SGD (*Stochastic Gradient Descent*) y sus variantes son probablemente los algoritmos de optimización más utilizados para el aprendizaje automático en general y para el aprendizaje profundo en particular. Es posible obtener una estimación imparcial del gradiente tomando el gradiente promedio en un mini lote de m ejemplos extraídos independientes e idénticamente distribuidos de la distribución de generación de datos [28].

Un parámetro crucial para el algoritmo SGD es la tasa de aprendizaje o *learning rate*. La tasa de aprendizaje es un parámetro de ajuste en un algoritmo de optimización que determina el tamaño del paso en cada iteración mientras se mueve hacia un mínimo de una función de pérdida. En la práctica, es necesario disminuir gradualmente la tasa de aprendizaje con el tiempo, por lo que ahora denotamos la tasa de aprendizaje en la iteración k como $\in_k$.

Esto se debe a que el estimador de gradiente SGD introduce una fuente de ruido (el muestreo aleatorio de m ejemplos de entrenamiento) que no desaparece incluso cuando llegamos a un mínimo. En comparación, el verdadero gradiente de la función de costo total se vuelve pequeño y luego 0 cuando nos acercamos y alcanzamos un mínimo usando el descenso del gradiente por lotes, por lo que el descenso del gradiente por lotes puede usar una tasa de aprendizaje fija. Una condición suficiente para garantizar la convergencia de SGD es que

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \tag{29}$$

y

$$\sum_{k=1}^{\infty} \epsilon_k^2 < \infty \tag{30}$$

En la práctica, es común disminuir la tasa de aprendizaje linealmente hasta la iteración τ:

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \tag{31}$$

con $\alpha = \frac{k}{\tau}$. Después de la iteración τ, es común dejar $\in$ constante.

La tasa de aprendizaje puede elegirse por prueba y error, pero generalmente es mejor elegirla monitoreando las curvas de aprendizaje que trazan la función objetivo como una función del tiempo. Esto es más un arte que una ciencia, y la mayoría de las guías sobre este tema deben considerarse con cierto escepticismo. Cuando se utiliza el programa lineal, los parámetros a elegir son $\in_0$, $\in_\tau$ y τ .Por lo general, τ se puede establecer en el número de iteraciones necesarias para realizar unos cientos de pasadas a través del conjunto de entrenamiento. Por lo general, $\in_\tau$ debe establecerse en aproximadamente el 1 % del valor de $\in_0$. La pregunta principal es cómo establecer $\in_0$. Si es demasiado grande, la curva de aprendizaje mostrará oscilaciones violentas, y la función de costo a menudo aumentará significativamente. Si la tasa de aprendizaje es baja, el aprendizaje avanza lentamente, y si la tasa de aprendizaje inicial es demasiado baja, el aprendizaje puede quedarse estancado con un valor de costo alto. Por lo general, la tasa de aprendizaje inicial óptima, en términos de tiempo total de entrenamiento y el valor del costo final, es más alta que la tasa de aprendizaje que produce el mejor rendimiento después de las primeras 100 iteraciones más o menos. Por lo tanto, generalmente es mejor monitorear las primeras iteraciones y usar una tasa de aprendizaje que sea más alta que la tasa de aprendizaje de mejor desempeño en este momento, pero no tan alta como para causar una inestabilidad severa.

2.5.2.2 Momentum

Si bien el descenso de gradiente estocástico sigue siendo una estrategia de optimización muy popular, aprender con ella a veces puede ser lento. El método de *momentum* [31] está diseñado para acelerar el aprendizaje, especialmente frente a curvaturas altas, gradientes pequeños pero constantes o gradientes ruidosos. El algoritmo de *momentum* acumula un promedio móvil exponencialmente decreciente de gradientes pasados y continúa moviéndose en su dirección. El efecto del *momentum* se ilustra en la Fig 19.

2.5.2.3 AdaGrad

El algoritmo AdaGrad (*Adaptive Gradient Algorithm*) adapta individualmente las tasas de aprendizaje de todos los parámetros del modelo al escalarlos de manera inversamente proporcional a la raíz cuadrada de la suma de todos sus valores cuadrados históricos [32]. Los parámetros con la mayor derivada

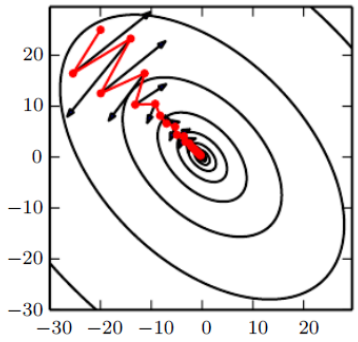


Figura 19: Las líneas de contorno representan una función de pérdida cuadrática. El camino rojo que atraviesa los contornos indica el camino seguido por la regla de aprendizaje de impulso, ya que minimiza esta función. En cada paso del camino, dibujamos una flecha que indica el paso que daría el descenso de gradiente en ese punto. Podemos ver que un objetivo cuadrático mal condicionado parece un valle o cañón largo y angosto con lados empinados. El impulso atraviesa correctamente el cañón a lo largo, mientras que los pasos de gradiente pierden tiempo moviéndose hacia adelante y hacia atrás a través del eje angosto del cañón [28].

parcial de la pérdida tienen una disminución correspondientemente rápida en su tasa de aprendizaje, mientras que los parámetros con pequeñas derivadas parciales tienen una disminución relativamente pequeña en su tasa de aprendizaje. El efecto neto es un mayor progreso en las direcciones de pendiente más suave del espacio de parámetros.

2.5.2.4 RMSProp

El algoritmo RMSProp (*Root Mean Square Propagation*) modifica AdaGrad para que funcione mejor en la configuración no convexa al cambiar la acumulación de gradiente en un promedio móvil exponencialmente ponderado [28]. AdaGrad está diseñado para converger rápidamente cuando se aplica a una función convexa. Cuando se aplica a una función no convexa para entrenar una red neuronal, la trayectoria de aprendizaje puede pasar por muchas estructuras diferentes y finalmente llegar a una región que es un cuenco localmente convexo. AdaGrad reduce la tasa de aprendizaje de acuerdo con el historial completo del gradiente cuadrático y puede haber hecho que la tasa de aprendizaje sea demasiado pequeña antes de llegar a una estructura tan convexa. RMSProp utiliza un promedio exponencialmente decreciente para descartar la historia del pasado extremo para que pueda converger rápida-

mente después de encontrar un cuenco convexo, como si fuera una instancia del algoritmo AdaGrad inicializado dentro de ese cuenco.

2.5.2.5 Adam

Adam es otro algoritmo de optimización de la tasa de aprendizaje adaptativo [33]. El nombre "Adam" deriva de la frase "momentos adaptativos". En el contexto de los algoritmos anteriores, quizás se vea mejor como una variante de la combinación de RMSProp e impulso con algunas distinciones importantes. Primero, en Adam, el impulso se incorpora directamente como una estimación del impulso de primer orden (con ponderación exponencial) del gradiente. La forma más sencilla de agregar impulso a RMSProp es aplicar impulso a los gradientes reescalados. El uso del impulso en combinación con el reescalado no tiene una motivación teórica clara. En segundo lugar, Adam incluye correcciones de sesgo a las estimaciones de los impulsos de primer orden (el término de cantidad de movimiento) y los impulsos de segundo orden (no centrados) para tener en cuenta su inicialización en el origen (ver algoritmo implementado por Goodfellow et al. [28]). RMSProp también incorpora una estimación del impulso de segundo orden (no centrado), sin embargo, carece del factor de corrección. Por lo tanto, a diferencia de Adam, la estimación del impulso de segundo orden de RMSProp puede tener un alto sesgo al principio del entrenamiento. En general, se considera que Adam es bastante robusto en la elección de hiperparámetros, aunque a veces es necesario cambiar la tasa de aprendizaje del valor predeterminado sugerido.

2.6 Redes Neuronales Residuales

En este trabajo, adoptamos la última generación de la ResNet (*Residual Neural Network*) introducida por He et al. [34], que ocupa el puesto número uno en el desafío de reconocimiento visual a gran escala de ImageNet 2016 [35] para la extracción de características. ResNet adopta el aprendizaje residual a algunas capas apiladas. En la Fig. 20 se muestra un bloque de construcción. Formalmente, podemos considerar un bloque de construcción definido como:

$$y = F(x, W_i) + x \tag{32}$$

Aquí x e y son los vectores de entrada y salida de la capa considerada. La función $F(x, W_i)$ representa el mapeo residual que debe aprender. Por ejemplo en la Fig. 20 que tiene dos capas $F = W_2\sigma(W_1x)$ en donde $\sigma(\cdot)$ denota ReLu [36] y los *bias* se omiten para simplificar las notaciones. La operación $F+x$ se realiza mediante una conexión de acceso directo y una adición de elementos. Adoptamos la segunda no linealidad después de la adición (es decir, $\sigma(y)$,

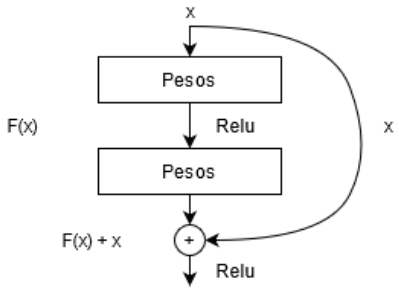


Figura 20: Aprendizaje residual. Un bloque de construcción.

ver Fig. 20). Las conexiones de acceso directo en la ecuación (32) no introducen ningún parámetro adicional ni complejidad de cálculo. Esto no solo es atractivo en la práctica, sino que también es importante en nuestras comparaciones entre redes simples y residuales. Podemos comparar de manera justa redes simples / residuales que simultáneamente tienen el mismo número de parámetros, profundidad, ancho y costo computacional (excepto por la suma insignificante de elementos). Las dimensiones de x y F deben ser iguales en la ecuación (32). Si este no es el caso (por ejemplo, al cambiar los canales de entrada / salida), podemos realizar una proyección lineal W_s mediante las conexiones de acceso directo para que coincidan con las dimensiones:

$$y = F(x, W_i) + W_s x \tag{33}$$

También observamos que, aunque las notaciones anteriores se refieren a capas completamente conectadas por simplicidad, son aplicables a capas convolucionales. La función $F(x, W_i)$ puede representar múltiples capas convolucionales. La adición de elementos se realiza en dos mapas de características, canal por canal. He et al. [34] presentaron distintas arquitecturas ResNet en función de la cantidad de capas que tiene cada bloque convolucional. En este trabajo utilizamos la ResNet de 50 capas (ResNet-50) que cuenta con cinco bloques convolucionales de 3 capas. El Cuadro 1 contiene el detalle de la arquitectura ResNet-50 utilizada para el trabajo.

2.7 Vectores de Fisher

El cálculo de los Vectores de Fisher se realizó de acuerdo a lo planteado por los autores Sánchez et al. [37]. Sea $X = \{x_t, t = 1...T\}$ una muestra de T observaciones $x_t \in \mathbb{R}^D$. Sea u_λ una función de densidad de probabilidad que modela el proceso generativo de elementos en $\mathbb{R}^D$ donde $\lambda = [\lambda_1, ..., \lambda_M]' \in \mathbb{R}^M$ denota el vector de M parámetros de u_λ. En estadística, la función de

Cuadro 1: Arquitectura de ResNet-50.

Capa/Bloque Residual	Parámetros (*kernels*, canales)
Conv1	$7 \times 7, 64$
Agrupación máxima (*max-pooling*), 3×3, 64	
Conv2_x	Bloque Residual 1-3 $\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv3_x	Bloque Residual 4-7 $\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
Conv4_x	Bloque Residual 8-13 $\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
Conv5_x	Bloque Residual 14-16 $\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
Agrupación promedio (*average pooling*), 7×7, 2048	
Softmax completamente conectado , 1×1, 1000	

puntuación viene dada por el gradiente de la probabilidad logarítmica de los datos en el modelo:

$$G_\lambda^X = \nabla_\lambda \log u_\lambda(X) \tag{34}$$

Este gradiente describe la contribución de los parámetros individuales al proceso generativo. En otras palabras, describe cómo los parámetros del modelo generativo u_λ deben modificarse para ajustarse mejor a los datos X. Observamos que $G_\lambda^X \in \mathbb{R}^M$ y, por lo tanto, la dimensionalidad de G_λ^X solo depende del número de parámetros M en λ y no del tamaño de muestra T. A partir de la teoría de la geometría de la información [38], una familia paramétrica de distribuciones $U = u_\lambda, \lambda \in \Lambda$ puede considerarse como M_Λ, una variedad de Riemanninan, con una métrica local dada por la FIM (*Fisher Information Matrix*) $F_\lambda \in \mathbb{R}^{M \times M}$:

$$F_\lambda = E_{x \sim u_\lambda}[G_\lambda^X G_\lambda^{X'}] \tag{35}$$

Siguiendo esta observación, Jaakkola et al. [39] propusieron medir la similitud entre dos muestras X e Y utilizando el Fisher Kernel (FK) que se define como:

$$K_{FK}(X, Y) = G_\lambda^{X'} F_\lambda^{-1} G_\lambda^Y \tag{36}$$

Dado que F_λ es semi-definido positivo, también lo es su inverso. Usando la descomposición de Cholesky $F_\lambda^{-1} = L_\lambda' L_\lambda$, el FK en 36 se puede reescribir

explícitamente como un producto escalar:

$$K_{FK}(X,Y) = \mathcal{G}_\lambda^{X'} \mathcal{G}_\lambda^Y \tag{37}$$

Para implementar la codificacion de FV (*Fisher Vectors*) a partir de los descriptores de una red convolucional es necesario especificar una distribucion de probabilidades de los descriptores. Para tal fin se utiliza el GMM (*Gaussian Mixture Model*) ya que es posible aproximar con precisión arbitraria cualquier distribución continua con un GMM [40]. Sea $\lambda = \{w_k, \mu_k, \Sigma_k, k = 1, ..., K\}$ los parámetros de k componentes de GMM donde w_k, μ_k y Σ_k son respectivamente el peso de la mezcla, el vector medio y la matriz de covarianza del k-ésimo Gaussiano.

$$\mu_\lambda(x) = \sum_{k=1}^{K} w_k \mu_k(x) \tag{38}$$

donde μ_k denota al Gaussiano k:

$$\mu_k(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)\right\} \tag{39}$$

y es necesario que:

$$\forall k : w_k \geq 0, \sum_{k=1}^{K} w_k = 1 \tag{40}$$

para asegurarse de que $\mu_\lambda(x)$ sea una distribución válida. En lo que sigue, asumimos matrices de covarianza diagonales que es una suposición estándar y denotamos por σ_k^2 al vector de varianza, es decir, la diagonal de Σ_k. Estimamos los parámetros de GMM en un gran conjunto de entrenamiento de descriptores locales utilizando el algoritmo EM (*Expectation Maximization*) para optimizar un criterio de máxima verosimilitud. Para obtener más detalles sobre la implementación de GMM, el lector puede consultar Sánchez et al. [37]. Para los parámetros de peso, adoptamos el formalismo propuesto por Krapac et al. [41] y definimos:

$$w_k = \frac{\exp(\sigma_k)}{\sum_{j=1}^{K} \exp(\sigma_j)} \tag{41}$$

La re-parametrización usando σ_k evita hacer cumplir explícitamente las restricciones en la Ecuación 40. Los gradientes de un solo descriptor x_t con respecto a los parámetros del modelo GMM, $\lambda = \{\alpha_k, \mu_k, \Sigma_k, k = 1, ..., K\}$, son:

$$\nabla_{\alpha_k} \log u_\lambda(x_t) = \gamma_t(k) - w_k \tag{42}$$

$$\nabla_{\mu_k} \log u_\lambda(x_t) = \gamma_t(k) \left(\frac{x_t - \mu_k}{\sigma_k^2} \right) \tag{43}$$

$$\nabla_{\sigma_k} \log u_\lambda(x_t) = \gamma_t(k) \left[\frac{(x_t - \mu_k)^2}{\sigma_k^3} - \frac{1}{\sigma_k} \right] \tag{44}$$

donde $\gamma_t(k)$ es la asignación suave de x_t, para el k Gaussiano, que también es conocido como su probabilidad posterior:

$$\gamma_t(k) = \frac{w_k u_k(x_t)}{\sum_{j=1}^{K} w_j u_j(x_t)} \tag{45}$$

donde la división y exponenciación de vectores debe entenderse como operaciones término por término.

La diagonal de FIM se puede tener en cuenta mediante una normalización coordinada de los vectores de gradiente, que produce los siguientes gradientes normalizados:

$$\mathcal{G}_{\alpha_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} (\gamma_t(k) - w_k) \tag{46}$$

$$\mathcal{G}_{\mu_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{x_t - \mu_k}{\sigma_k} \right) \tag{47}$$

$$\mathcal{G}_{\sigma_k}^X = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_t(k) \frac{1}{\sqrt{2}} \left[\frac{(x_t - \mu_k)^2}{\sigma_k^2} - 1 \right] \tag{48}$$

Es importante mencionar que $\mathcal{G}_{\alpha_k}^X$ es un escalar mientras que $\mathcal{G}_{\mu_k}^X$ y $\mathcal{G}_{\sigma_k}^X$ son vectores de dimensionalidad D. El FV final es la concatenación de los gradientes $\mathcal{G}_{\alpha_k}^X$, $\mathcal{G}_{\mu_k}^X$ y $\mathcal{G}_{\sigma_k}^X$ para $k = 1, ..., K$ y por lo tanto su dimensión es $E = (2D + 1)K$.

Para evitar la dependencia del tamaño de la muestra (ver, por ejemplo, la normalización de la longitud de la secuencia por Smith et al. [42]), normalizamos el FV resultante por el tamaño de la muestra T, es decir, realizamos la siguiente operación:

$$\mathcal{G}_\lambda^X \leftarrow \frac{1}{T} \mathcal{G}_\lambda^X \tag{49}$$

En la práctica, T es casi constante en nuestros experimentos, ya que cambiamos el tamaño de todas las imágenes al mismo número de píxeles. También se debe tener en cuenta que las ecuaciones 46 - 48 se pueden calcular en términos de las siguientes estadísticas de orden 0, primer y segundo orden (ver Algoritmo 1):

$$S_k^0 = \sum_{t=1}^{T} \gamma_t(k) \tag{50}$$

$$S_k^1 = \sum_{t=1}^{T} \gamma_t(k) x_t \tag{51}$$

$$S_k^2 = \sum_{t=1}^{T} \gamma_t(k) x_t^2 \tag{52}$$

donde $S_k^0 \in \mathbb{R}$, $S_k^1 \in \mathbb{R}^D$ y $S_k^2 \in \mathbb{R}^D$. Como antes, el cuadrado de un vector debe entenderse como una operación término a término.

En el presente trabajo para calcular los Vectores de Fisher se utilizó el algoritmo 1. En dicho algoritmo se observan dos normalizaciones: la normalización de potencia y la normalización L_2. Perronnin et al. [43] presentaron estas normalizaciones, donde demostraron ser necesarias para obtener resultados competitivos cuando el FV se combina con un clasificador lineal. En los siguientes apartados se desarrollan algunos conceptos que se utilizaron con los Vectores de Fisher en este trabajo.

2.8 Modelo de mezcla gaussiana

Un modelo de mezcla gaussiana (GMM) es una función de densidad de probabilidad paramétrica representada como una suma ponderada de las densidades de los componentes gaussianos. Los parámetros GMM se estiman a partir de los datos de entrenamiento utilizando el algoritmo iterativo EM o la estimación MAP (*Maximum A Posteriori*) de un modelo previo bien entrenado [44].

Un modelo de mezcla gaussiana es una suma ponderada de M densidades gaussianas de componentes dadas por la ecuación

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i), \tag{53}$$

donde x es un vector de datos de valor continuo D-dimensional, w_i, $i = 1, ..., M$, son los pesos de la mezcla y $g(x|\mu_i, \Sigma_i)$, $i = 1, ..., M$ son las densidades gaussianas de los componentes. La densidad de cada componente es una función gaussiana de la forma

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right\} \tag{54}$$

Algoritmo 1: Cálculo de Vectores de Fisher a partir de descriptores [43].

Entrada:

- Descriptores locales de una imagen $X = \{x_t \in \mathbb{R}^D, t = 1, ..., T\}$
- Parámetros del modelo GMM $\lambda = \{w_k, \mu_k, \sigma_k, k = 1, ..., K\}$

Salida:

- Vector de Fisher normalizado $\mathcal{G}_\lambda^X \in \mathbb{R}^{k(2D+1)}$

1. Calcular estadísticos
   - Para $k = 1, ..., K$ inicializar acumuladores
     - $S_k^0 \leftarrow 0$, $S_k^1 \leftarrow 0$, $S_k^2 \leftarrow 0$
   - Para $t = 1, ..., T$:
     - Calcular $\gamma_t(k)$ usando la Ecuación 45
     - Para $k = 1, ...K$:
       - $S_k^0 \leftarrow S_k^0 + \gamma_t(k)$
       - $S_k^1 \leftarrow S_k^1 + \gamma_t(k)x_t$
       - $S_k^2 \leftarrow S_k^2 + \gamma_t(k)x_t^2$
2. Calcular los Vectores de Fisher
   - Para $k = 1, ..., K$:
     - $\mathcal{G}_{\alpha_k}^X = (S_k^0 - Tw_k)/\sqrt{w_k}$
     - $\mathcal{G}_{\mu_k}^X = (S_k^1 - \mu_k S_k^0)/(\sqrt{w_k}\sigma_k)$
     - $\mathcal{G}_{\sigma_k}^X = (S_k^2 - 2\mu_k S_k^1 + (\mu_k^2 - \sigma_k^2)S_k^0)/(\sqrt{2w_k}\sigma_k^2)$
   - Concatenar todos los componentes del vector de Fisher en un solo vector
     - $\mathcal{G}_\lambda^X = (\mathcal{G}_{\alpha_1}^X, ..., \mathcal{G}_{\alpha_K}^X, \mathcal{G}_{\mu_1}^X{}', ..., \mathcal{G}_{\mu_K}^X{}', \mathcal{G}_{\sigma_1}^X{}', ..., \mathcal{G}_{\sigma_K}^X{}')'$
3. Aplicar normalizaciones
   - Para $i = 1, ..., K(2D + 1)$ aplicar la normalización de potencia
     - $[\mathcal{G}_\lambda^X]_i \leftarrow sign([\mathcal{G}_\lambda^X]_i)\sqrt{|[\mathcal{G}_\lambda^X]_i|}$
   - Aplicar normalización L_2:
     - $\mathcal{G}_\lambda^X = \mathcal{G}_\lambda^X / \sqrt{\mathcal{G}_\lambda^X{}'\mathcal{G}_\lambda^X}$

con vector de medias μ_i y matriz de covarianza Σ_i. Los pesos de la mezcla satisfacen la restricción de que $\sum_{i=1}^{M} w_i = 1$. El modelo de mezcla gaussiana completo se parametriza mediante vectores medios, matrices de covarianza y pesos de mezclas de todas las densidades de los componentes. Estos parámetros están representados colectivamente por la notación:

$$\lambda = \{w_i, \mu_i, \Sigma_i\} \quad i = 1, ..., M \tag{55}$$

2.9 Análisis de componentes principales

PCA (*Principal Component Analysis*) es una técnica multivariada que analiza un conjunto de datos en el cual las observaciones se describen mediante varias variables dependientes cuantitativas interrelacionadas. Su objetivo es extraer la información más importante de los datos y representarla como un conjunto de nuevas variables ortogonales llamadas componentes principales [45].

Sea X, el conjunto de datos, una matriz de $m \times n$, donde m es el número de variables y n es el número de muestras. El objetivo se resume a encontrar alguna matriz ortonormal P en $Y = PX$ tal que $C_Y = \frac{1}{n} YY'$ sea una matriz diagonal donde C_Y es la matriz de covarianzas de Y. Las filas p_i de P son los componentes principales de X [46] que a su vez son los autovectores de $C_X = \frac{1}{n} XX'$, siendo C_X la matriz de covarianzas de X. El i-ésimo valor de la diagonal de C_Y es la varianza de X a lo largo de p_i.

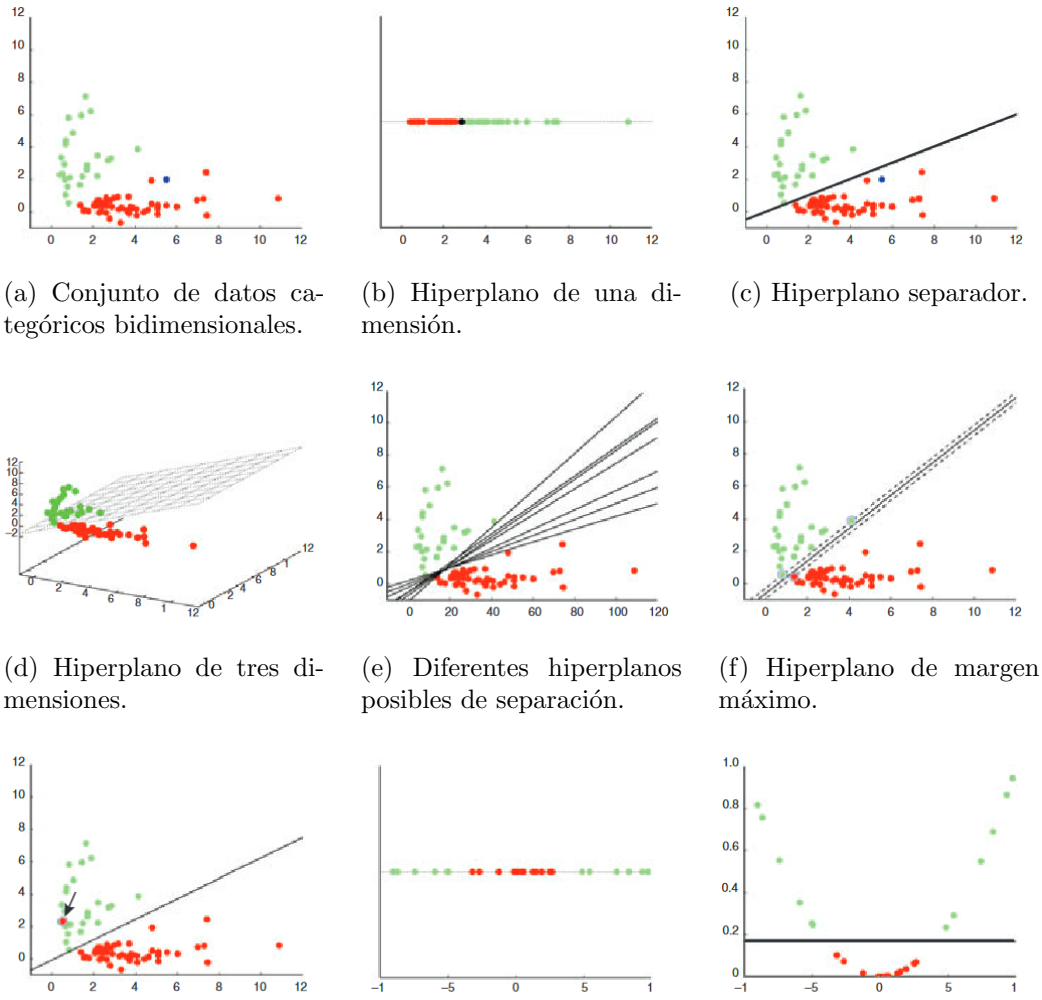
En la práctica, calcular PCA de un conjunto de datos X implica (1) restar la media de cada tipo de medición y (2) calcular los autovectores de C_X.

2.10 Máquina de vectores de soporte

SVM (*Support Vector Machines*) implementa la siguiente idea: mapea los vectores de entrada x en un espacio de características de alta dimensión Z a través de un mapeo no lineal, elegido a priori. En este espacio se construye un hiperplano separador óptimo [47]. A continuación se presentan las principales ideas detras del algoritmo SVM en los problemas de clasificación.

Para definir la noción de un hiperplano, se puede considerar un conjunto de datos unidimensional de dos categorías. En este caso podemos separar ambas categorías a partir de un punto (Fig. 21b). En dos dimensiones (Fig. 21c), una línea recta divide el espacio por la mitad, y en tres dimensiones, necesitamos un plano para dividir el espacio (Fig. 21d). Podemos extrapolar matemáticamente este procedimiento a dimensiones superiores. El término general para una línea recta en un espacio de alta dimensión es un hiperplano,

por lo que el hiperplano de separación es, esencialmente, la línea que separa las muestras de un conjunto de datos [48].



(a) Conjunto de datos categóricos bidimensionales.

(b) Hiperplano de una dimensión.

(c) Hiperplano separador.

(d) Hiperplano de tres dimensiones.

(e) Diferentes hiperplanos posibles de separación.

(f) Hiperplano de margen máximo.

(g) Hiperplano de separación con *soft margin*.

(h) Un conjunto de datos unidimensional no separable.

(i) Separación de datos previamente no separables.

Figura 21: Funcionamiento de la máquina de vectores de soporte [48].

Para encontrar el hiperplano que separa mejor las dos clases (Fig. 21e), SVM adopta la distancia máxima de cualquiera de las muestras de cada clase (Fig. 21f). Si definimos la distancia desde el hiperplano de separación al vector de expresión más cercano como el margen del hiperplano, entonces SVM selecciona el hiperplano de separación de margen máximo. La selección

de este hiperplano en particular maximiza la capacidad de SVM para predecir la clasificación correcta de ejemplos nunca antes vistos.

En el caso de los datos que no se pueden separar de manera lineal de forma total, el algoritmo SVM debe modificarse agregando un *soft margin* (Fig. 21g). Esencialmente, esto permite que algunos puntos de datos se abran camino a través del margen del hiperplano de separación sin afectar el resultado final. Sin embargo, este procedimiento no es suficiente cuando al incorporar un *soft margin* no es posible separar las clases como en el ejemplo de la figura (Fig. 21h). Debido a esto, se incorpora el concepto de función *kernel* en SVM. En esencia, la función *kernel* permite que SVM realice una clasificación "bidimensional"de un conjunto de datos originalmente unidimensionales. En general, una función *kernel* proyecta datos desde un espacio de baja dimensión a un espacio de mayor dimensión (Fig. 21i). El objetivo de esta proyección es poder separar las clases en la nueva dimensión.

3 Desarrollo y experimentación

A lo largo de esta sección se detallará la metodología empleada para abordar los objetivos del proyecto y se describen los experimentos realizados para la detección de melanomas. En primer lugar, se presentan los datos utilizados y la infraestructura informática empleada en la experimentación. A continuación, se explican las métricas adoptadas para evaluar los resultados de los experimentos. Finalmente, se describe el diseño experimental, que incluye el uso de descriptores de imágenes, la red convolucional ResNet-50 y los Vectores de Fisher. Para cada caso, se especifican los procedimientos realizados y los parámetros utilizados.

3.1 Conjunto de datos

En el presente trabajo se utilizó el conjunto de datos de la competencia ISIC 2016 [49]. El objetivo del conjunto de datos es apoyar la investigación y el desarrollo de algoritmos para el diagnóstico automatizado de melanoma a partir de imágenes dermatoscópicas. El desafío se dividió en sub-desafíos para cada tarea involucrada en el análisis de imágenes, incluida la segmentación de lesiones, la detección de características dermatoscópicas dentro de una lesión y la clasificación del melanoma. El estudio realizado solo se enfocará en el último desafío, es decir, la clasificación de melanomas.

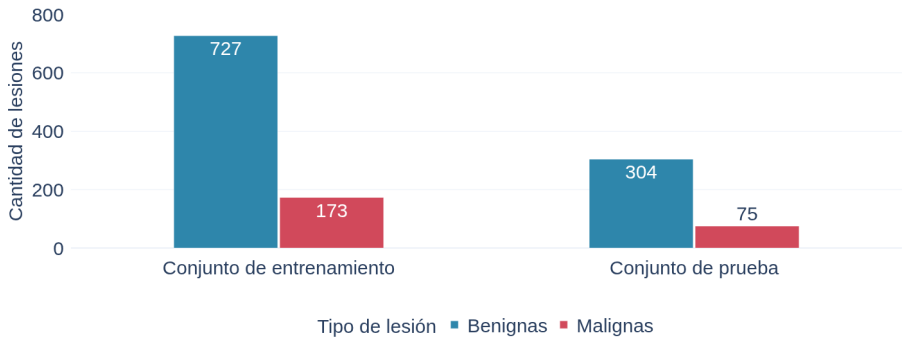


Figura 22: Conjunto de datos.

Este conjunto de datos se compone de 900 imágenes de entrenamiento y 379 imágenes de prueba. El primer conjunto se divide en 727 imágenes de lesiones benignas y 173 melanomas. En tanto que el conjunto de prueba se conforma a partir de 304 lesiones benignas y 75 melanomas. En ambos casos el conjunto de datos fue generado a partir de datos reales por un panel de

expertos en dermatoscopia. En la Fig. 22 se presenta la distribución del conjunto de datos. Para cada una de las imágenes de lesión de piel se dispone además, de una mascara binaria asociada a la región de la lesión en formato PNG. Las imágenes del conjunto de datos de la competencia ISIC 2016 tiene un amplio rango de tamaño, desde la imagen más chica de 722×542 píxeles, hasta la más grande de 4288×2848 píxeles. La Fig. 23 muestra ejemplos comparativos de lesiones benignas y malignas en imágenes dermatoscópicas, mientras que la Fig. 24 incluye ejemplos de lesiones de piel con sus correspondientes máscaras binarias.

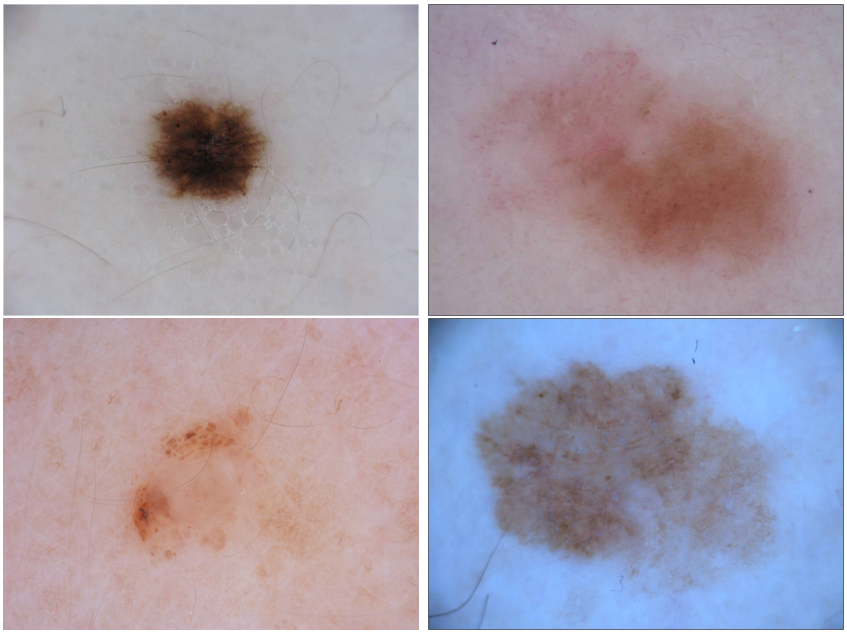


Figura 23: Ejemplos de imágenes dermatoscópicas de lesiones de piel. Las dos imágenes de la izquierda corresponden a lesiones benignas y las dos imágenes de la derecha corresponden a lesiones malignas.

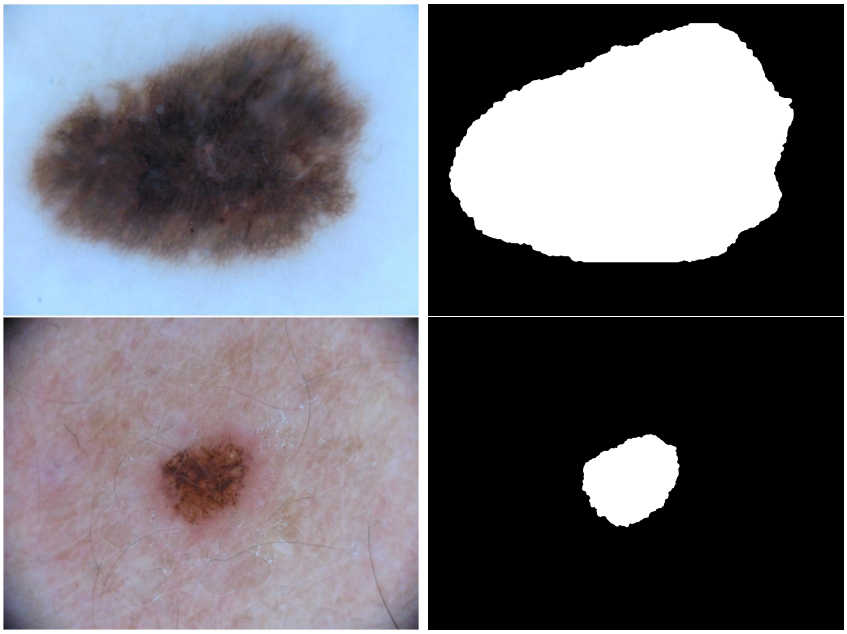


Figura 24: Ejemplos de imágenes dermatoscópicas de lesiones de piel y su correspondiente máscara binaria, utilizadas para la extracción de características de textura mediante LBP y HOG (ver Sección 3.4.1 para detalles del método).

3.2 Infraestructura utilizada

La infraestructura donde se realizó el trabajo tiene las siguientes características:

- *Hardware*:
  - RAM: 62 Gb
  - CPU: Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz
  - GPU: GP102 - GeForce GTX 1080 Ti
- Software:
  - Sistema operativo: Ubuntu - 18.04.5 LTS (Bionic Beaver)
  - Lenguaje de programación: Python 3.6.8
- Librerías de Python utilizadas:
  - numpy
  - opencv-python
  - pandas
  - requests
  - tensorflow
  - keras
  - scikit-learn
  - pillow

3.3 Métricas de evaluación

Antes de mencionar las métricas que se utilizaron en el presente trabajo para evaluar los modelos, se describen las métricas *Precision* y *Recall* ya que a partir de ellas se basan varias de las que se usaron para evaluar los modelos.

- *Precision* P se define como el número de verdaderos positivos T_P sobre el número de verdaderos positivos más el número de falsos positivos F_P.

$$P = \frac{T_P}{T_P + F_P} \tag{56}$$

- *Recall* R se define como el número de verdaderos positivos T_P sobre el número de verdaderos positivos más el número de falsos negativos F_N

$$R = \frac{T_P}{T_P + F_N} \tag{57}$$

A continuación se presentan las métricas utilizadas para evaluar la precisión de los modelos entrenados:

- mAP (*Mean Average Precision*). La precisión media resume una curva de *Precision-Recall* como la media ponderada de las precisiones logradas en cada umbral, con el aumento en el *Recall* desde el umbral anterior utilizado como peso:

$$mAP = \sum_{i=1}^{n} (R_i - R_{i-1}) P_i \tag{58}$$

  donde R_i es el *Recall* o sensibilidad o tasa de verdaderos positivos para el umbral i y P_i es la *Precision* para el umbral i. En la sumatoria n representa la cantidad de umbrales considerados para crear la curva de *Precision-Recall*.

- *F1 Score*. La métrica *F1 Score* se puede interpretar como una media armónica de *Precision* P y *Recall* R, donde una puntuación *F1 Score* alcanza su mejor valor en 1 y su peor puntuación en 0. La fórmula para el *F1 Score* es la siguiente:

$$F1 = \frac{2 * (P * R)}{P + R} \tag{59}$$

- Exactitud (*Accuracy*). Para los problemas de clasificación binaria, el *Accuracy* es la fracción de predicciones de clase correctas.

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{60}$$

  donde T_P son los verdaderos positivos, T_N los verdaderos negativos, F_P los falsos positivos y F_N los falsos negativos.

- AUC (*Area Under the ROC Curve*). El área bajo la curva ROC se utiliza para problemas de clasificación binaria. La curva ROC es la curva de $sensibilidad$ frente a $1 - especificidad$ en varios umbrales de

predicción [50]. La *sensibilidad* es la facción entre los casos positivos correctamente clasificados sobre el total de casos positivos de la muestra. Por otro lado, la $especificidad$ mide la proporción de los casos negativos correctamente clasificados sobre el total de casos negativos de la muestra:

$$especificidad = \frac{T_N}{T_N + F_P} \tag{61}$$

$$sensibilidad = \frac{T_P}{T_P + F_N} \tag{62}$$

donde T_P son los verdaderos positivos, T_N son los verdaderos negativos, F_P son los falsos positivos y F_N son los falsos negativos.

- Entropía cruzada binaria (*Binary cross entropy*). Para el entrenamiento de la red ResNet-50 también se consideró a la entropía cruzada binaria como una de las principales métricas. Esta es la función de pérdida utilizada en la regresión logística y sus extensiones, como las redes neuronales, definida como la verosimilitud logarítmica negativa de un modelo logístico que devuelve probabilidades p para sus datos de entrenamiento y. Para un conjunto de n elementos se puede definir a la *Binary cross entropy* de la siguiente manera:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \tag{63}$$

donde $y_i \in \{0, 1\}$ es la clase de la instancia i y p_i la probabilidad de la instancia i.

3.4 Clasificación con descriptores de imágenes

En este apartado se describen los experimentos realizados utilizando Vectores de Fisher generados a partir de descriptores de imágenes. Primero, se detalla el procedimiento empleado para generar los Vectores de Fisher a partir de cada uno de los descriptores de imágenes seleccionados. A continuación, se presentan los experimentos realizados, comenzando con el uso del conjunto de datos original, seguido por experimentos con un balanceo de clases. Por último, se incluyen pruebas adicionales utilizando descriptores generados mediante el método LBP.

3.4.1 Procedimiento para generar descriptores y obtener Vectores de Fisher

Los primeros modelos a desarrollar consistieron en clasificadores SVM que utilizaron Vectores de Fisher generados a partir de descriptores de imágenes GLCM, LBP y HOG. El proceso para obtener los descriptores utilizando estos algoritmos depende del procesamiento de imagen requerido para cada uno de ellos. A continuación se describe el trabajo realizado con cada descriptor y más adelante se detallan los experimentos realizados con estos procedimientos.

El proceso utilizado con GLCM se ilustra en la Fig. 25a, donde en primer lugar se transforma la imagen en escala de grises y se selecciona la región de la imagen que contiene a la lesión de piel. Luego se obtiene el histograma de co-ocurrencia de grises $P \in \mathbb{R}^{H \times W \times D \times \Theta}$ donde $P_{i,j,d,\theta}$ es el número de veces que el nivel de gris j ocurre a distancia d y ángulo θ del nivel de gris i, con $0 \leq i < H$ y $0 \leq j < W$. Aquí H y W hacen referencia al alto y ancho de la imagen en cantidad de píxeles. Para el cálculo del histograma de co-ocurrencias se utilizaron distancias de $D = 50$ píxeles mientras que los ángulos en radianes fueron $0, \pi/4, \pi/2$ y $3\pi/4$, es decir, $\Theta = 4$. Los valores donde el nivel de gris es igual a 0, es decir, $i = 0$ o $j = 0$, se igualaron a 0 para evitar que el color negro predominante en la imagen afecte a los descriptores obtenidos. Finalmente se calcularon las propiedades de textura tal como se describen en la Sección 2.1.1.

De la misma manera que en el proceso donde se utilizó GLCM, con LBP se transformó la imagen original a una imagen en escala de grises. Luego a partir de esta imagen se obtuvieron los patrones locales binarios, por último se generó el histograma de frecuencia de los bits correspondientes a la lesión de piel utilizando la mascara binaria. Inicialmente, los parámetros utilizados para obtener los patrones locales binarios fueron el método uniforme con radio de círculo equivalente a 3 y 12 puntos simétricos circulantes. Más adelante se presentarán otras pruebas con distintos valores para estos parámetros. El procedimiento se describe en la Fig. 25b.

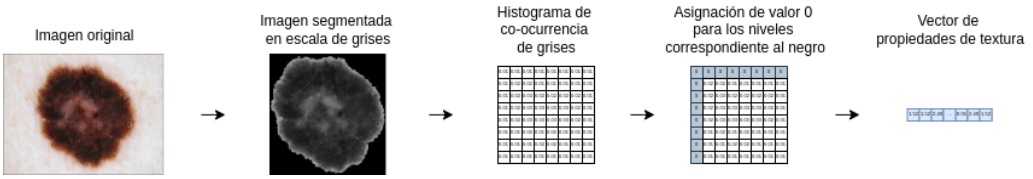
Para utilizar el descriptor de HOG se segmentó la lesión de piel a partir de su máscara binaria. A continuación se realizó un ajuste del tamaño de la imagen de 1024×1024 para luego obtener los descriptores de HOG aplanando sus valores obteniendo un único vector de dimensión 32768. Con el objetivo de generar los descriptores de HOG se utilizaron 9 orientaciones, sin normalización de los píxeles para reducir los efectos de la iluminación, con celdas de 8×8 píxeles y bloques de celdas de tamaño 3×3, y una normalización de los histogramas utilizando $L2\text{-}Hys$. La Fig. 25c ilustra el proceso mencionado.

El proceso para generar los Vectores de Fisher y entrenar un clasificador

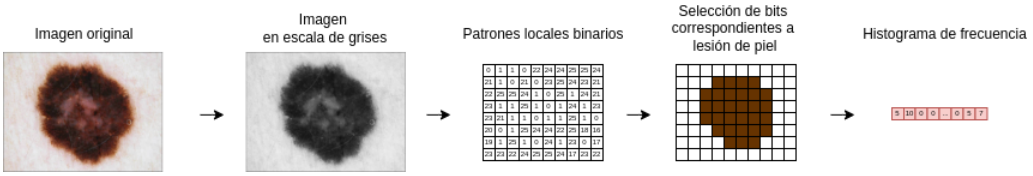
SVM a partir de los descriptores de cada imagen se ilustra en la Fig. 25d. En esta figura, se ejemplifica el proceso a partir de los descriptores GLCM, el cual se replica para los otros descriptores, LBP y HOG. En primer lugar, se llevaron a cabo transformaciones sobre los descriptores, que incluyen la normalización y la reducción de dimensionalidad. Los descriptores fueron escalados a un rango entre 0 y 1. Posteriormente, para los casos de GLCM y HOG, se aplicó PCA para reducir la cantidad de variables, seleccionando 32 componentes principales en cada caso. A continuación se generaron los modelos de mezcla gaussianas donde se trabajó con 16 componentes. Luego, con las componentes principales del descriptor en cuestión y el modelo de mezcla gaussianas se calcularon los Vectores de Fisher siguiendo el Algoritmo 1. Finalmente, se entrenó un clasificador SVM a partir de los Vectores de Fisher. En este clasificador se utilizaron como parámetros del mismo $C = 1$, $gamma = 0{,}1$ y la función de base radial como *kernel*. Cabe destacar que tanto el cálculo de los Vectores de Fisher como el entrenamiento del clasificador SVM se realizaron de manera independiente para cada tipo de descriptor por separado.

3.4.2 Experimento 1: conjunto de datos originales

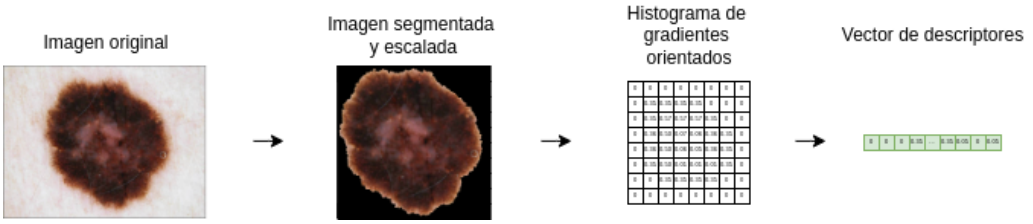
Para cada uno de los descriptores, GLCM, HOG y LBP, se realizó el proceso descrito en la sección anterior usando el conjunto de datos de entrenamiento indicado en la Fig. 22. Las métricas obtenidas en el conjunto de prueba de la competencia se presentan en el Cuadro 2, en las mismas se utilizó un umbral de 0.5 aplicado sobre la salida del SVM para determinar la clase. En estos resultados se observa la presencia de sobreajuste (en inglés *overfitting*). Este problema ocurre cuando un modelo de *machine learning* ajusta sus parámetros a los datos de entrenamiento y no es posible generalizar para hacer inferencias en otros conjuntos de datos, como ser el conjunto de prueba en este caso. La presencia del sobreajuste se entiende a partir de las métricas obtenidas, principalmente donde el *Accuracy* es igual a 0.802 para los modelos que utilizaron los descriptores de LBP y HOG. Esta clasificación se debe a que el modelo memorizó que la mayoría de los casos en el conjunto de entrenamiento pertenecen a la clase lesión benigna y de esta manera al hacer la inferencia en un nuevo conjunto de datos siempre asigna esta clase a todos los elementos de este conjunto. En nuestro caso el 80,2 % de las imágenes corresponden a la clase lesión benigna por este motivo el *Accuracy* obtenido es el mencionado anteriormente. En el caso de las métricas *Precision* y *Recall*, debido a que la cantidad de verdaderos positivos es 0, los resultados obtenidos en estas métricas también es 0. Como consecuencia que tanto el *Precision* y el *Recall* son 0, *F1 Score* también toma este valor.



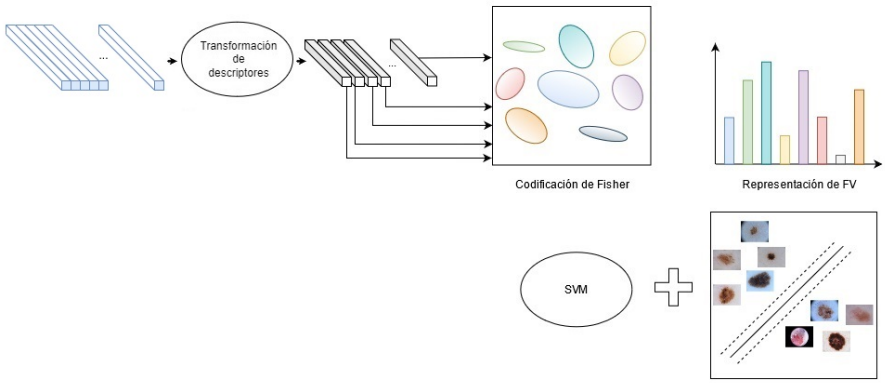
(a) Preparación de datos para obtener descriptores con GLCM.



(b) Preparación de datos para obtener descriptores con LBP.



(c) Preparación de datos para obtener descriptores con HOG.



(d) Codificación de Vectores de Fisher y clasificación a partir de descriptores GLCM.

Figura 25: Esquema propuesto para trabajar con descriptores y Vectores de Fisher. Proceso de construcción del descriptor para a) GLCM, b) LBP y c) HOG. (d) Codificación de los descriptores de GLCM en Vectores de Fisher, seguida de la clasificación mediante SVM.

Cuadro 2: Resultados por cada método de descriptor con los datos originales utilizando un umbral sobre la salida del SVM = 0.5.

Descriptor	mAP	AUC	*Accuracy*	*F1 Score*	*Precision*	*Recall*
GLCM	**0.352**	**0.685**	0.799	0	0	0
LBP	0.329	0.660	**0.802**	0	0	0
HOG	0.316	0.641	**0.802**	0	0	0

3.4.3 Experimento 2: conjunto de datos balanceado

Una de las estrategias para abordar el problema de un conjunto de datos desbalanceados es ajustar la cantidad de muestras de cada clase de manera tal que sean equivalentes. En este caso se realizó *undersampling*, es decir, se eliminaron aleatoriamente casos de la clase mayoritaria, las lesiones benignas, para que ambas clases estén representadas con la misma cantidad de muestras. Debido a que la clase minoritaria, las lesiones malignas, tiene 173 imágenes se utilizó esta cantidad para ambas clases. Una vez preparado el conjunto de datos, se realizó el mismo procedimiento que se mencionó en el Apartado 3.4.1 para entrenar un modelo con cada uno de los descriptores sobre este conjunto de datos. El Cuadro 3 muestra las métricas obtenidas para los modelos generados sobre el conjunto de prueba con un umbral sobre la salida del SVM = 0.5.

Cuadro 3: Resultados por cada método de descriptor con *undersampling* utilizando un umbral sobre la salida del SVM = 0.5.

Descriptor	mAP	AUC	*Accuracy*	*F1 Score*	*Precision*	*Recall*
GLCM	0.341	0.620	**0.746**	0.368	**0.363**	0.373
LBP	**0.402**	**0.688**	0.662	**0.457**	0.335	**0.720**
HOG	0.163	0.404	0.656	0.097	0.101	0.093

Si bien los valores para el *Accuracy* son menores a los modelos que se obtuvieron con el conjunto de datos original, en este caso son más precisos porque no solamente están clasificando correctamente ejemplos de una sola clase sino de ambas. En los experimentos realizados con el conjunto de datos original, los modelos siempre clasificaban las imágenes como lesiones benignas, en cambio, con el balanceo de datos que se realizó ahora los modelos también clasifican correctamente casos de lesiones malignas. Esto se puede observar por las otras métricas utilizadas. La métrica *Precision*, nos señala la tasa de acierto del modelo al momento de indicar que una imagen es una lesión maligna. Con los datos balanceados, la mejor tasa de acierto en las lesiones malignas se obtuvo con los descriptores GLCM, donde el 36,3 % de

las veces que el modelo predice la clase lesión de piel maligna, la predicción es correcta. Por su parte, el *Recall* nos señala la tasa de lesiones malignas detectadas, así por ejemplo, para el descriptor LBP, el modelo detecta el 72 % de las lesiones malignas. Comparando todos los modelos obtenidos con *undersampling*, estas tasas son mejores a las obtenidas con el conjunto de datos original. En cuanto al *F1 Score*, esta métrica representa el promedio armónico del *Precision* y el *Recall*, por lo que también muestra mejores valores para el conjunto de datos balanceado que para el conjunto de datos original. El AUC nos muestra la probabilidad que si tomamos aleatoriamente una imagen de lesión maligna, tenga una más alta probabilidad de predicción de ser maligna que una imagen aleatoria de lesión benigna. Mientras más alta sea el valor de esta métrica, el modelo le asignará mejores probabilidades de predicción a cada imagen. Para las pruebas realizadas, el modelo generado con LBP asigna de mejor manera estas probabilidades. Finalmente, el mAP nos indica el promedio de *Precision* para distintos umbrales de decisión. Esto quiere decir que si bien con el descriptor GLCM para un umbral sobre la salida del SVM = 0.5, la métrica *Precision* equivalente a 0.363 es mejor que la obtenida con los otros descriptores, promediando los valores de *Precision* para distintos umbrales de decisión el modelo obtenido con el descriptor LBP es el mejor de los tres con un mAP de 0.402. Si bien los resultados obtenidos con un balanceo de clases son mejores que sin hacer *undersampling*, no dejan de estar lejos de lo esperado por un modelo de *machine learning* que se pueda utilizar en una aplicación de la vida real. Por este motivo se decidió continuar el estudio de estos modelos modificando los parámetros para generar los descriptores de imágenes. Debido a que el descriptor LBP obtuvo mejores métricas comparando mAP, AUC y *F1 Score*, se decidió utilizar al mismo en pruebas con distintos parámetros. El trabajo realizado se menciona en el siguiente apartado.

3.4.4 Experimento 3: LBP

En estos experimentos se utilizó el descriptor LBP con distintos valores en sus parámetros sobre el conjunto de datos balanceados. Para la cantidad de puntos simétricos circulantes (p) se utilizaron los valores 12 y 24, mientras que los métodos utilizados fueron ror y $uniform$. El procedimiento con ror ajusta el patrón binario para que siempre comience desde la misma posición relativa, independientemente de la rotación. Esto permite que el descriptor sea invariante a rotaciones. En cuanto al método $uniform$, cuando hay 2 o menos transiciones en un patrón binario, este patrón se considera uniforme. En el descriptor LBP, los patrones uniformes se asignan a un valor único, mientras que los patrones no uniformes se agrupan en una sola categoría.

Esto reduce la dimensionalidad del descriptor y captura características de textura más significativas.

Por otra parte, al radio de círculo (n) se lo mantuvo constante en un valor equivalente a 3. El Cuadro 4 contiene los resultados alcanzados en el conjunto de prueba utilizando un umbral sobre la salida del SVM equivalente a 0.5.

Cuadro 4: Resultados para distintos modelos utilizando el descriptor LBP con *undersampling*.

Método	n	p	mAP	AUC	*Accuracy*	*F1 Score*	*Precision*	*Recall*
uniform	3	12	**0.402**	**0.688**	0.662	**0.457**	0.335	0.720
uniform	3	24	0.337	0.683	0.620	0.433	0.307	**0.733**
ror	3	12	0.320	0.626	**0.680**	0.421	0.328	0.586
ror	3	24	0.319	0.671	0.646	0.436	**0.693**	0.304

En los resultados es posible observar que cada uno de los modelos se destaca en al menos una de las métricas utilizadas.

El primero de los modelos, aquel con $p = 12$ y $método = uniform$, que se incluye en el análisis es el mismo que se analizó en el Apartado 3.4.3 donde se destacó sus resultados en mAP, AUC y *F1 Score*.

El segundo modelo, con $p = 24$ y $método = ror$, tiene el valor más alto en *Precision*, lo que implica que cuando este modelo predice la clase lesión de piel maligna, acierta en el 69,3 % de las veces.

El tercer modelo, con $p = 12$ y $método = ror$, es el que tiene el *Accuracy* mas alto, por lo cual acierta en la mayoría de sus predicciones. En este caso en el 68 % de las mismas.

Finalmente, el último modelo, con $p = 24$ y $método = uniform$, es el que tiene valor más alto en *Recall*, esto significa que de las imágenes de lesión de piel maligna, detecta como tal al 73,3 % de los casos. Teniendo en cuenta el contexto del problema que se quiere resolver, la tasa de falsos negativos debería ser la menor posible, dado que es mas peligroso clasificar a una persona que tiene una lesión de piel maligna como sana. Por este motivo, la métrica *Recall* sería la más acertada para tener en cuenta. De esta forma el modelo que tendría que considerarse en un contexto de aplicación en situaciones de la vida real sería este último. Si bien se lograron mejoras balanceado los datos de entrenamiento y modificando los parámetros para obtener los descriptores, los resultados obtenidos distan de lo esperado para un modelo de *machine learning* que pueda integrarse en una aplicación real. Por tal motivo, se continuó el estudio de esta problemática utilizando redes convolucionales y Vectores de Fisher generados a partir de descriptores obtenidos por una red convolucional. Estos desarrollos se presentan en los siguientes apartados.

3.5 Clasificación con ResNet-50

En los siguientes apartados, se abordará el entrenamiento de la red convolucional ResNet-50. En primer término, se explorará la división del conjunto de datos en entrenamiento y validación. Luego se describirán las distintas operaciones realizadas con las imágenes a fin de tener una mayor variedad de imágenes para entrenar el modelo. Por último, se describirá las distintas etapas del entrenamiento de la red convolucional ResNet-50 partiendo de un modelo previamente entrenado con otros datos.

3.5.1 Preparación de datos

Para entrenar la red ResNet-50, el conjunto de entrenamiento inicial fue dividido de manera aleatoria en una única partición fija, asignando el 80 % de las muestras para entrenamiento y el 20 % restante para validación. En ambas muestras se respetó la distribución de lesiones benignas y lesiones malignas. De esta manera se obtuvieron los conjuntos de datos detallados en la Fig. 26.

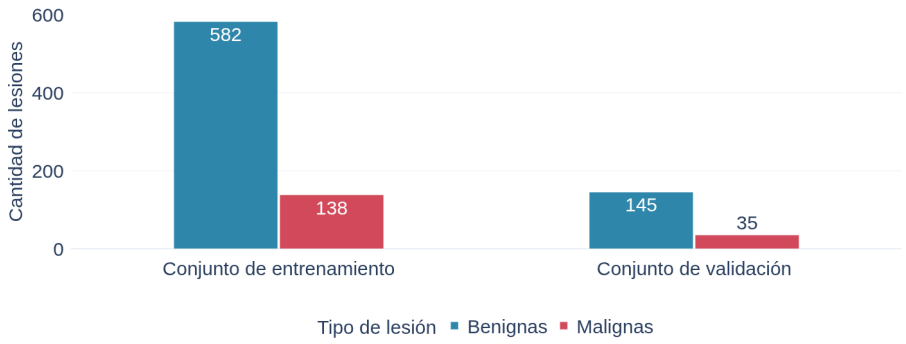
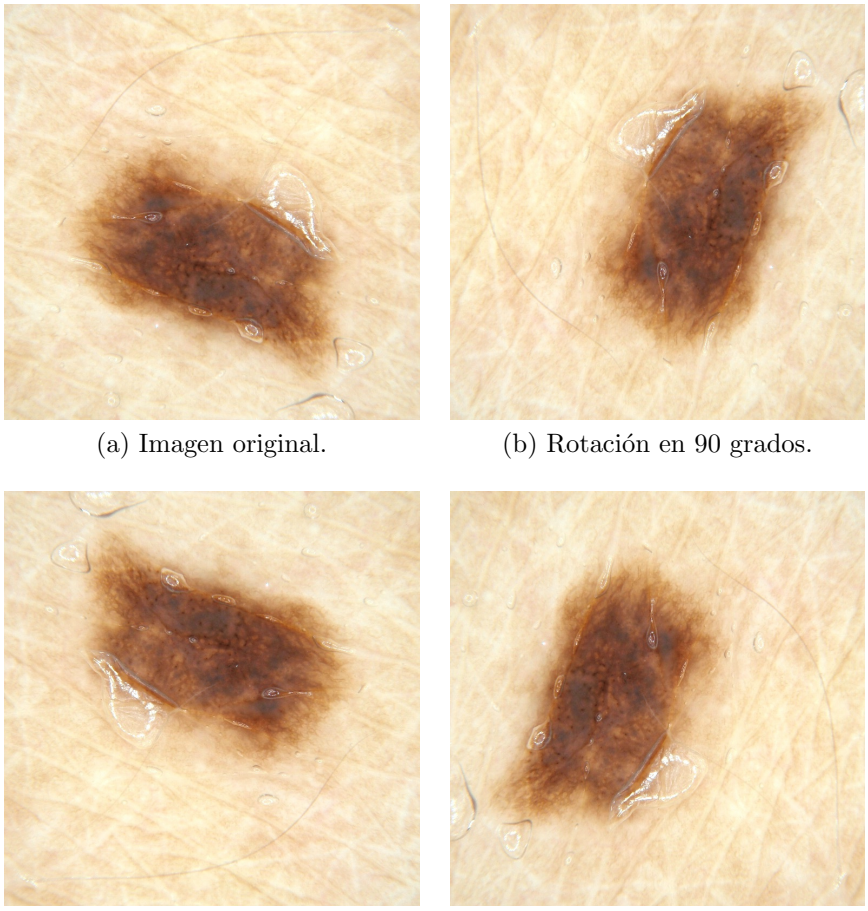


Figura 26: División del conjunto de datos en entrenamiento y validación.

Luego, para cada conjunto de datos se realizaron rotaciones en 90, 180 y 270 grados. Con el objetivo de tener la mayor variedad de imágenes para entrenar el modelo, se aplicaron operaciones de Aumento de datos (*Data Augmentation*) en las imágenes de entrenamiento al momento de entrenar el modelo. Las operaciones realizadas fueron:

- Redimensionamiento de las imágenes al tamaño de 224×224 píxeles. Debido a la amplia variedad de tamaños que presentan las imágenes del conjunto de datos se decidió utilizar el mismo tamaño con el que se entrenó originalmente la red ResNet-50 [34].



(a) Imagen original. (b) Rotación en 90 grados.

(c) Rotación en 180 grados. (d) Rotación en 270 grados.

Figura 27: Imagen original de una lesión de piel con rotaciones en 90, 180 y 270 grados.

- Zoom de valor aleatorio entre el 80 % y el 120 % de la imagen original.
- Reflejos aleatorios en los ejes x e y.
- Rotaciones aleatorias entre -40 y 40 grados.
- Traslaciones sobre los ejes x e y de la imagen.
- Escalado para convertir los valores de los píxeles del rango $0 - 255$ al rango $0 - 1$. El motivo de esta operación radica en que los valores entre 0 y 1 son los más adecuados para trabajar con redes convoluciones.
- Normalización y estandarización por imagen. La normalización escala los valores de los píxeles de una imagen restando a cada valor la media de la imagen, de esta manera su nueva media será igual a cero. Mientras que la estandarización consiste en dividir el valor de cada píxel por el desvío estándar de los píxeles de la imagen.

Para el conjunto de validación solo se aplicaron las operaciones de redimensionamiento, escalado, normalización y estandarización. La Fig. 28 especifica la distribución de los conjuntos de entrenamiento y validación luego de las operaciones mencionadas.

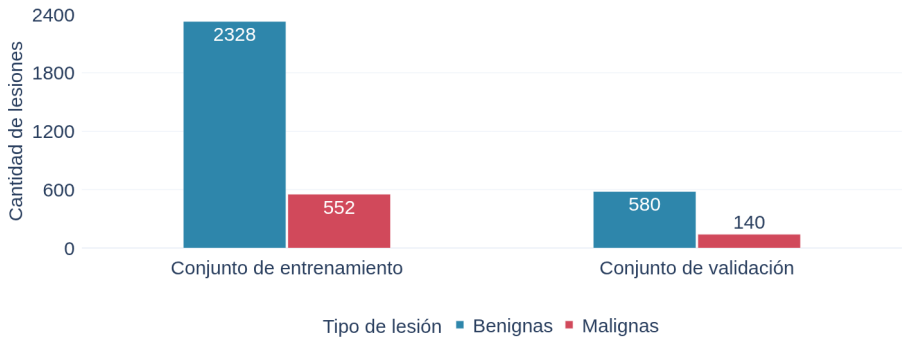


Figura 28: Conjunto de datos utilizado para el entrenamiento de la red ResNet-50 luego de aplicar operaciones de *Data Augmentation*.

3.5.2 Fine tuning y entrenamiento

A partir de la red entrenada con ImageNet, ver Cuadro 1, se realizó transferencia de aprendizaje de dicha red. Para adaptar este modelo a la aplicación actual fue necesario reemplazar la última capa totalmente conectada de 1000

salidas por una capa de una única neurona de salida con activación sigmoide como se muestra en la Fig. 29.

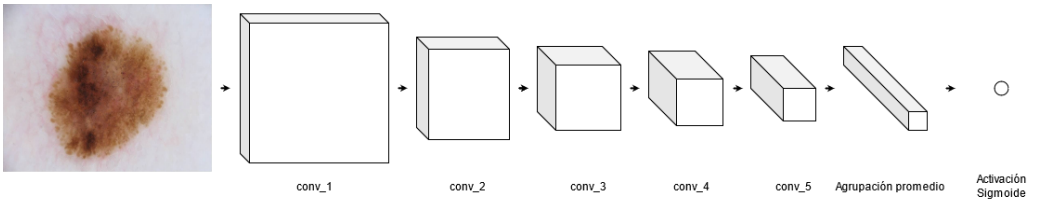


Figura 29: Estructura de la ResNet-50 con una neurona en la capa totalmente conectada utilizando una activación sigmoide.

Luego la red fue entrenada por etapas descongelando los pesos de las neuronas desde la capa superior hasta las capas menores. Así, en primer lugar, se realizaron distintos experimentos para entrenar la capa totalmente conectada de la red durante 100 épocas. En estos experimentos se probaron distintos valores del optimizador, la tasa de aprendizaje y el tamaño del lote de datos. La combinación de valores para cada experimento se detalla en el Apéndice A.1.2.

La elección de estos valores para los hiperparámetros de la red se detalla a continuación. En el caso de la tasa de aprendizaje, si su valor es demasiado grande la curva de aprendizaje mostrará oscilaciones violentas. Por el contrario, si la tasa de aprendizaje es demasiado baja, el aprendizaje avanza lentamente y puede quedarse estancado con un valor de costo alto. Estas situaciones se reflejan en el Apéndice A.1.1, donde al probar con un valor de *learning rate* = 0.01 la función de pérdida oscila bruscamente a lo largo de las distintas épocas, mientras que para un valor de *learning rate* = 0.0001 el aprendizaje avanza lentamente. Debido a estas pruebas realizadas se decidió profundizar el análisis con valores intermedios de la tasa de aprendizaje 0.001 y 0.0001.

En cuanto al *batch size*, algunos tipos de *hardware* logran un mejor tiempo de ejecución con tamaños específicos de lotes. Especialmente cuando se usan GPU (*Graphics Processing Unit*), como es el caso del presente trabajo, es común que la potencia de 2 tamaños de lote ofrezca un mejor tiempo de ejecución. Kandel et al. [51] demostraron que los lotes de datos pequeños obtienen mejores rendimientos con tasas de aprendizajes bajas. Como los valores de la tasa de aprendizaje son bajos, también se eligieron valores bajos de *batch size*. Estos valores son 16 y 32.

Finalmente, se eligió realizar los experimentos con dos de los optimizadores más utilizados en los problemas de aprendizaje profundo, ellos son el

optimizador de Adam y SGD.

Las métricas utilizadas para evaluar la performance de cada experimento realizado fueron el *Accuracy* y la entropía cruzada binaria. El mejor modelo generado, fue el realizado en el Experimento 4, es decir, con optimizador Adam, *tamaño de lote* = 32 y *tasa de aprendizaje* = 0.0001. La elección de esta prueba por sobre las demás se debe a que mientras las épocas avanzan el *Accuracy* aumenta lentamente y sin grandes oscilaciones llegando a un valor óptimo al completar las 100 épocas. En el caso del *Binary cross entropy*, su valor va disminuyendo a medida que avanzan las épocas también de manera lentamente. En ambos casos no se observa un sobreajuste a los datos de entrenamiento. La evolución del *Accuracy* y el *Binary cross entropy* para el Experimento 4, junto con los resultados de los demás experimentos, se detallan en el Apéndice A.1.2.

Posteriormente se descongelaron los pesos de las neuronas de los bloques convolucionales de manera progresiva en cuatro etapas. En la primera se descongelaron los pesos del quinto bloque convolucional, en la segunda los pesos del cuarto bloque convolucional, en la tercera los pesos del tercer bloque convolucional y finalmente, en la última etapa, se entreno la red completa. En cada etapa se realizó un entrenamiento durante 100 épocas con optimizador SGD, *batch size* = 32 y *learning rate* = 0.0001. La decisión de trabajar con SGD y un valor bajo de *learning rate* se debe a que el objetivo de esta fase del entrenamiento es modificar suavemente los pesos de la red convolucional sin destruir lo aprendido en el paso anterior.

En el Apéndice A.1.3 se presenta el *Accuracy* y *Binary cross entropy* para el entrenamiento de todas las épocas de la red convolucional. Analizando el *Accuracy*, se observa que esta métrica mejora progresivamente con el correr de las épocas sobre el conjunto de entrenamiento. Mientras que en el conjunto de validación, su valor mejora pero más lentamente hasta permanecer constante por encima de 0.8. En el caso del *Binary cross entropy*, la diferencia entre su valor para el conjunto de entrenamiento y el conjunto de validación es más notoria. En este caso, a partir de la época 200 el valor de la función de pérdida del conjunto de entrenamiento es menor que la del conjunto de validación. Este fenómeno es algo esperable en los modelos de aprendizaje automático debido a que el modelo va ajustando sus parámetros a partir de los datos de entrenamiento, lo importante aquí es que no exista una gran diferencia entre la métrica del conjunto de entrenamiento y la del conjunto de validación. Esto último ocurre en las últimas épocas del entrenamiento, por este motivo se decidió definir como punto de corte del entrenamiento a la época 300. En esta época se completa el entrenamiento luego de descongelar los pesos del cuarto bloque convolucional, a partir de ella el valor del *Accuracy* permanece constante para el conjunto de validación y por otro lado, el *Binary cross entropy* aumenta con más velocidad para el mismo conjunto de datos.

3.6 Clasificación con Vectores de Fisher

En el presente apartado se exploran alternativas de clasificación generando los Vectores de Fisher a partir de los descriptores de las red convolucional ResNet-50. Inicialmente, se realizaron experimentos trabajando con muestras de imágenes que luego se procesaron con la red convolucional ResNet-50 para después generar los Vectores de Fisher y entrenar un clasificador. Seguidamente, en un segundo grupo de experimentos, se realizaron pruebas procesando la imagen de manera completa por la red convolucional a fin de posteriormente generar los Vectores de Fisher. Por último se experimentó generar un ensamble de los dos modelos previamente mencionados.

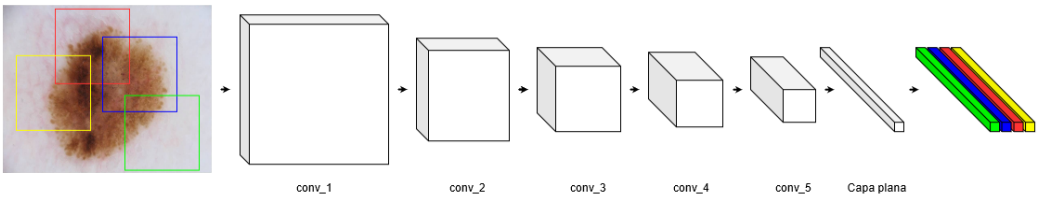
3.6.1 Fisher con muestras

En las siguientes páginas se explora una alternativa de modelo de clasificación a partir de Vectores de Fisher generados con descriptores de la red convolucional ResNet-50. En primer lugar, se describe el proceso de generación de descriptores a partir de muestras de una imagen utilizando la red convolucional mencionada, así como la posterior codificación de estos descriptores en Vectores de Fisher. Luego se establecen los valores a utilizar para los distintos parámetros en los experimentos. Finalmente, se presentan los resultados de los experimentos utilizando el método de validación cruzada.

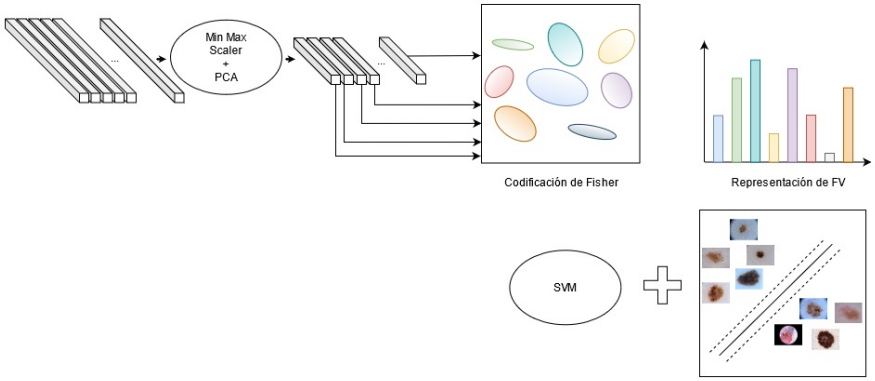
3.6.1.1 Procedimiento

Para obtener las representaciones de las imágenes a través de una red CNN y procesar las mismas con FV se siguió una metodología similar a la planteada por Yu et al. [5]. En nuestro esquema de representación, cada imagen de lesión cutánea se divide en múltiples subimágenes mediante muestreo aleatorio. Posteriormente, se extraen características profundas de cada una de estas subimágenes. Específicamente, a partir de la red previamente entrenada en la Sección 3.5 las capas de agrupación promedio y activación sigmoide fueron reemplazadas por una capa plana como se muestra en la Fig. 30a. Esta red toma imágenes de tamaño 224×224 como entrada. Cada una de las subimágenes es procesada por la red y se obtienen las representaciones generadas por la capa plana. Por lo tanto, se obtiene un conjunto de N vectores de características de dimensión 2048 para cada imagen, siendo N la cantidad de subimágenes procesadas.

Para un cálculo más eficiente, los vectores de características son escalados y su dimensionalidad es reducida mediante PCA. Luego se entrena un modelo de mixtura de gaussianas (GMM) con K componentes. Seguidamente a partir de los descriptores de una imagen y las componentes de GMM se obtienen



(a) Descriptores de cada muestra.



(b) Codificación de Vectores de Fisher y clasificación.

Figura 30: Esquema propuesto para trabajar con Vectores de Fisher. (a) Proceso para obtener los descriptores de cada muestra con una capa Flatten al final de la red (b) Codificación de los descriptores obtenidos mediante Vectores de Fisher.

los FV aplicando el Algoritmo 1. Finalmente, se entrena un clasificador de máquinas de vectores de soporte (SVM). Tanto la cantidad de componentes principales como la cantidad de componentes gaussianas son hiperparámetros del modelo. Para entrenar los modelos de PCA y GMM se trabajó con $N = 32$ muestras a partir de cada imagen, mientras que para generar los FV se realizaron $N = 64$ muestras a partir de cada imagen. La diferencia en la cantidad de muestras utilizadas para cada etapa del proceso radica en el mayor costo computacional del entrenamiento de PCA y GMM. A fin de determinar los mejores parámetros del clasificador SVM se realizó un entrenamiento de validación cruzada con 5 iteraciones. La imagen 30b resume el proceso de codificación de FV y el entrenamiento del clasificador.

3.6.1.2 Elección de valores de los hiperparámetros

Con el objetivo de definir la cantidad de componentes principales a trabajar en los experimentos del modelo, se analizó la varianza explicada a partir de la

cantidad de componentes principales. En la Fig. 31 se presenta esta información donde es posible apreciar que con 244 componentes es posible explicar el 80 % de la varianza total. Por tal motivo se decidió realizar experimentos con valores cercanos a 244. Además se eligió utilizar valores equivalentes a potencias de 2 y valores intermedios entre los mismos. De esta manera se definió trabajar con $32, 64, 128, 256, 384$ y 512 componentes principales.

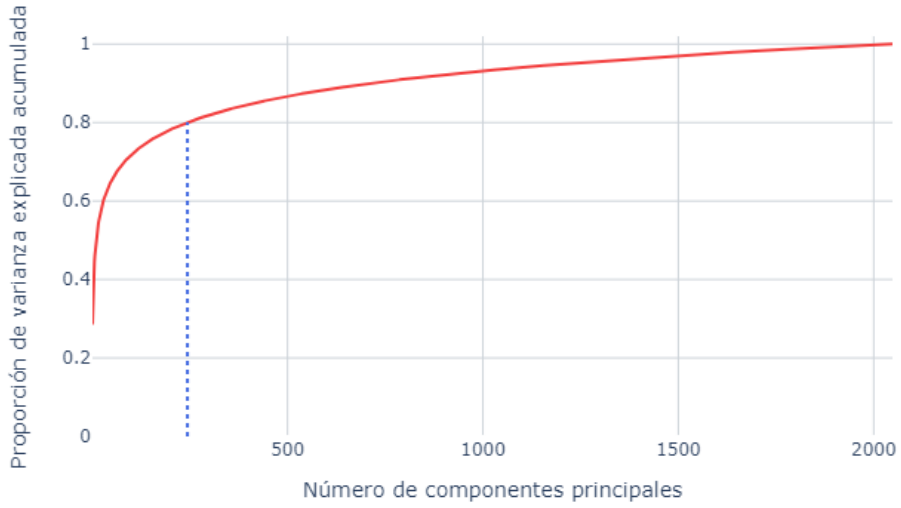


Figura 31: Varianza explicada acumulada de los vectores de características de las imágenes según el número de componentes principales que se retengan. Con 244 componentes se explica el 80 % de la varianza total.

En el caso de las componentes gaussianas se optó por emplear los mismos valores utilizados en aplicaciones previas [7, 6], es decir trabajar con $16, 32, 64$ y 100 componentes gaussianas. De esta manera se realizaron 24 experimentos combinando los distintos valores de componentes principales y componentes gaussianas. Estos experimentos se detallan en el Apéndice A.2.

En cada experimento se probaron distintos valores de algunos de los parámetros del clasificador SVM mediante el método de validación cruzada. Así para el clasificador SVM se utilizaron dos funciones *kernel*, la función polinómica de grado 3 y RBF (*Radial Basis Function*). En todos los casos se estudiaron los parámetros C y *gamma*. El parámetro de regularización C, compensa la clasificación errónea de los ejemplos de entrenamiento frente

a la simplicidad de la superficie de decisión. Un valor de C bajo suaviza la superficie de decisión, mientras que una C alta apunta a clasificar correctamente todos los ejemplos de entrenamiento. Por su parte, *gamma* define cuánta influencia tiene un solo ejemplo de entrenamiento. Cuanto mayor sea *gamma*, más cerca deben estar otros ejemplos para verse afectados [52]. Los valores utilizados para el parámetro C son $[1, 10, 100, 1000]$ mientras que para el parámetro *gamma* se usaron los valores [0.1, 0.001, 0.0001, 0.00001].

Debido a que existe un desbalance entre las clases, ya que la cantidad de imágenes de lesiones benignas son mayores a la cantidad de melanomas, se trabajó ponderando los ejemplos de cada clase con el parámetro *class weight* del clasificador SVM. El parámetro *class weight* se compone de p elementos CW_i donde p representa la cantidad de clases a clasificar y CW_i la penalización de un error en la clasificación de la clase i. De esta manera, un valor de CW_i alto significa que se quiere poner más énfasis en la clase i. El conjunto de datos utilizado tiene 900 imágenes, con 727 lesiones benignas y 173 melanomas, es decir, hay $723/173 \approx 4{,}2$ imágenes de lesiones benignas por cada imagen de melanoma. Por este motivo las siguientes ponderaciones para cada clase fueron utilizadas:

- $CW_{Lesiones\ de\ piel\ benignas} = 1$
- $CW_{Melanomas} = 4{,}2$

3.6.1.3 Resultado de valores de los hiperparámetros

En el Apéndice A.2 se detalla las distintas combinaciones de los parámetros cantidad de componentes principales y cantidad de componentes gaussianas. Para cada combinación de parámetros se realizó un entrenamiento con validación cruzada de 5 iteraciones. Es decir, para cada caso el conjunto de entrenamiento se divide en 5 grupos iguales respetando la distribución de las imágenes de lesiones benignas y melanomas. Luego se entrena un modelo utilizando 4 grupos y dejando un restante para obtener las métricas de evaluación del modelo resultante. Esta etapa se realiza 5 veces utilizando todos los grupos como conjunto de prueba. Finalmente se obtiene un promedio de cada métrica de evaluación el cual se utiliza para analizar los resultados de los parámetros utilizados.

En este apartado se analizan las métricas AUC y *mean Average Precision*, para más información sobre los resultados de los modelos consultar el Apéndice A.3.

En primer lugar se analiza el rendimiento del modelo a partir de la cantidad de componentes principales y la cantidad de componentes gaussianas,

estos resultados se presentan en el Apéndice A.3.1. Comparando la distribución del AUC, se observa que con 32 componentes principales se obtienen valores más bajos tanto en AUC como en *mean Average Precision*, a partir de 64 componentes no se observan diferencias significativas si se agregan más componentes al modelo. En relación a la cantidad de componentes gaussianas se observa mayor dispersión en los resultados a mayor cantidad de componentes gaussianas, especialmente con 100 componentes gaussianas. Por estos motivos se decidió investigar a los modelos generados con 64 componentes principales y 16 componentes gaussianas.

Tomando los modelos generados con 64 componentes principales y 16 componentes gaussianas, el siguiente parámetro a analizar es el parámetro C del clasificador SVM (ver Apéndice A.3.2). En este caso, al analizar la distribución del AUC y el *mean Average Precision* se observa claramente que mientras más pequeño es el valor de C mejor es la performance del modelo. Debido a esto se optó por trabajar con un valor de C equivalente a 1.

Por último, analizando el rendimiento de los modelos a partir de los parámetros *kernel* y *gamma* del clasificador SMV, se observa que el *kernel* rbf tiene mejores resultados para los distintos valores de *gamma* tanto en el AUC y en el *mean Average Precision*. Estos resultados se presentan en el Apéndice A.3.3. En cuanto al valor de *gamma*, aunque no se observaron diferencias significativas, se seleccionó 0.1 por lograr las mejores métricas en los experimentos.

En conclusión, los valores elegidos para el método de Vectores de Fisher con muestras son los siguientes:

- Número de componentes principales = 64
- Número de componentes gaussianas = 16
- $C = 1$
- $kernel$ = rbf
- $gamma = 0.1$

Una vez encontrados los mejores parámetros del modelo, el paso siguiente fue entrenar el modelo utilizando la totalidad de los datos de entrenamiento con los mejores valores de cada hiperparámetro. Los resultados del modelo generado en el conjunto de prueba se analizarán en la Sección 4.

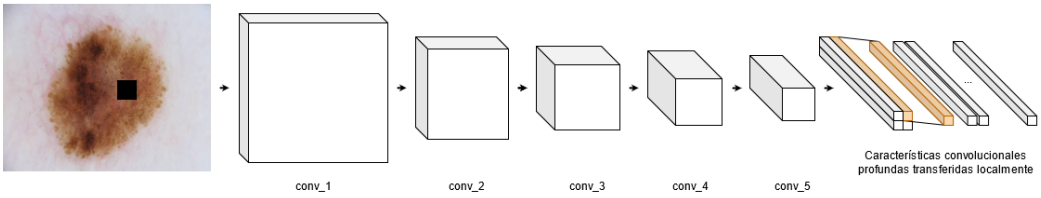


Figura 32: Proceso para obtener los descriptores a partir del quinto bloque convolucional.

3.6.2 Fisher con descriptores

Este apartado se centra en los experimentos realizados a partir de Vectores de Fisher generados con descriptores de la red convolucional ResNet-50. Como punto de partida se expondrá el proceso para generar los Vectores de Fisher, a continuación se determinaran los distintos valores de los parámetros en los experimentos y por ultimo, se presentarán los resultados de los experimentos siguiendo el método de validación cruzada.

3.6.2.1 Procedimiento

El proceso es similar al planteado por Yu et al. [7]. La implementación partió de la red entrenada que fue descrita en la Sección 3.5. Posteriormente, se eliminaron las capas de agrupación promedio y activación sigmoide, utilizando exclusivamente los descriptores generados por el quinto bloque convolucional como se muestra en la Fig. 32. Dada una imagen X_i se realizan rotaciones en 90°, 180° y 360° obteniendo cuatro muestras $\mathbb{X}_i = \{X_{i1}, X_{i2}, X_{i3}, X_{i4}\}$ de igual tamaño, en este caso 224×224. Cada una de estas muestras es procesada por la red obteniendo como salida del quinto bloque convolucional un vector de dimensión $7 \times 7 \times 2048$ para cada muestra. Para continuar con el proceso se decidió trabajar con vectores de dos dimensiones 49×2048, de esta manera, a partir de la imagen original se obtienen cuatro vectores $\mathbb{F}_i = \{F_{i1}, F_{i2}, F_{i3}, F_{i4}\}$ con $F_{ij} \in \mathbb{R}^{49 \times 2048}$ para $j = 1, ..., 4$. Con los vectores de todas las imágenes se entrena un modelo de mixtura de gaussianas, previo escalamiento y reducción de la dimensionalidad con PCA según lo planteado en la Fig. 30b. Una vez entrenado un GMM, a partir del mismo se obtienen los Vectores de Fisher de cada imagen con los descriptores siguiendo el Algoritmo 1. Finalmente los parámetros del clasificador SVM fueron elegidos utilizando validación cruzada con 5 iteraciones.

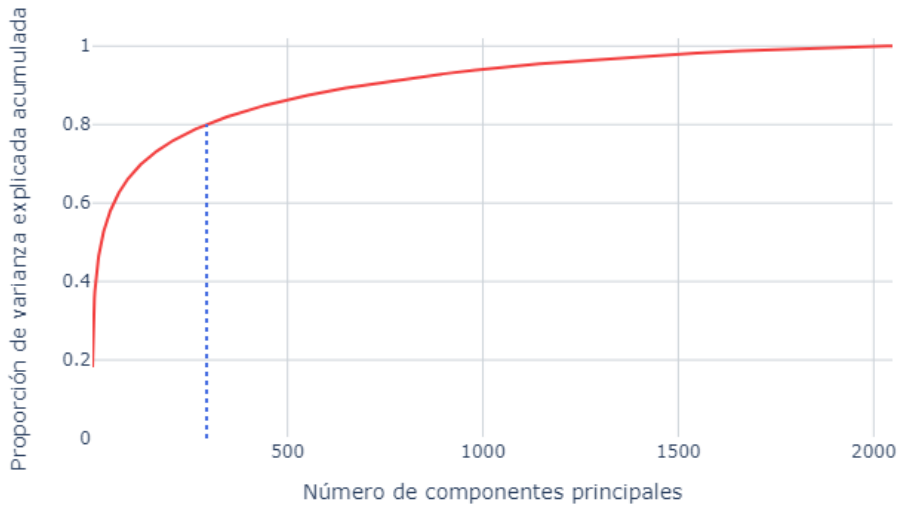


Figura 33: Varianza explicada acumulada de los descriptores de las imágenes según el número de componentes principales que se retengan. Con 293 componentes se explica el 80 % de la varianza total.

3.6.2.2 Elección de valores de los hiperparámetros

Los hiperparámetros elegidos para este modelo son los mismos que se eligieron en el método de Vectores de Fisher con muestras (Apartado 3.6.1.2). En el Apéndice A.2 se detallan los valores de número de componentes principales y número de componentes gaussianas.

La Fig. 33 muestra la varianza explicada por cantidad de componentes principales. En este caso con 293 componentes se logra explicar el 80 % de la varianza total. Si bien aquí es necesaria una mayor cantidad de componentes para explicar el 80 % de la varianza que en el método desarrollado en el Apartado 3.6.1, se decidió utilizar los mismos valores de cantidad de componentes en estos experimentos ya que la curva de varianza explicada es similar en ambos casos.

Al momento de entrenar el clasificador SVM para cada experimento detallado en el Apéndice A.2 se realizó *cross validation* con 5 *folds*. Los parámetros que se investigaron en este procedimiento son los siguientes:

- *kernel*: polinómico y radial

- *gamma*: 0.1, 0.01, 0.001, 0.0001
- C: 1, 10, 100, 1000

De igual forma que en el método de Vectores de Fisher con muestras, se trabajó ponderando los ejemplos de cada clase para afrontar el problema de desbalanceo de clases. Las ponderaciones por cada imagen según su clase son:

- Lesiones de piel benignas: 1
- Melanomas: 4,20

3.6.2.3 Resultado de valores de los hiperparámetros

El análisis inicial evaluó el comportamiento de los resultados según el número de componentes principales y gaussianas. En el Apéndice A.4.1 se presentan la distribución del AUC, *mean Average Precision* y *Accuracy* en función de los parámetros mencionados anteriormente. Analizando las primeras dos métricas mencionadas, es posible observar que los mejores resultados fueron alcanzados con 64 componentes principales ya que con este valor se alcanzan las medianas más altas para ambas métricas. También puede mencionarse que con 16 y 32 componentes principales la dispersión de los resultados es menor. En el caso de la cantidad de componentes gaussianas no se aprecia un patrón general, ya que la variación se ve muy afectada por la cantidad de componentes principales que se este considerando. Para continuar con el estudio se eligió la alternativa de 64 componentes principales y 32 componentes gaussianas ya que es la que tiene las medianas más altas en ambas métricas.

Partiendo del experimento con 64 componentes principales y 32 componentes gaussianas, se analizó el comportamiento de los parámetros del clasificador SVM. En el Apéndice A.4.2 se muestra la distribución de las métricas AUC, *mean Average Precision* y *Accuracy* en función del valor de *gamma*. Para los distintos valores de *gamma*, el primer cuartil presenta valores similares, mientras que la mediana y el tercer cuartil varían entre grupos, especialmente para $gamma = 0.1$. Este último grupo alcanza el mejor rendimiento en ambas métricas.

En el Apéndice A.4.3 se comparó el rendimiento de los modelos a partir de los parámetros *kernel* y C del clasificador SVM. Allí no se observan diferencias significativas en las métricas planteadas. En este caso se escogió tomar la mejor combinación de parámetros, es decir, $kernel = \text{rbf}$ y $C = 10$ ya que esta combinación alcanzó el mejor rendimiento en ambas métricas.

Resumiendo, se eligieron los siguientes parámetros:

- Número de componentes principales = 64
- Número de componentes gaussianas = 32
- $C = 10$
- $kernel =$ rbf
- $gamma = 0.1$

De igual forma que en el método de Vectores de Fisher con muestras, luego de determinar los valores óptimos de cada hiperparámetro, se realizó el entrenamiento del modelo utilizando todos los datos del conjunto de entrenamiento con los mejores valores de cada hiperparámetro. Más adelante, en la Sección 4, se presentarán las métricas de evaluación en el conjunto de prueba.

3.6.3 Ensamble de modelos de Vectores de Fisher

Los métodos de ensamble son técnicas de aprendizaje automático que construyen un conjunto de modelos predictivos y fusionan sus resultados en una única predicción. El objetivo de combinar múltiples modelos es lograr un mejor rendimiento predictivo, y se ha demostrado en numerosos casos que los ensambles pueden ser más precisos que los modelos individuales [53]. Por este motivo, a partir de los modelos entrenados con Vectores de Fisher se creó un cuarto modelo que es el ensamble de los mismos. La probabilidad que este ensamble asigna a una lesión de piel corresponde al promedio de las probabilidades generadas por los métodos de Vectores de Fisher. Las métricas de este ensamble se expondrán en la Sección 4.

4 Resultados

A lo largo de esta sección, se presentan y analizan los resultados obtenidos en los experimentos con ResNet-50 y Vectores de Fisher, cuyos fundamentos metodológicos se detallaron en secciones anteriores. Primeramente, se analizan los resultados en el conjunto de datos de prueba de los modelos entrenados considerando las distintas métricas de evaluación. Luego se realiza una comparación con otras aplicaciones que utilizaron el mismo conjunto de datos.

4.1 Resultados en el conjunto de datos de prueba

Uno de los objetivos del presente trabajo es desarrollar modelos con capacidad de generalización, lo que implica mantener un rendimiento consistente cuando se aplica el modelo a conjuntos de imágenes diferentes a las utilizadas en la fase de desarrollo. Para validar este objetivo, en la presente sección se analiza el rendimiento de los modelos desarrollados sobre el conjunto de prueba de la competencia ISIC 2016.

En los cuadros y figuras siguientes, los modelos se identifican mediante abreviaturas para facilitar la presentación de resultados. Estas corresponden a: ResNet-50-FT (*Fine tuning* a la red convolucional ResNet-50), ResNet-50-FV-CM (Vectores de Fisher con muestras) y ResNet-50-FV (Vectores de Fisher con descriptores) y FV-Ensamble (Ensamble de modelos de Vectores de Fisher). En el Cuadro 5 se presentan los resultados de los métodos propuestos en el conjunto de prueba. Para comparar los modelos se utilizaron las métricas: *mean Average Precision* (mAP), área bajo la curva ROC (AUC), *Accuracy* (Acc), *F1 Score*, *Precision* y *Recall*. En el caso de las últimas cuatro, se utilizó un umbral sobre la salida del SVM = 0.5 para el cálculo de las mismas. Esto quiere decir que a partir de la probabilidad asignada por un modelo a una imagen de lesión de piel, se clasifica a la misma como melanoma si su probabilidad es mayor a 0.5, y en caso contrario se considera a la lesión de piel como benigna.

Cuadro 5: Métodos propuestos con un umbral sobre la salida del SVM = 0.5.

Método	mAP	AUC	*Accuracy*	*F1 Score*	*Precision*	*Recall*
ResNet-50-FT	0.5852	0.7962	0.8258	**0.5217**	0.5714	**0.4800**
ResNet-50-FV-CM	0.5462	0.7558	0.8416	0.4000	0.8000	0.2666
ResNet-50-FV	0.6134	0.8093	**0.8469**	0.4423	0.7931	0.3066
FV-Ensamble	**0.6446**	**0.8295**	0.8390	0.3296	**0.9375**	0.2000

De las métricas se puede observar que el modelo ResNet-50-FT es el que alcanzó mejores resultados en las métricas *Recall* y *F1 Score*, en contraste, con un valor bajo en *Precision*. Esto se debe a que, por un lado, clasifica erróneamente como lesión benigna a menos cantidad de casos que los otros modelos utilizando un umbral sobre la salida del SVM = 0.5. Por otro lado, clasifica erróneamente como melanomas a más lesiones que los otros modelos.

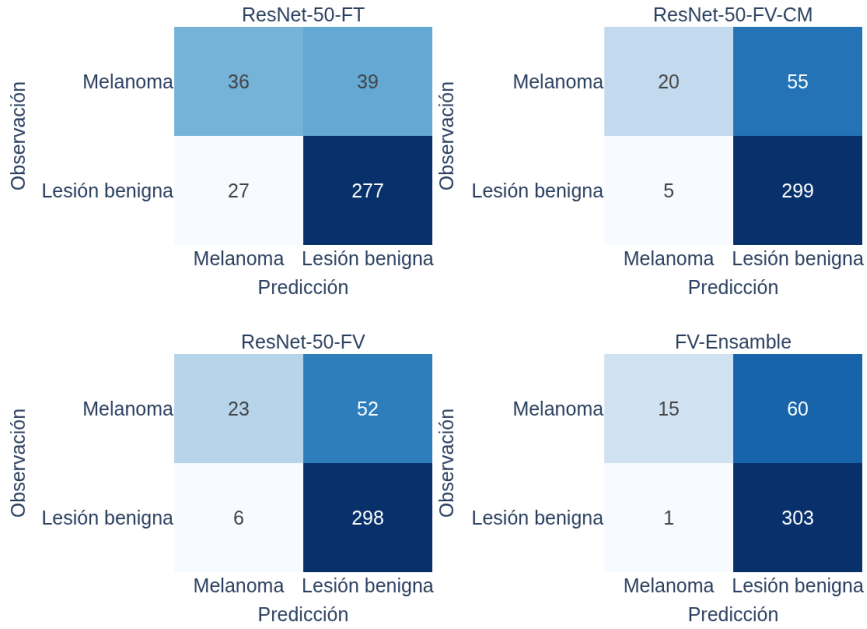


Figura 34: Matrices de confusión de los modelos ResNet-50-FT, ResNet-50-FV-CM, ResNet-50-FV y FV-Ensamble.

Entre los modelos generados con Vectores de Fisher, el que utiliza los descriptores (ResNet-50-FV) es el que mejores resultados logró en la mayoría de las métricas. La única excepción ocurre con la métrica *Precision* donde el modelo generado a partir de muestras de una imagen tiene levemente un mejor rendimiento.

Finalmente se puede mencionar que el Ensamble propuesto tiene los mejores resultados en mAP, AUC y *Precision* pero su rendimiento es el peor al considerar el *F1 Score* y el *Recall*. La Fig. 34 muestra las matrices de confusión de los modelos mencionados, evaluados sobre el conjunto de prueba.

En la Fig. 35 se presentan los gráficos de las curvas ROC de cada modelo. El Área bajo la curva ROC (AUC) se puede interpretar como la probabilidad de que un clasificador ordenará o puntuará un Melanoma elegido aleatoria-

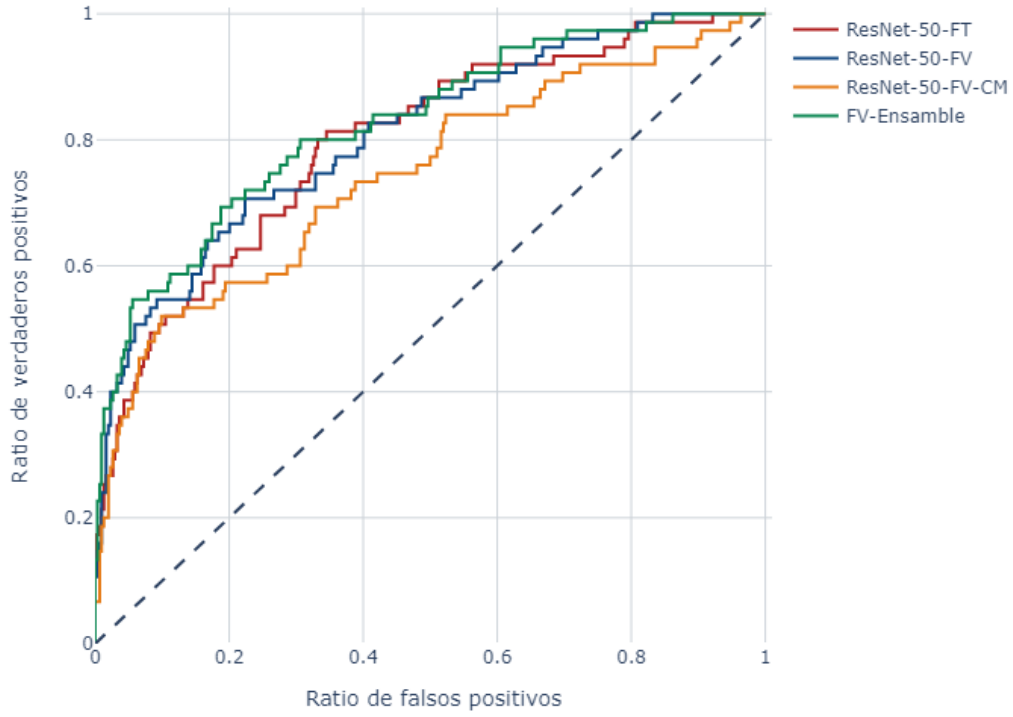


Figura 35: Curvas ROC de los distintos modelos.

mente más alto que a una lesión de piel benigna. En el caso de los modelos analizados quien puntúa mejor a las lesiones de piel es el FV-Ensamble.

Cuando se trata de problemas de clasificación con clases desbalanceadas, como en nuestro caso, es importante observar la Curva *Precision-Recall* para analizar cómo se comportan las métricas *Precision* y *Recall* con diferentes valores del umbral sobre la salida del SVM. En la Fig. 36 se presentan las curvas *Precision-Recall* para cada modelo, donde el FV-Ensamble muestra la mejor combinación de estas métricas para distintos umbrales. En contraposición el modelo que peor curva presenta es el ResNet-50-FV-CM.

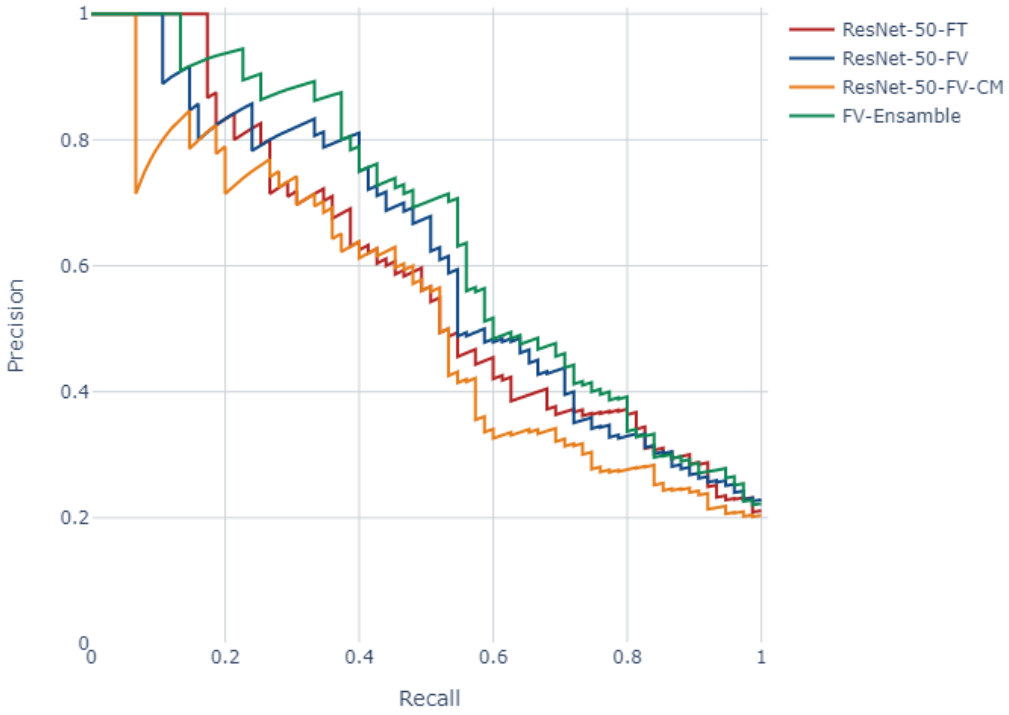


Figura 36: Curvas *Precision-Recall*.

4.2 Comparación con otras aplicaciones

El Cuadro 6 compara el rendimiento del ensamble con otras aplicaciones realizadas utilizando el mismo conjunto de datos de prueba de ISIC 2016 [49]. Yu et al. [6, 7] desarrollaron una arquitectura similar a la propuesta en la Sección 3.6.2. Por otro lado, en la Sección 3.6.1 se replicó el esquema planteado por Yu et al. [5]. La principal diferencia entre estas aplicaciones y el presente trabajo radica en la red convolucional utilizada para extraer los descriptores profundos locales. Mientras que en los trabajos mencionados se emplea una red entrenada con ImageNet, en este desarrollo se utilizó una red preentrenada específicamente con un conjunto de datos de lesiones de piel.

Continuando con otras aplicaciones, Yu et al. [4] presentó un esquema de redes convolucionales residuales para realizar la segmentación de las lesiones de piel en la imagen, luego integran este modelo con una red convolucional para resolver el problema de clasificación de imágenes. Zang et al. [8] proponen un esquema de modelo de Aprendizaje Profundo Sinérgico utilizando de manera simultanea múltiples redes convolucionales donde además permiten el aprendizaje entre ellas. A su vez, en Yu et al. [10] extraen los descriptores locales de múltiples redes convolucionales, luego seleccionan las características profundas de la región mas relevante de los mapas de activación para finalmente codificarlos con Vectores de Fisher. Por último, Liberman et al. [9] plantean un Ensamble de tres modelos de clasificación, una red convolucional VGG-16, y dos codificaciones de Vectores de Fisher, una con segmentación y la restante sin segmentación.

En el Cuadro 6 para el cálculo del *Accuracy*, al igual que en el Apartado 4.1, se utilizó un umbral sobre la salida del SVM = 0.5. Como se puede observar, el ensamble propuesto es superior al mejor clasificador de la competencia ISBI 2016 [4] en las métricas mAP y AUC. Sin embargo existen otras aplicaciones desarrolladas que lograron mejores rendimientos a ambos trabajos [7, 10].

Cuadro 6: Comparación de las aplicaciones con otros métodos.

Método	Red	mAP	AUC	Acc
FV-Ensamble	ResNet-50	0.6446	0.8295	0.8390
DCNN-FV [7]	ResNet-50	0.6849	0.8520	**0.8681**
DCNN-FV [5]	AlexNet-FC6	0.5350	0.7957	0.8309
CUMED [4]		0.6370	0.8040	0.8550
SLD [8]		0.6810	0.8220	0.8630
M-CNN-FV [10]		**0.6873**	**0.8582**	**0.8681**
VGG-16-FV [9]	Ensamble	0.6813	0.8211	0.8029

5 Discusión

A lo largo del trabajo, surgieron dos dificultades principales relacionadas con las imágenes disponibles para el desarrollo del modelo. En primer lugar, el número total de imágenes de lesiones cutáneas era reducido (900 casos), lo que en problemas de clasificación de imágenes puede conducir a sobreajuste, donde los modelos memorizan características específicas del conjunto de entrenamiento en lugar de aprender patrones relevantes. Para contrarrestar este efecto, se implementaron técnicas de *data augmentation* que incluían rotaciones y traslaciones de las imágenes. Si bien estas estrategias permitieron alcanzar resultados del orden de magnitud de trabajos anteriores, persisten limitaciones inherentes al tamaño muestral original. Aún cuando existían otras fuentes de información que podrían haber ampliado la muestra, se optó por trabajar exclusivamente con el repositorio ISIC 2016 para garantizar la comparabilidad del trabajo con estudios previos en el área. De haberse optado por ampliar el conjunto de datos, habría sido necesario abordar desafíos computacionales adicionales, ya que el procesamiento de un volumen mayor de imágenes habría requerido mayor capacidad de almacenamiento, procesamiento y tiempos de entrenamiento extendidos.

Un segundo desafío fue la distribución desigual entre clases: aproximadamente el 80 % de las imágenes correspondían a lesiones benignas y solo el 20 % a melanomas. Este desbalance, puede afectar el rendimiento del algoritmo al generar un sesgo hacia la clase mayoritaria, por lo que fue necesario implementar estrategias específicas durante el entrenamiento. Una de las estrategias utilizadas fue realizar un *undersampling* de la clase predominante. Este enfoque se utilizó con los descriptores de imágenes (GLCM, LBP y HOG) donde no se obtuvieron los resultados esperados. Para el entrenamiento del clasificador SVM basado en Vectores de Fisher derivados de los descriptores de la ResNet-50, se implementó una estrategia de ponderación de clases, asignando un peso mayor a los ejemplos pertenecientes a la clase minoritaria, con el fin de compensar el desbalanceo inherente al conjunto de datos. Al observar los resultados alcanzados, especialmente en la métrica *Recall* donde ningún experimento logra superar el valor de 0.5, se puede concluir que esta estrategia no fue suficiente para resolver el problema del desbalanceo de clases.

6 Conclusiones

Este capítulo sintetiza las conclusiones fundamentales del trabajo, organizadas a partir de los logros alcanzados y las limitaciones persistentes, donde se evalúa el cumplimiento de los objetivos iniciales y se plantean los desafíos no resueltos. Luego se presentan líneas de trabajo futuro derivadas de las dificultades encontradas en el trabajo. Finalmente, se presenta el acceso al desarrollo técnico, incluyendo la referencia al repositorio público que contiene el código.

6.1 Logros y limitaciones

En el presente trabajo se desarrollaron modelos de clasificación de lesiones de piel que permiten distinguir las lesiones benignas de los melanomas. En primer lugar, se trabajaron modelos basados en clasificadores SVM, los cuales emplearon Vectores de Fisher generados a partir de descriptores de imágenes como GLCM, LBP y HOG. Seguidamente, se entrenó una red convolucional ResNet-50 para abordar el problema anteriormente mencionado. Luego, a partir de esta red, los descriptores profundos locales de cada imagen fueron generados para codificarlos en Vectores de Fisher. La codificación en Vectores de Fisher se implementó mediante dos estrategias. En la primera se obtuvieron los descriptores a partir de muestras de la imagen original, para luego codificarlos. En la segunda, se procesó la imagen completa con la red convolucional, generando los descriptores que posteriormente se codificaron. Para cada variante, se entrenó un clasificador SVM. Finalmente, se generó un modelo ensemble combinando ambos enfoques basados en Vectores de Fisher.

Al iniciar el trabajo se plantearon como objetivos desarrollar un modelo de detección de lesiones malignas con capacidad de generalización en un conjunto de imágenes no utilizadas para el desarrollo del modelo. Los resultados de este modelo deberían ser comparables o superadores al de otras aplicaciones desarrolladas sobre este conjunto de datos. Partiendo de estos objetivos se pueden mencionar los siguientes logros:

- Los modelos desarrollados que combinan la red convolucional ResNet-50 y los Vectores de Fisher demostraron capacidad de generalización al alcanzar un rendimiento consistente en el conjunto de prueba, con un mAP promedio de 0.6 y un AUC de 0.8. Estos resultados confirman que las arquitecturas propuestas logran identificar patrones para diferenciar entre lesiones benignas y malignas.

- El rendimiento de los métodos propuestos se encuentran en el orden de magnitud de clasificación de las aplicaciones desarrolladas. Si se comparan los modelos propuestos con otras aplicaciones que se realizaron y evaluaron en el mismo conjunto de datos es posible observar que el desarrollo realizado logra resultados del mismo orden de magnitud en las métricas AUC, *mean Average Precision* y *Accuracy*.

En cuanto a los desafíos no resueltos a lo largo del trabajo, se pueden mencionar los siguientes puntos:

- Los resultados obtenidos con los descriptores de imágenes GLCM, LBP y HOG distan de lo esperado para un modelo de machine learning que pueda integrarse en una aplicación real. Los modelos podrían estar sobreajustado a los datos de entrenamiento, ya que su desempeño en conjuntos de validación fue significativamente inferior.

- La estrategia de *data augmentation*, que incluía rotaciones y traslaciones, demostró un impacto limado en el resultado final. Esta limitación se evidencia claramente en el máximo valor de mAP alcanzado 0.6446.

- El desbalanceo de clases no pudo ser resuelto. A pesar de implementar técnicas de ponderación al momento de entrenar los modelos, el *Recall* menor o igual a 0.5 en todos los modelos nos indica que la desproporción de datos afectó significativamente la precisión al identificar las lesiones de piel malignas.

- El máximo valor de mAP alcanzado equivalente a 0.6446, si bien se sitúa dentro del rango reportado en estudios recientes, ver Cuadro 6, evidencia un margen de mejora significativo para alcanzar los umbrales requeridos en entornos clínicos reales.

Como conclusión final, se determina que la arquitectura ResNet-50 representa la mejor opción para la detección de melanomas, destacándose por su capacidad para minimizar falsos negativos, un criterio clínico prioritario, dado que pasar por alto un melanoma tiene consecuencias potencialmente graves para los pacientes. Para garantizar su efectividad en entornos clínicos reales, se recomienda aumentar el tamaño de la muestra de datos especialmente la cantidad de imágenes de melanomas para tener una muestra balanceada. En el caso de no ser posible equilibrar los tipos de lesiones de piel, es importante considerar técnicas de balanceo de clases durante el entrenamiento. Esto mejorará la generalización del modelo, asegurando un desempeño robusto en entornos reales.

6.2 Trabajos futuros

Basado en los desafíos identificados, se proponen las siguientes líneas de acción para investigaciones futuras:

- Explorar técnicas avanzadas de aumento de datos. Con el objetivo de ampliar la muestra de datos utilizados especialmente en las imágenes de melanomas seria interesare experimentar síntesis de imágenes con GANs (*Generative Adversarial Networks*) para generar muestras realistas de melanomas.

- Implementar funciones de pérdida para clases desbalanceadas. Algunas alternativas a considerar podrían ser *Focal Loss* o *Dice Loss* para ponderar las imágenes de melanoma.

- Implementar SMOTE (*Synthetic Minority Oversampling Technique*) sobre los Vectores de Fisher correspondientes a lesiones malignas, con el objetivo de generar ejemplos sintéticos que equilibren la distribución de clases en el conjunto de entrenamiento.

Por otro lado se podrían explorar las siguientes líneas de trabajo futuro para el desarrollo de modelos que permitan mejorar el rendimiento de los propuestos en este trabajo:

- Cantidad de muestras tomadas en una imagen. Al momento de entrenar el modelo ResNet-50-FV-CM se utilizaron dos variables constantes que fueron la cantidad de muestras de una imagen para entrenar los modelos de PCA y GMM, y la cantidad de muestras tomadas para calcular los Vectores de Fisher. En este trabajo no se analizó el impacto de distintas cantidades de muestras tomadas a partir de una imagen en el rendimiento del modelo. Una alternativa de análisis sería investigar esta línea y comprobar si de esta manera se puede mejorar la performance de la arquitectura de este modelo.

- Utilizar otras redes convolucionales para el problema de clasificación de lesiones de piel con aplicaciones de Vectores de Fisher. Otro camino de investigación es utilizar otras redes convolucionales diferentes a la ResNet-50, ejemplos de otras arquitecturas convolucionales a utilizar son Swin Transformer [54] y ConvNeXt [55].

- Experimentar si las técnicas de procesamiento de imágenes como ser *image thresholding* o aplicando filtro Sobel mejoran el rendimiento de los modelos.

6.3 Repositorio del proyecto

El desarrollo del proyecto se encuentra almacenado en el siguiente repositorio de GitLab:

`https://gitlab.com/clsalto/variantes-vectores-fisher`

7 Referencias

[1] Vanessa Gray-Schopfer, Claudia Wellbrock, and Richard Marais. Melanoma biology and new target therapy. *Nature*, 445:851–7, 03 2007.

[2] Stephanie Carr, Christy Smith, and Jessica Wernberg. Epidemiology and risk factors of melanoma. *Surgical Clinics*, 100(1):1–12, Feb 2020.

[3] Christina Ring, Nathan Cox, and Jason B. Lee. Dermatoscopy. *Clinics in Dermatology*, 2021.

[4] Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE Transactions on Medical Imaging*, 36(4):994–1004, 2017.

[5] Z. Yu, D. Ni, S. Chen, J. Qin, S. Li, T. Wang, and B. Lei. Hybrid dermoscopy image classification framework based on deep convolutional neural network and fisher vector. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 301–304, 2017.

[6] Zhen Yu, Xudong Jiang, Tianfu Wang, and Baiying Lei. Aggregating deep convolutional features for melanoma recognition in dermoscopy images. In *Machine Learning in Medical Imaging*, volume 10541, pages 238–246, 09 2017.

[7] Z. Yu, X. Jiang, F. Zhou, J. Qin, D. Ni, S. Chen, B. Lei, and T. Wang. Melanoma recognition in dermoscopy images via aggregated deep convolutional features. *IEEE Transactions on Biomedical Engineering*, 66(4):1006–1016, 2019.

[8] Jianpeng Zhang, Yutong Xie, Qi Wu, and Yong Xia. Medical image classification using synergic deep learning. *Medical Image Analysis*, 54:10–19, 2019.

[9] Gastón Liberman, Daniel Acevedo, and Marta Mejail. Classification of melanoma images with fisher vectors and deep learning. In *Iberoamerican Congress on Pattern Recognition (CIARP)*, pages 732–739, 2019.

[10] Zhen Yu, Feng Jiang, Feng Zhou, Xinzi He, Dong Ni, Siping Chen, Tianfu Wang, and Baiying Lei. Convolutional descriptors aggregation via cross-net for skin lesion recognition. *Applied Soft Computing*, 92:106281, 2020.

[11] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, and Inso Kweon. Fisher kernel for deep neural activations. *arXiv:1412.1628 e-prints*, 12 2014.

[12] Guo-Sen Xie, Xu-Yao Zhang, Shuicheng Yan, and Cheng-Lin Liu. Hybrid cnn and dictionary-based models for scene recognition and domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1263–1274, 2017.

[13] Boyang Li, Weihua Su, Hang Wu, Ruihao Li, Wenchang Zhang, Wei Qin, and Shiyue Zhang. Aggregated deep fisher feature for vhr remote sensing scene classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(9):3508–3523, 2019.

[14] Mryka Hall-Beyer. Glcm texture: A tutorial v. 3.0 march 2017. *National Council on Geographic Information and Analysis Remote Sensing Core Curriculum*, 03 2017.

[15] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.

[16] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[17] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.

[18] Maria M. P. Petrou and Sei-Ichiro Kamata. *Image Processing: Dealing with Texture*. Wiley, 2021.

[19] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[21] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[22] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs*, 2, 2010.

[23] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[24] D H Hubel and T N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195:215–243, 1968.

[25] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629, Aug 2018.

[26] Farhana Sultana, A. Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129, 11 2018.

[27] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[29] D Stutz. Seminar report: understanding convolutional neural networks. *Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII*, 2014.

[30] Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[31] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 12 1964.

[32] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 07 2011.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[36] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[37] Jorge Sánchez, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105, 12 2013.

[38] Shun-Ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry (Translations of Mathematical Monographs) (Tanslations of Mathematical Monographs)*. American Mathematical Society, 04 2007.

[39] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NIPS'98, page 487–493, Cambridge, MA, USA, 1998. MIT Press.

[40] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York, 1985.

[41] Josip Krapac, Jakob Verbeek, and Frederic Jurie. Modeling spatial layout with fisher vectors for image categorization. In *2011 International Conference on Computer Vision*, pages 1487–1494. Conferencia ICCV, 11 2011.

[42] N. Smith and Mark Gales. Speech recognition using svms. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[43] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision – ECCV 2010*, pages 143–156, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[44] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[45] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[46] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[47] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[48] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[49] David Gutman, Noel C. F. Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic), 2016.

[50] Isabelle Guyon, Kristin Bennett, Gavin Cawley, Hugo Jair Escalante, Sergio Escalera, Tin Kam Ho, Núria Macià, Bisakha Ray, Mehreen Saeed, Alexander Statnikov, et al. Design of the 2015 chalearn automl challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[51] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4):312–315, 2020.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[53] Sašo Džeroski, Panče Panov, and Bernard Ženko. Ensemble methods in machine learning. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, volume 6, pages 5317–5325. Springer, 2009.

[54] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[55] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

A Apéndice

A.1 Experimentos ResNet-50 Fine tuning

En este apéndice se presentan los experimentos realizados durante el *fine-tuning* de la ResNet-50. Para un análisis detallado de la metodología empleada, véase la Sección 3.5.2, donde se desarrolla el tema en profundidad.

A.1.1 Experimento Inicial

El experimento inicial evaluó el comportamiento de la función de pérdida con *learning rates* de 0.01 y 0.00001. Los resultados demostraron que con valores altos de *learning rate*, la función de pérdida presenta oscilaciones bruscas entre épocas. Por el contrario, con un *learning rate* más bajo, la función de pérdida muestra una convergencia más estable pero lenta. Estas situaciones se reflejan en la Fig. 37.

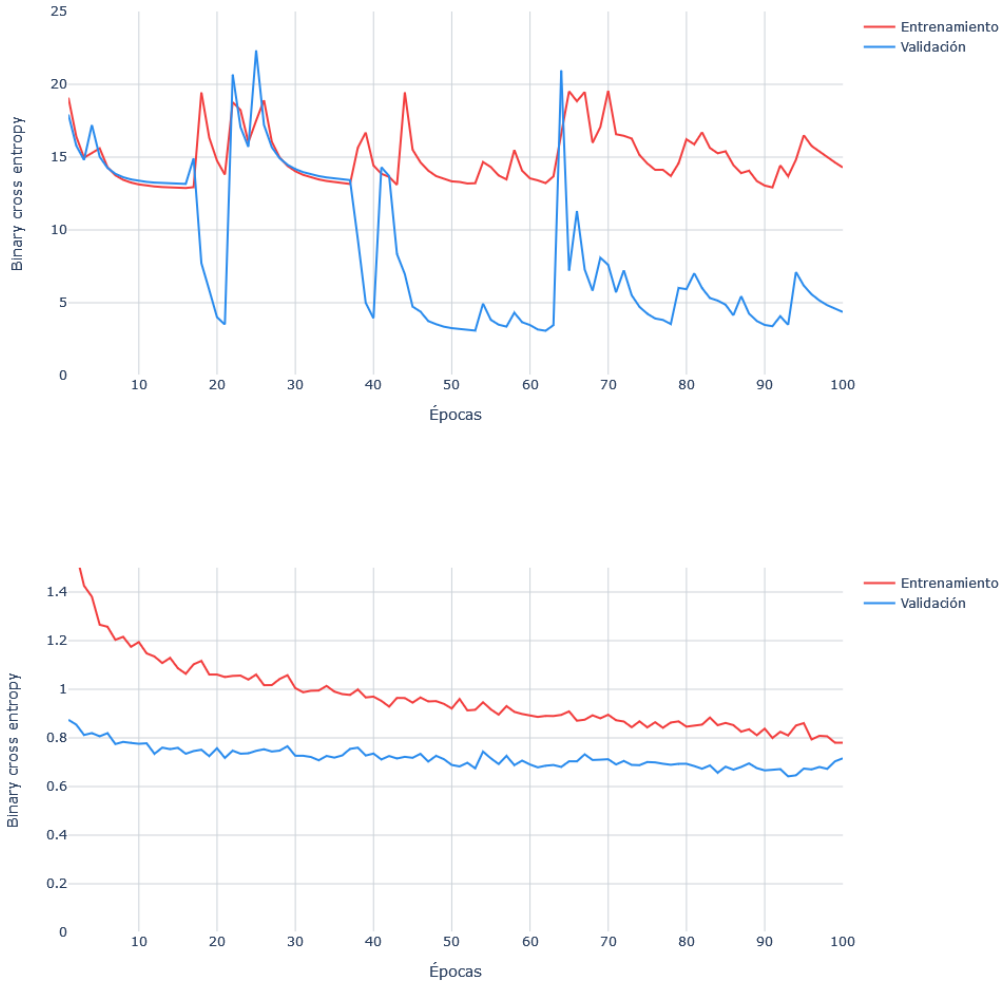


Figura 37: *Binary cross entropy* para las pruebas con *learning rate* = 0.01 arriba y *learning rate* = 0.00001 abajo.

A.1.2 Experimentos con distintos parámetros

Para el entrenamiento de la ResNet-50 se utilizó la combinación de parámetros descritos en el Cuadro 7.

Cuadro 7: Parámetros de los experimentos durante el entrenamiento de la capa totalmente conectada de la red ResNet-50.

Experimento	Optimizador	Tamaño del lote	Tasa de aprendizaje
1	Adam	16	0.001
2	Adam	16	0.0001
3	Adam	32	0.001
4	Adam	32	0.0001
5	SGD	16	0.001
6	SGD	16	0.0001
7	SGD	32	0.001
8	SGD	32	0.0001

A continuación se presentan los resultados de cada experimento, mostrando el *Accuracy* y el *Binary cross entropy* obtenidos en cada época.

Experimento 1
$optimizer = adam$
$batch\ size = 16$
$learning\ rate = 0.001$

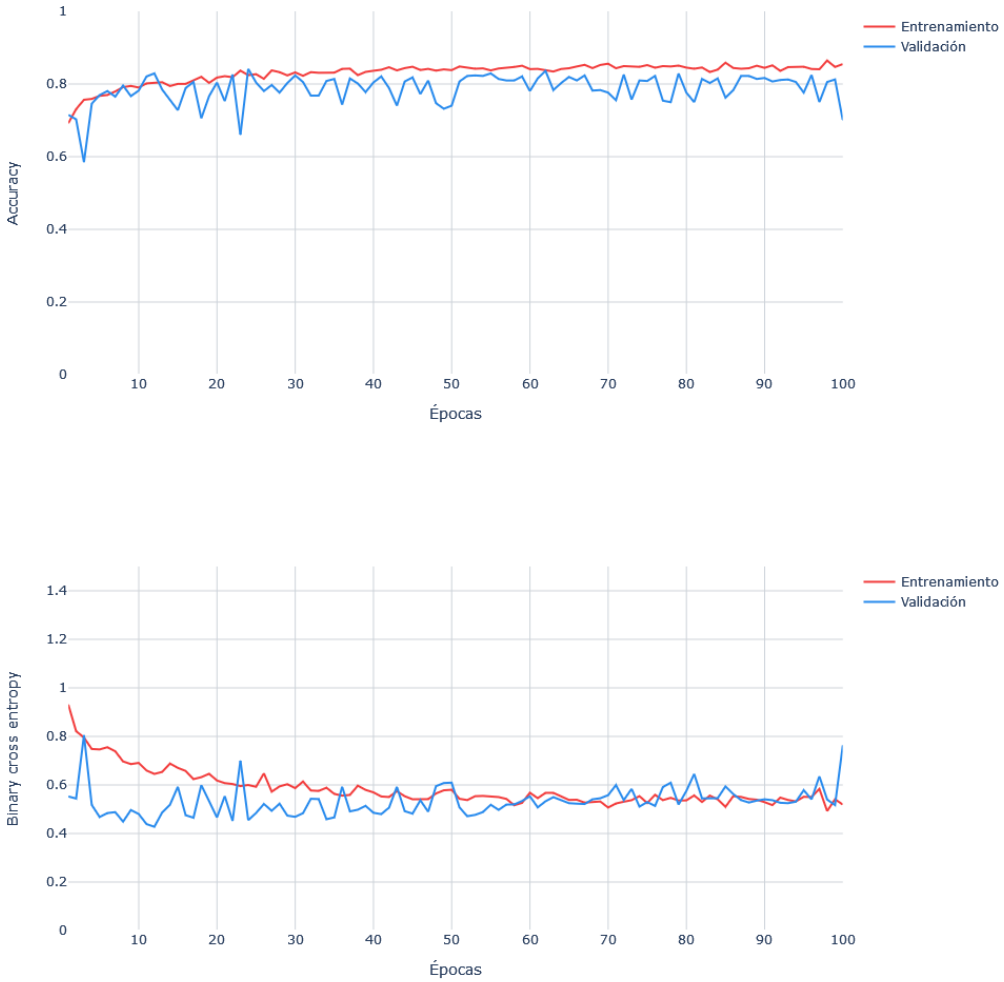


Figura 38: *Accuracy* y *Binary cross entropy* para el experimento 1.

Experimento 2
optimizer = *adam*
batch size = 16
learning rate = 0.0001

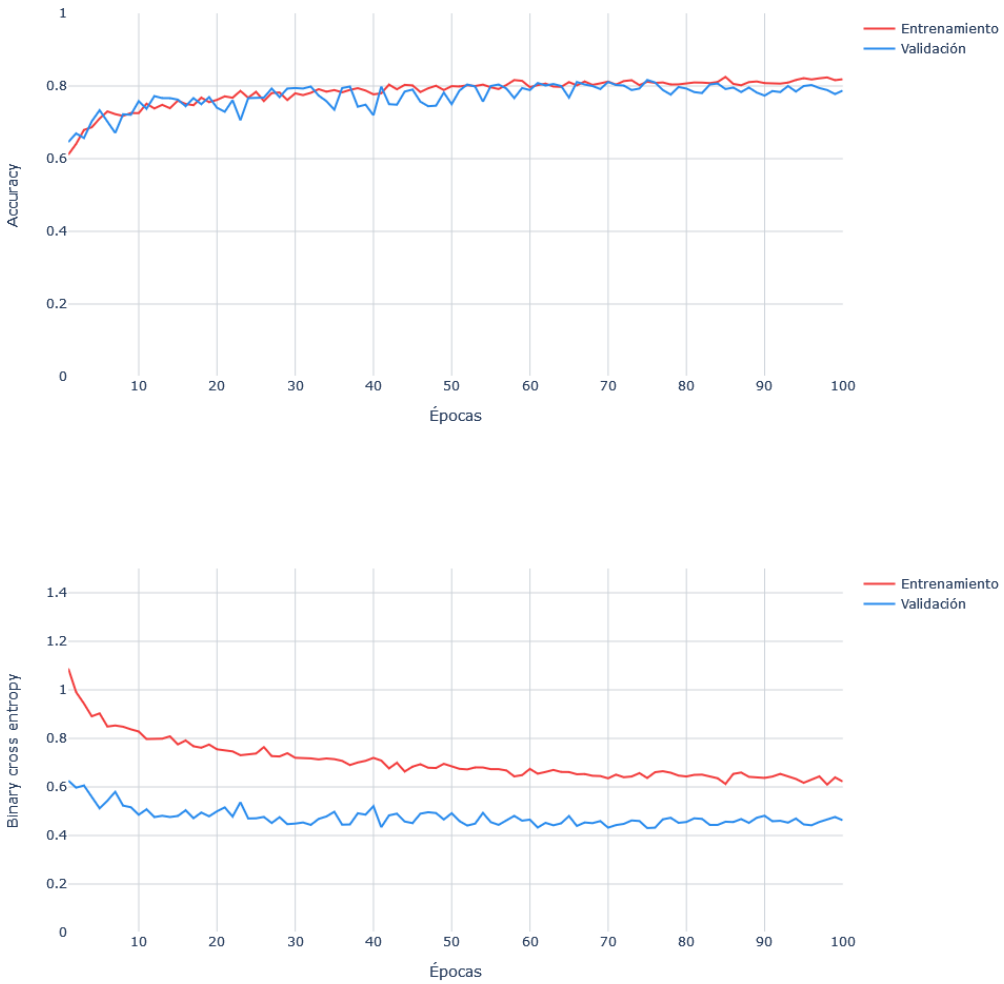


Figura 39: *Accuracy* y *Binary cross entropy* para el experimento 2.

Experimento 3
optimizer = *adam*
batch size = 32
learning rate = 0.001

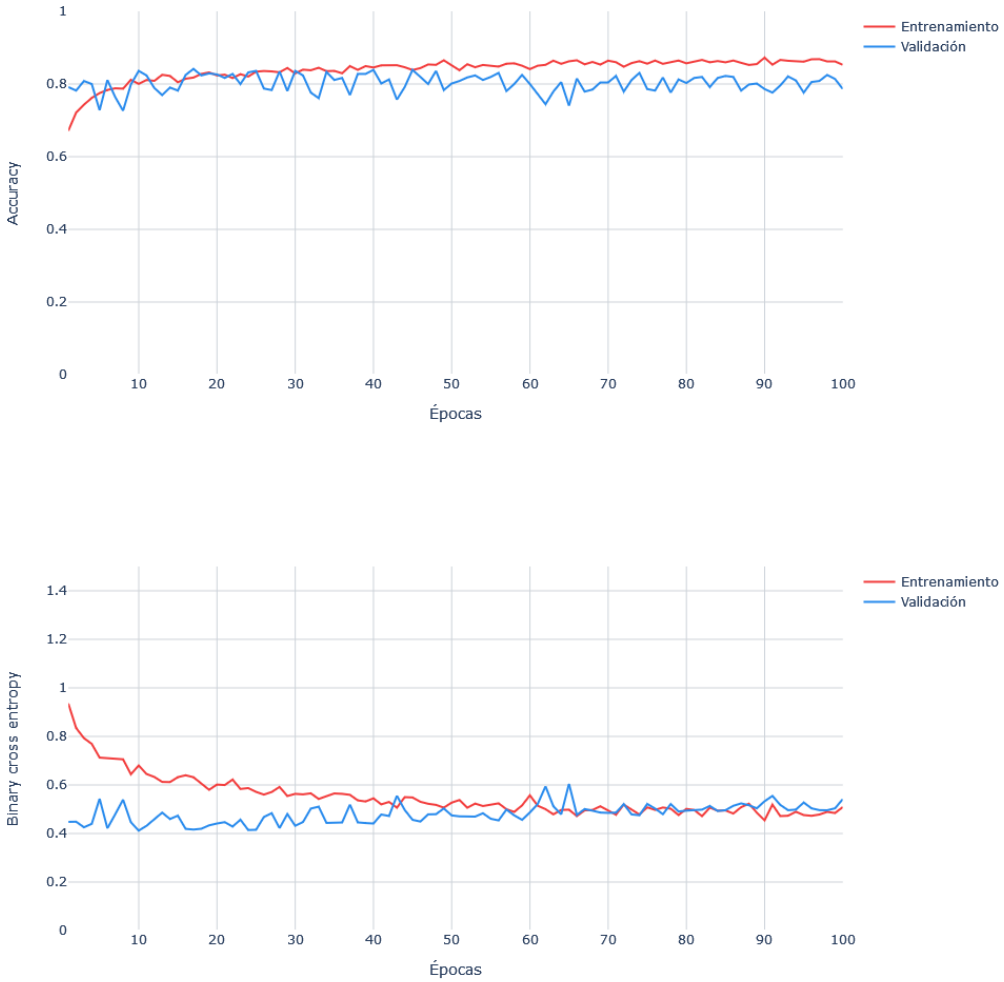


Figura 40: *Accuracy* y *Binary cross entropy* para el experimento 3.

Experimento 4
optimizer = *adam*
batch size = 32
learning rate = 0.0001

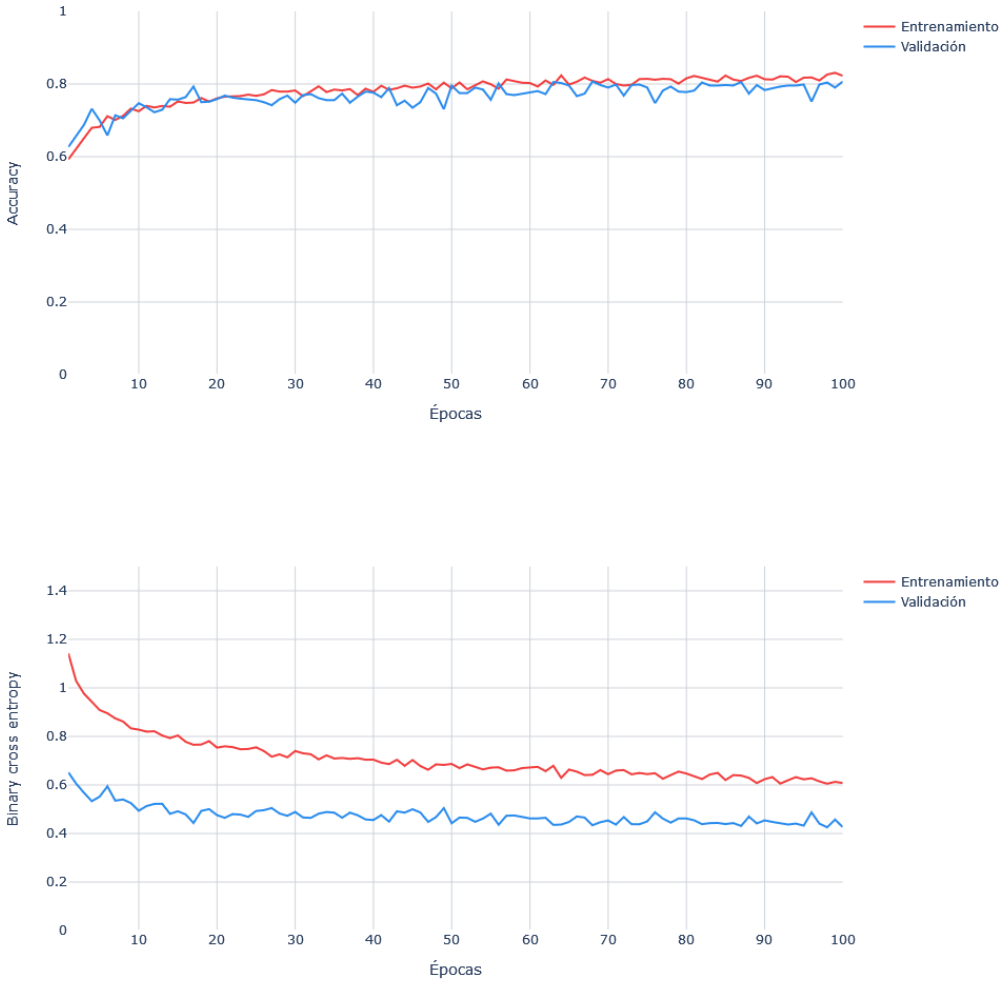


Figura 41: *Accuracy* y *Binary cross entropy* para el experimento 4.

Experimento 5
optimizer = *sgd*
batch size = 16
learning rate = 0.001

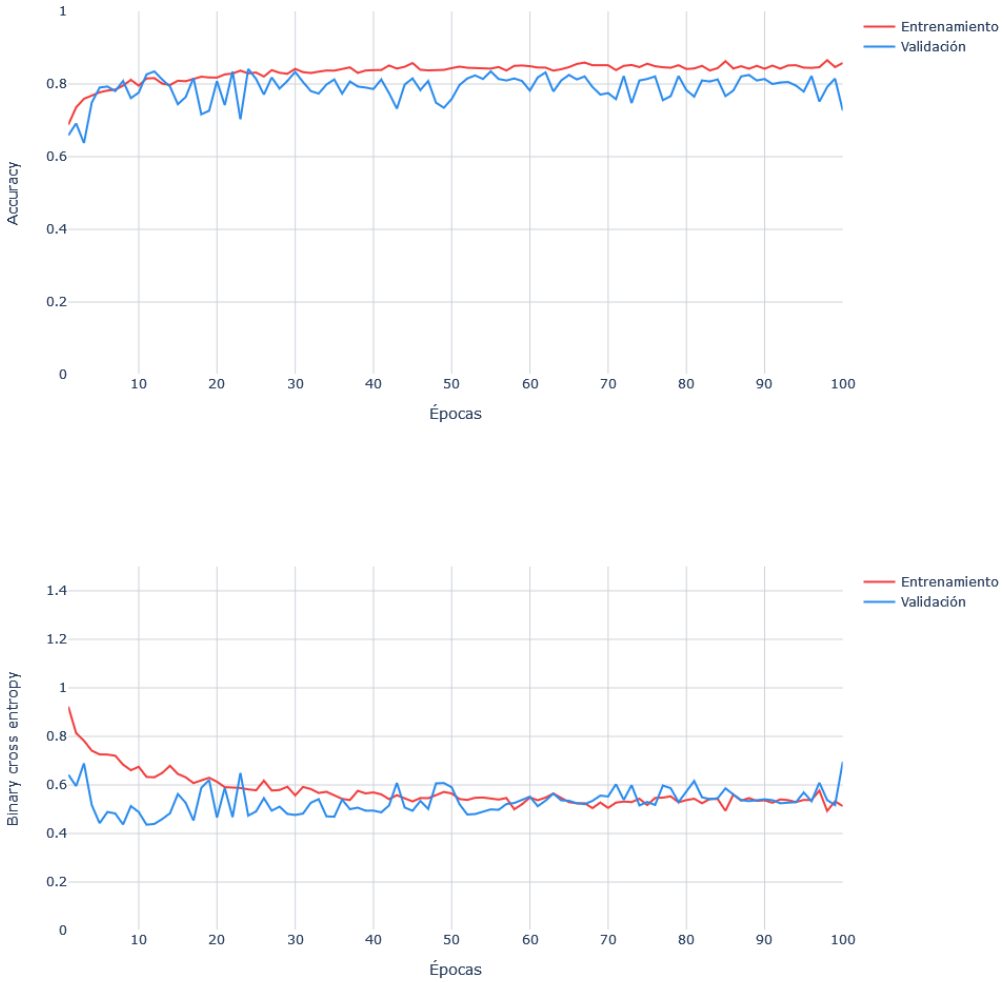


Figura 42: *Accuracy* y *Binary cross entropy* para el experimento 5.

Experimento 6
optimizer = *sgd*
batch size = 16
learning rate = 0.0001

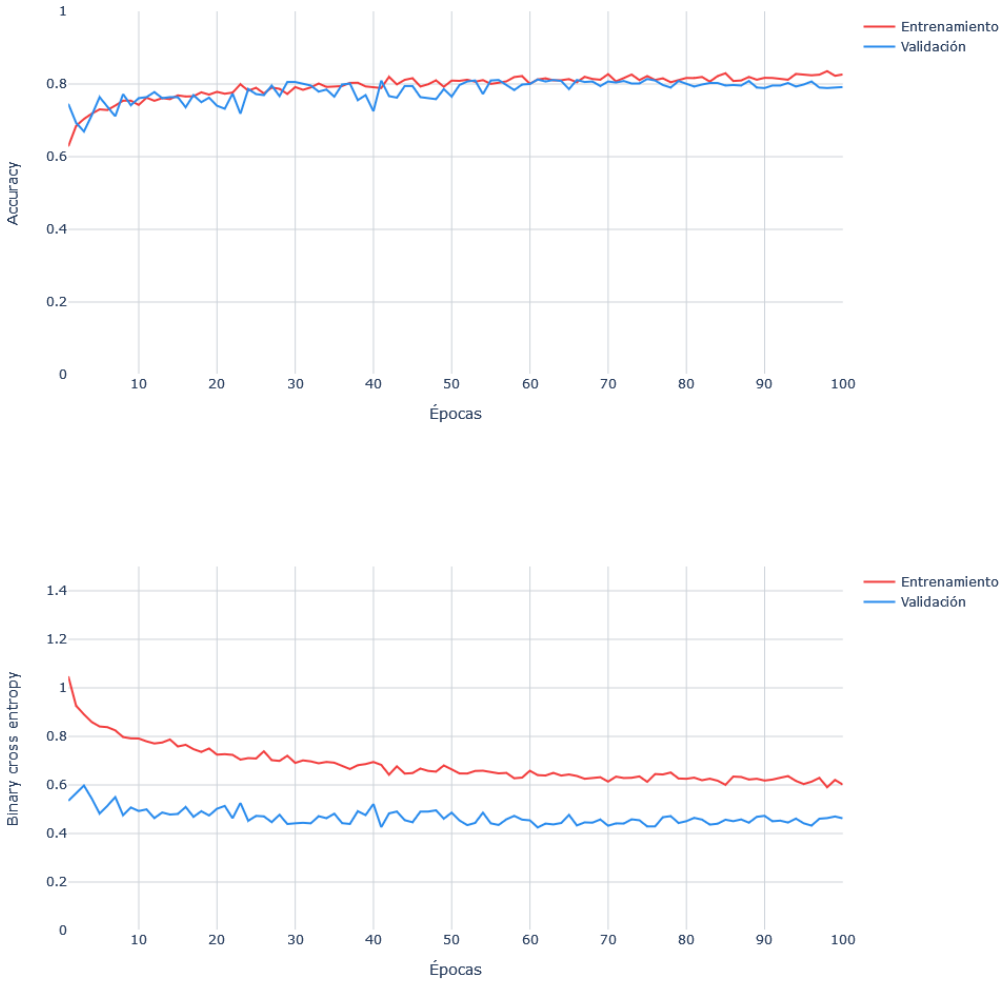


Figura 43: *Accuracy* y *Binary cross entropy* para el experimento 6.

Experimento 7
optimizer = *sgd*
batch size = 32
learning rate = 0.001

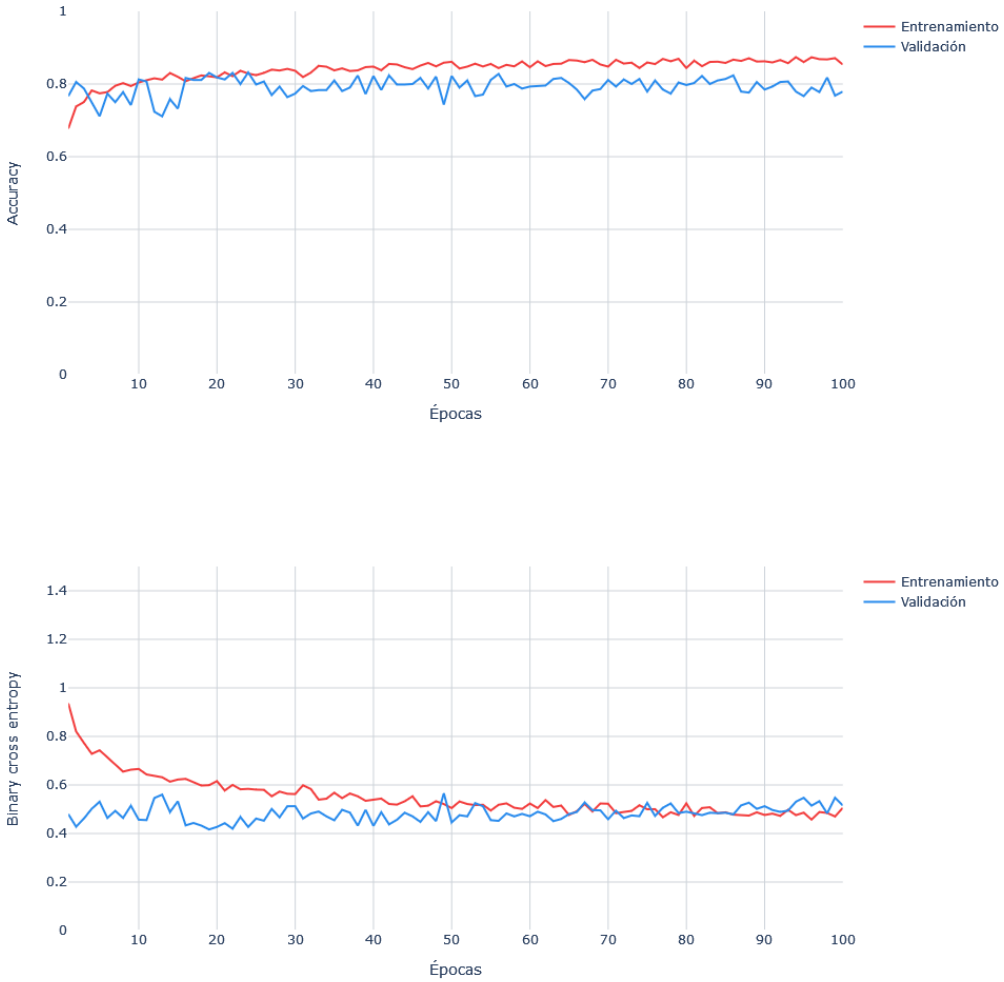


Figura 44: *Accuracy* y *Binary cross entropy* para el experimento 7.

Experimento 8
optimizer = *sgd*
batch size = 32
learning rate = 0.0001

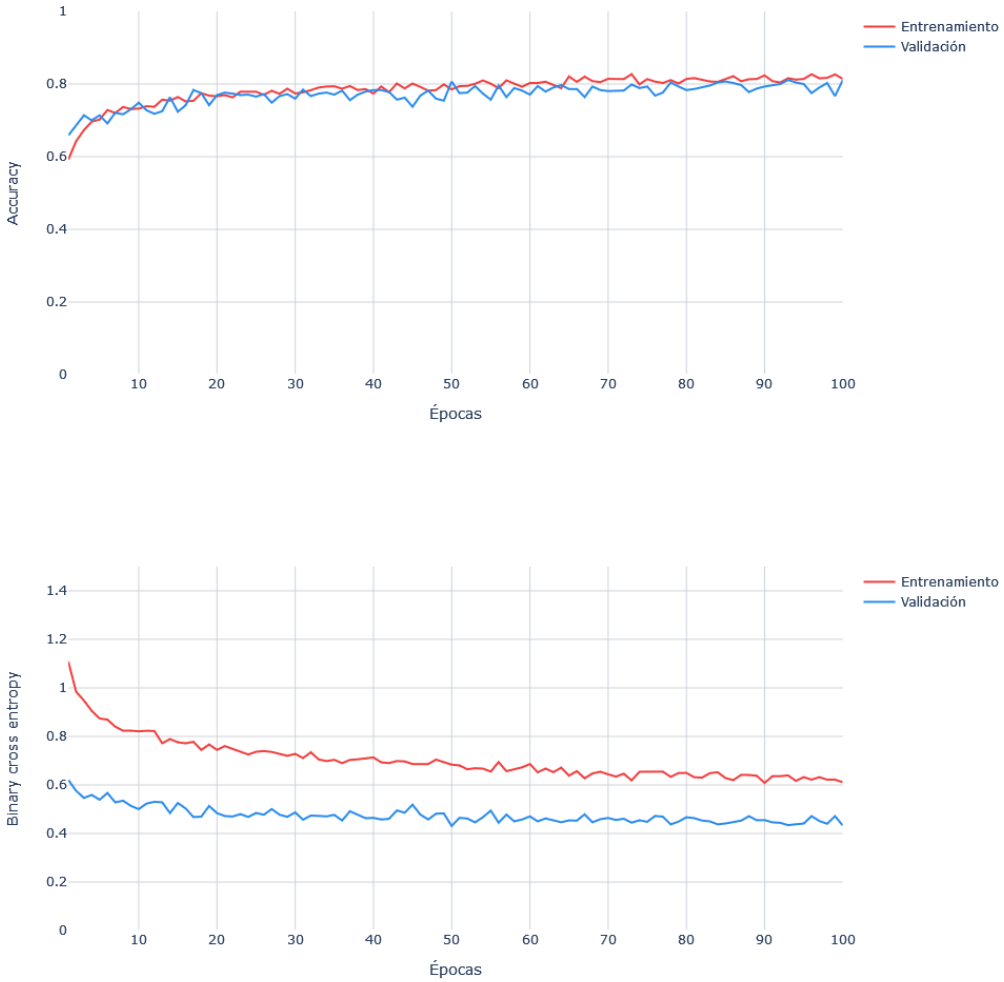


Figura 45: *Accuracy* y *Binary cross entropy* para el experimento 8.

A.1.3 Fine tunning de la red convolucional

En la Fig. 46 se presenta el *Accuracy* y *Binary cross entropy* para el entrenamiento de todas las épocas de la red convolucional. El modelo base utilizado fue el obtenido en el experimento 4, al cual se aplicó una estrategia de descongelamiento progresivo de pesos distribuida en cuatro etapas consecutivas. En la primera se descongelaron los pesos del quinto bloque convolucional, en la segunda los pesos del cuarto bloque convolucional, en la tercera los pesos del tercer bloque convolucional y finalmente, en la última etapa, se entreno la red completa. En cada etapa se realizó un entrenamiento durante 100 épocas.

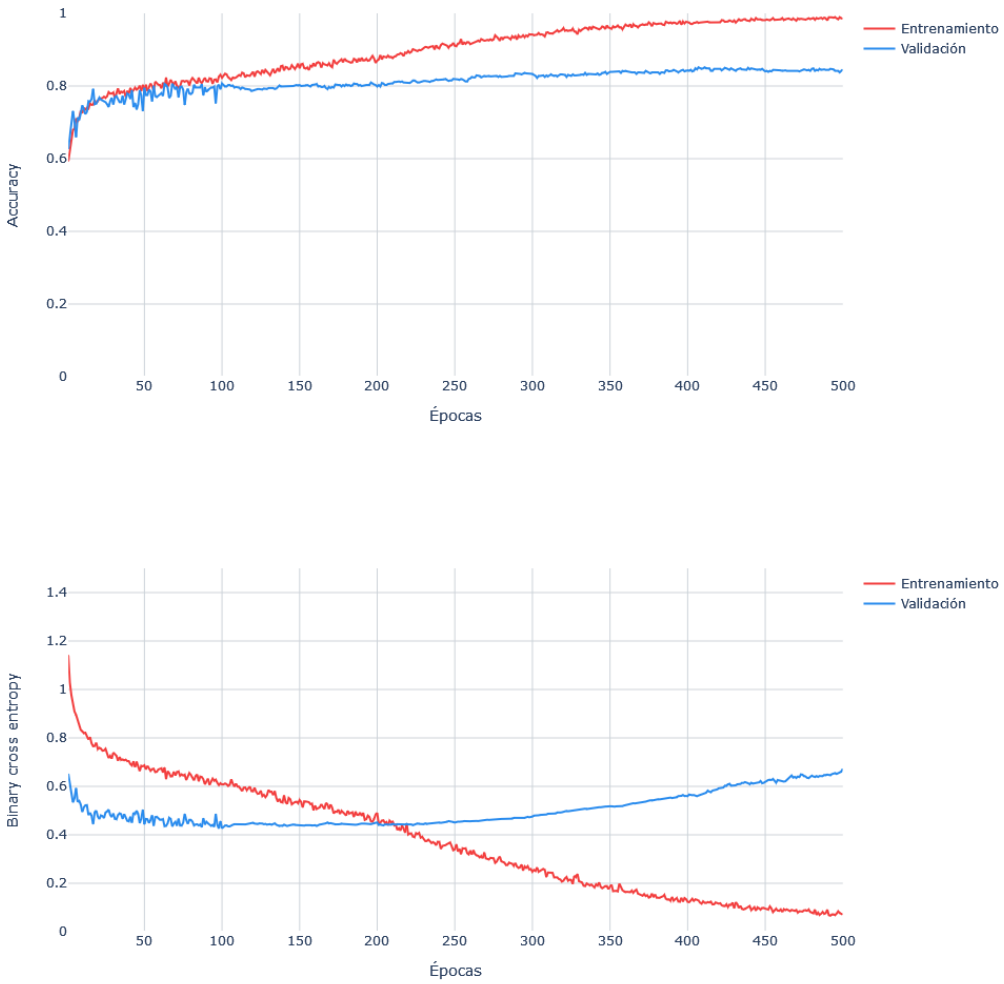


Figura 46: *Accuracy* y *Binary cross entropy* para el entrenamiento completo de la red ResNet-50.

A.2 Parámetros de los experimentos con Vectores de Fisher

El Cuadro 8 muestra los valores de los parámetros correspondientes a la cantidad de componentes principales y a la cantidad de componentes gaussianas de los distintos experimentos realizados con Vectores de Fisher.

Cuadro 8: Parámetros de los experimentos durante el entrenamiento con Vectores de Fisher.

Experimento	Cantidad de componentes principales	Cantidad de componentes gaussianas
1	32	16
2	32	32
3	32	64
4	32	100
5	64	16
6	64	32
7	64	64
8	64	100
9	128	16
10	128	32
11	128	64
12	128	100
13	256	16
14	256	32
15	256	64
16	256	100
17	384	16
18	384	32
19	384	64
20	384	100
21	512	16
22	512	32
23	512	64
24	512	100

A.3 Experimentos Vectores de Fisher con muestras

Este apéndice detalla los experimentos de entrenamiento de un clasificador SVM utilizando Vectores de Fisher generados a partir de descriptores extraídos de diferentes muestras de una imagen. Los fundamentos metodológicos se detallan en la Sección 3.6.1. Los resultados mostrados en este apéndice reflejan el rendimiento en los conjuntos de prueba durante el proceso de *cross validation*.

A.3.1 Resultados por número de componentes principales y número de componentes gaussianas

En primer lugar se evalúa el rendimiento del modelo en función de dos parámetros, el número de componentes principales y el número de componentes gaussianas. Las Figuras 47, 48 y 49 muestran la distribución de las métricas AUC, *mean Average Precision* y *Accuracy*, en función de los parámetros inicialmente mencionados.

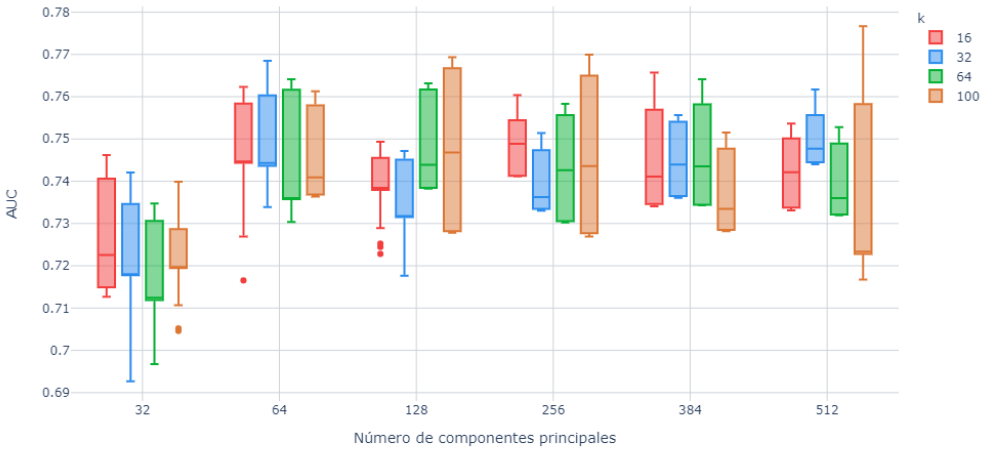


Figura 47: Distribución del AUC promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

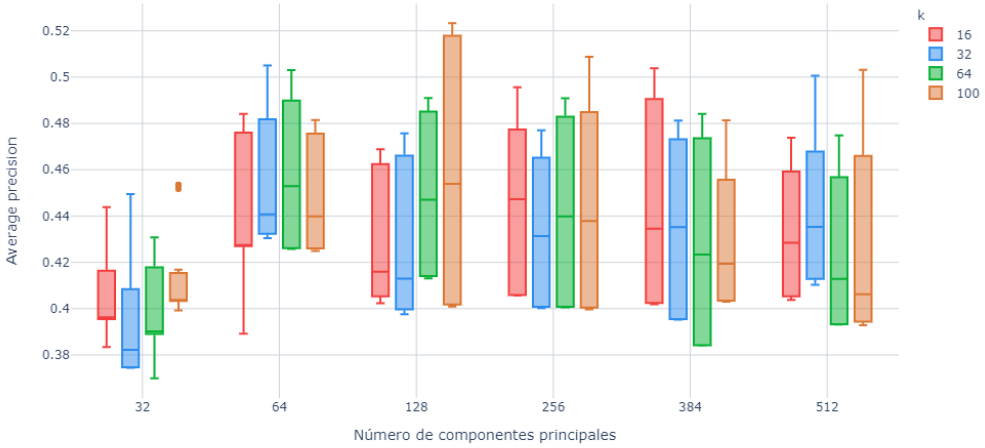


Figura 48: Distribución del *mean Average Precision* promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

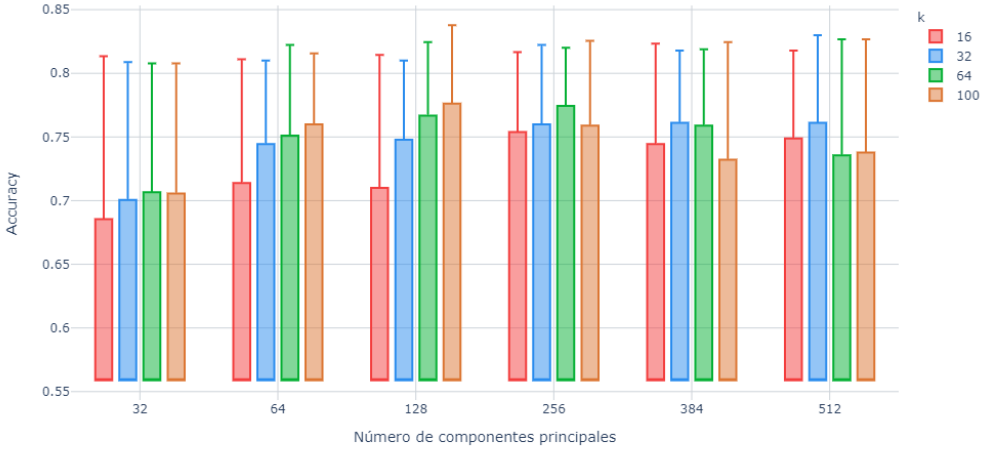


Figura 49: Distribución del *Accuracy* promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

A.3.2 Resultados por parámetro C con 64 componentes principales y 16 componentes gaussianas

Utilizando como base los modelos entrenados con 64 componentes principales y 16 componentes gaussianas, el análisis se centra ahora en evaluar el impacto del parámetro C del clasificador SVM. Las Figuras 50, 51 y 52 presentan la distribución comparativa de las métricas de evaluación para distintos valores de este parámetro.

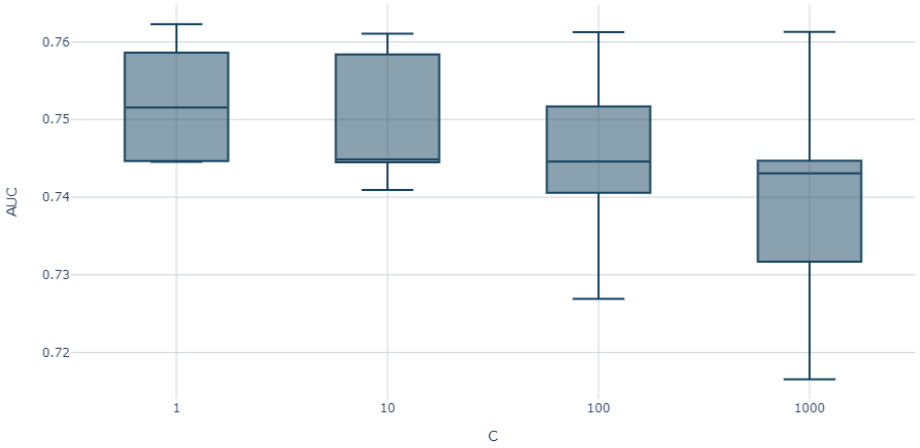


Figura 50: Distribución del AUC promedio en el conjunto de prueba por parámetro C del clasificador SVM.

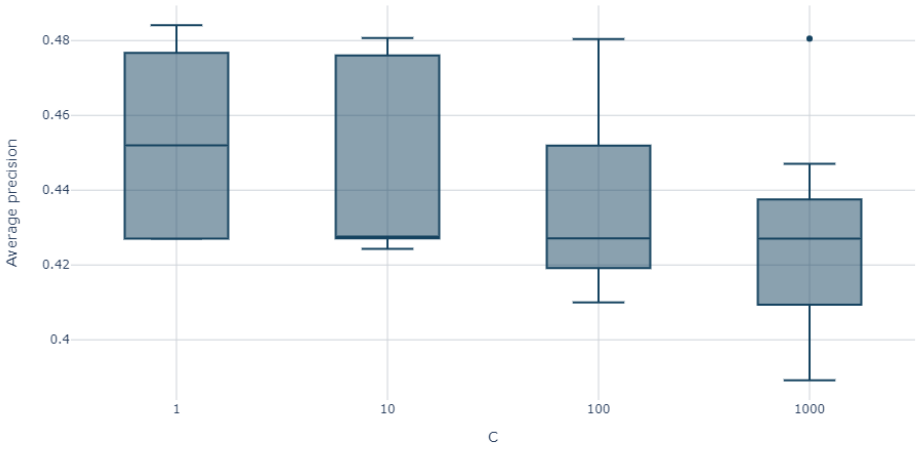


Figura 51: Distribución del *mean Average Precision* promedio en el conjunto de prueba por parámetro C del clasificador SVM.

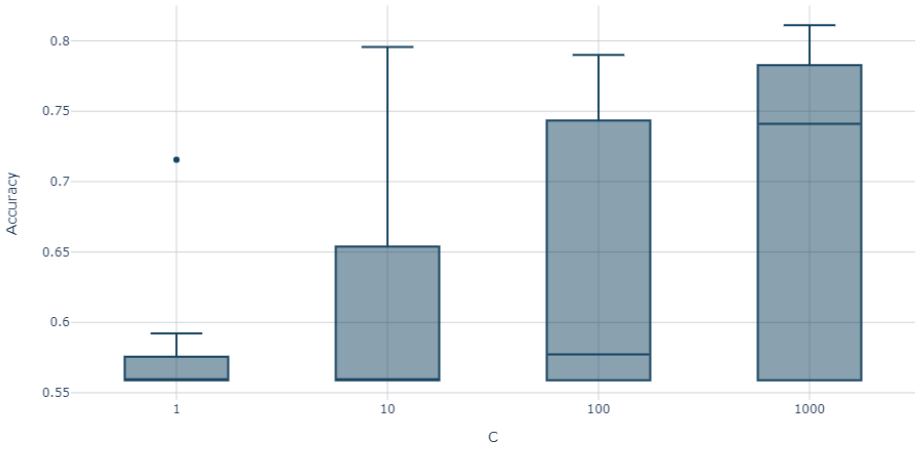


Figura 52: Distribución del *Accuracy* promedio en el conjunto de prueba por parámetro C del clasificador SVM.

A.3.3 Resultados por parámetros *gamma* y *kernel* con 64 componentes principales, 16 componentes gaussianas y $C = 1$

Por último, se presenta el rendimiento de los modelos a partir de los parámetros *kernel* y *gamma* del clasificador SMV.

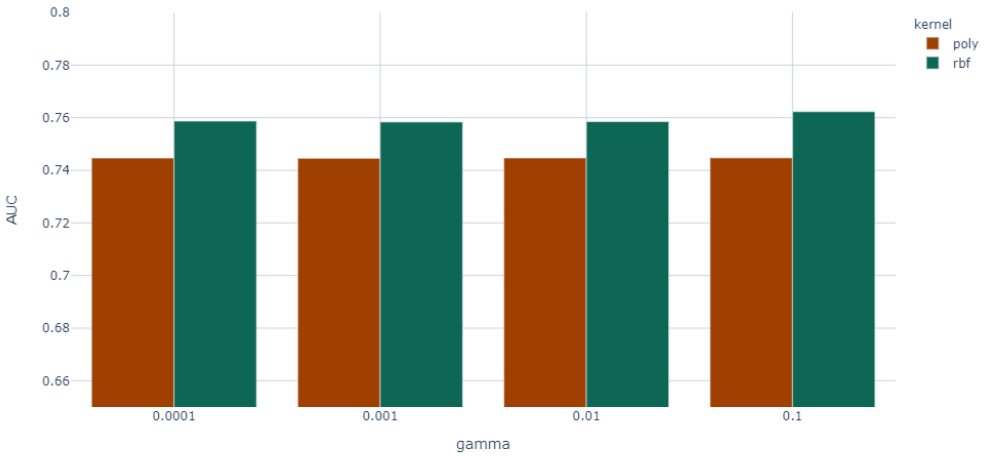


Figura 53: AUC por parámetros *kernel* y *gamma* del clasificador SVM.

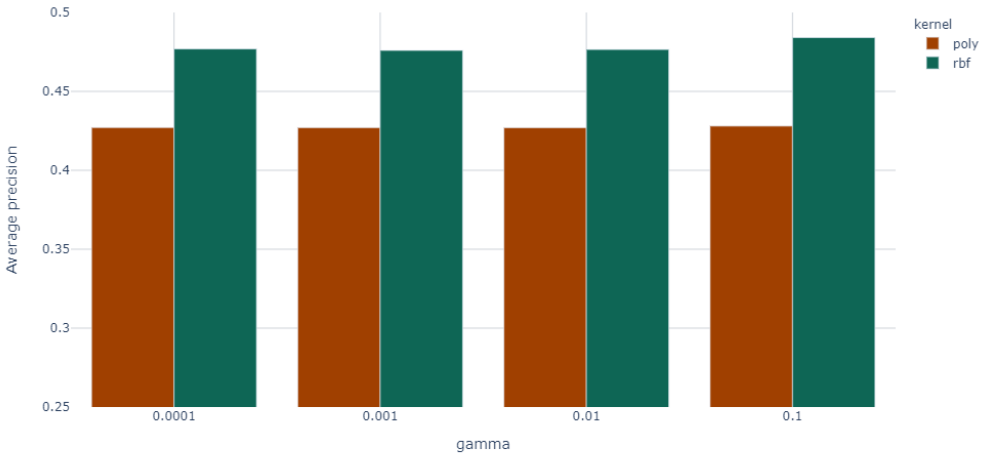


Figura 54: *Mean Average Precision* por parámetros *kernel* y *gamma* del clasificador SVM.

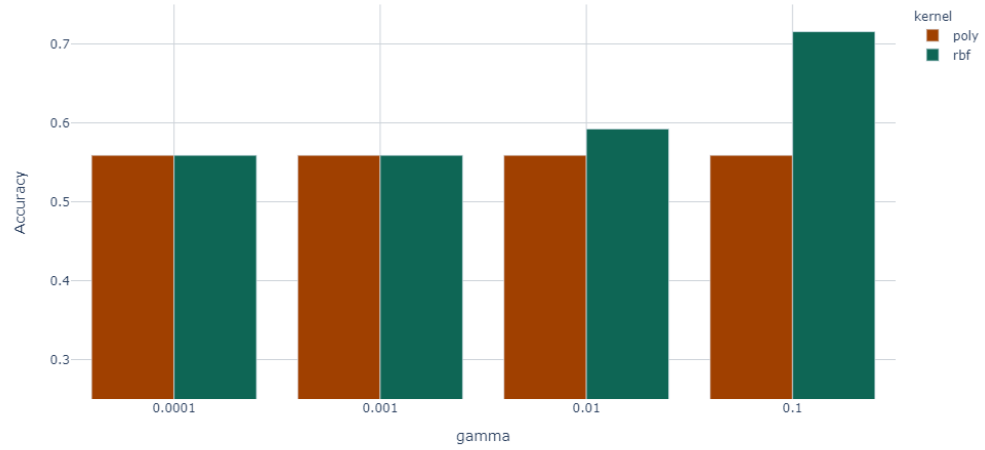


Figura 55: *Accuracy* por parámetros *kernel* y *gamma* del clasificador SVM.

A.4 Experimentos Vectores de Fisher con descriptores

A.4.1 Resultados por número de componentes principales y número de componentes gaussianas

Inicialmente se evalúa el impacto del número de componentes principales y el número de componentes gaussianas. Las Figuras 56, 57 y 58 muestran la distribución de las métricas AUC, *mean Average Precision* y *Accuracy*, en función de los parámetros anteriormente mencionados.

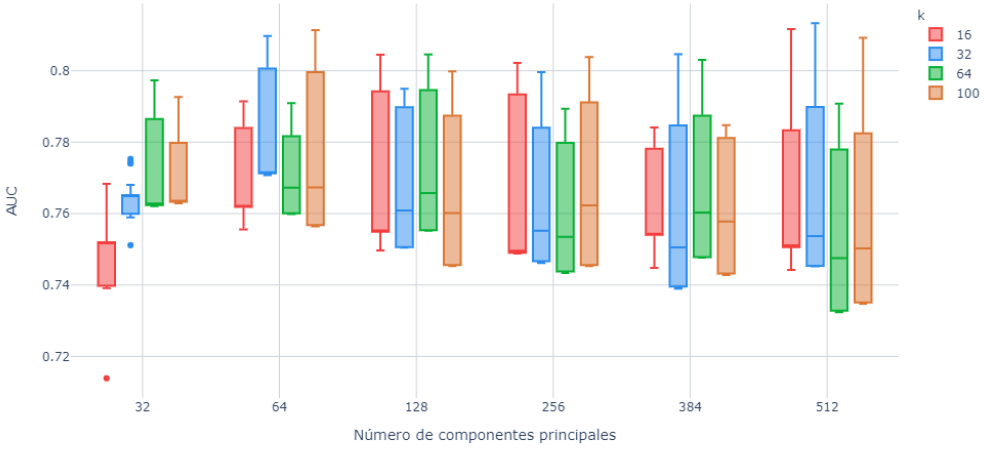


Figura 56: Distribución del AUC promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

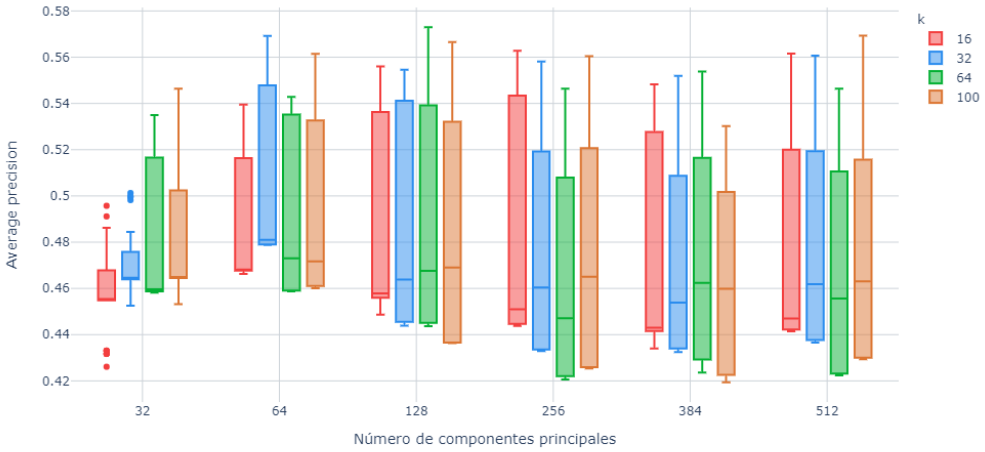


Figura 57: Distribución del *mean Average Precision* promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

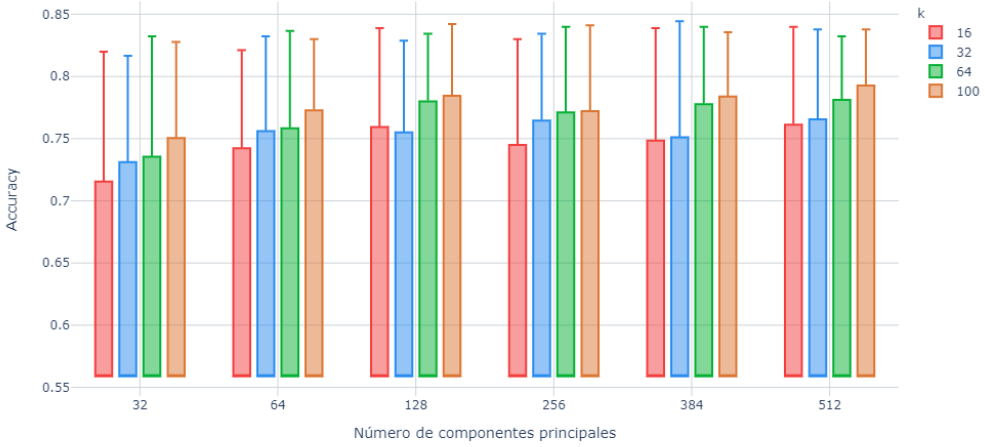


Figura 58: Distribución del *Accuracy* promedio en el conjunto de prueba por número de componentes principales y número de componentes gaussianas (k).

A.4.2 Resultados por parámetro *gamma* con 64 componentes principales y 32 componentes gaussianas

Para los modelos con 64 componentes principales y 32 componentes gaussianas, se presentan a continuación las distribuciones de las métricas AUC, *mean Average Precision* y *Accuracy* en función de los valores de *gamma*.

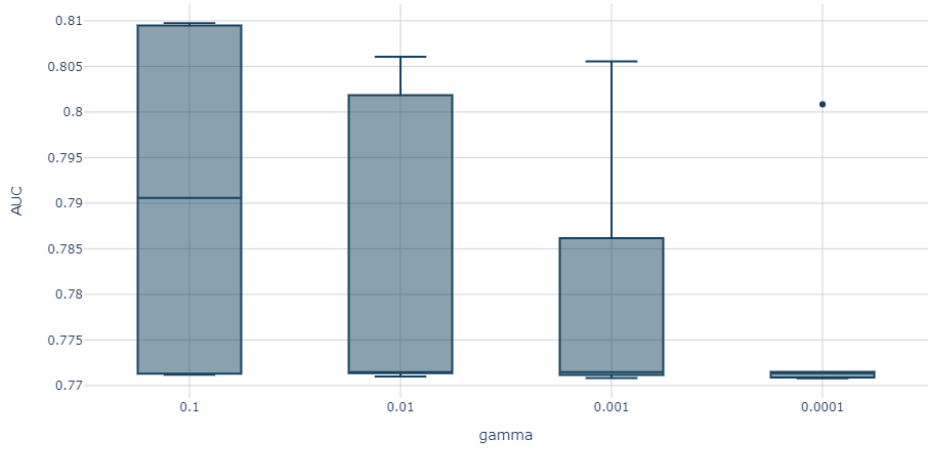


Figura 59: Distribución del AUC promedio en el conjunto de prueba por parámetro *gamma* del clasificador SVM.

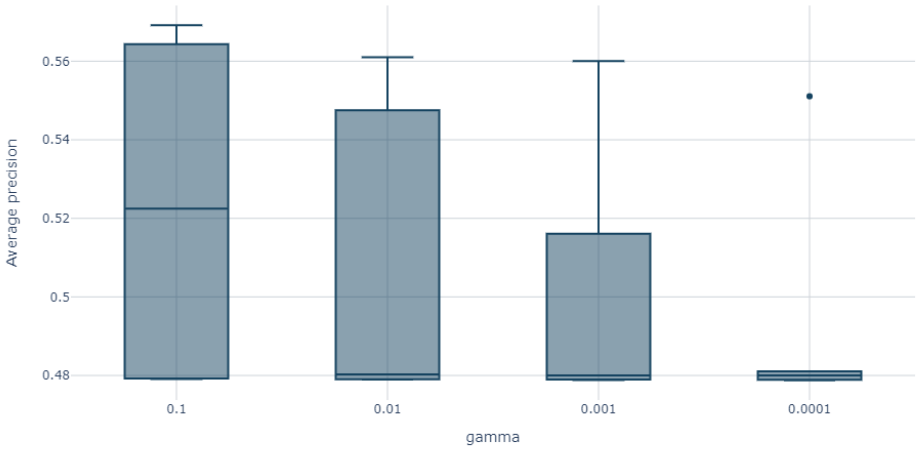


Figura 60: Distribución del *mean Average Precision* promedio en el conjunto de prueba por parámetro *gamma* del clasificador SVM.

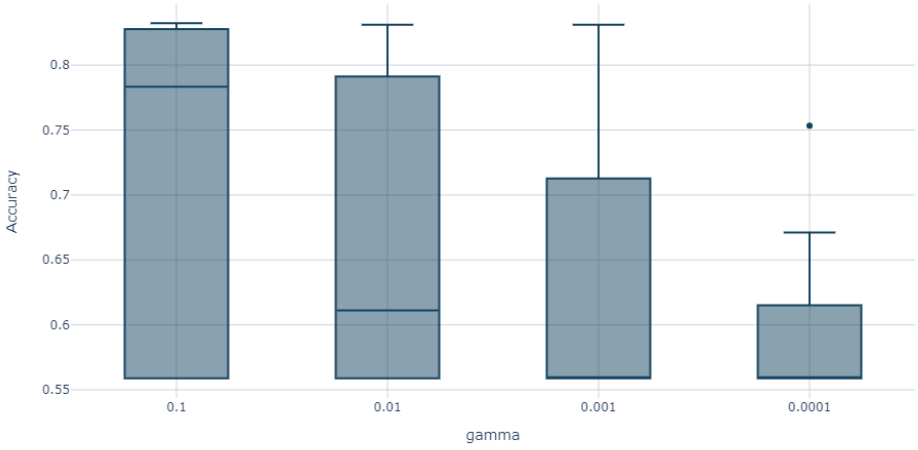


Figura 61: Distribución del *Accuracy* promedio en el conjunto de prueba por parámetro *gamma* del clasificador SVM.

A.4.3 Resultados por parámetros C y *kernel* con 64 componentes principales, 32 componentes gaussianas y $gamma = 0.1$

Finalmente, se compara el rendimiento de los modelos a partir de los parámetros *kernel* y C del clasificador SVM.

Cuadro 9: Métricas para los experimentos de Fisher con descriptores en función de los parámetros *kernel* y C del clasificador SVM con $gamma = 0.1$.

Kernel	C	AUC	mean Average Precision	Accuracy
poly	1	0.771347	0.479082	0.558889
rbf	1	0.798823	0.545817	0.756667
poly	10	0.771269	0.479211	0.558889
rbf	10	**0.809738**	**0.569169**	**0.832222**
poly	100	0.771190	0.479192	0.558889
rbf	100	0.809493	0.564311	0.827778
poly	1000	0.782322	0.499093	0.810000
rbf	1000	0.809493	0.564311	0.827778