Trabajo Fin de Máster
Máster Universitario en Lógica, Computación e Inteligencia Artificial

Análsis semántico de vídeo mediante Deep Learning

Autor:
Aliaksei Kandratsenka

Tutor:
Miguel Ángel Martínez-del-Amor

Departamento de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla
Sevilla, Septiembre 2019

Trabajo Fin de Master: Análsis semántico de vídeo mediante Deep Learning

Autor: Aliaksei Kandratsenka

Tutor: Miguel Ángel Martínez-del-Amor

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificacíon de:

Sevilla, 2019

Resumen

Hoy en día vivimos en una época en la que la tecnología es una parte integra de la vida de todos y cada vez más aplicaciones y sistemas son creados con el apoyo del aprendizaje automático y la inteligencia artificial.

Este proyecto pretende mostrar cómo funcionan en día los algoritmos que permiten a los modelos de inteligencia artificial clasificar vídeo en diferentes clases.

Cabe destacas que, aunque no se ha buscado obtener un alto rendimiento el la clasificación, se ha diseñado un sistema implementado en Python3, utilizando la librería Keras, que podría marcarse como punto de partida para construir un sistema de clasificación robusto.

Abstract

Today we live in an amazing time when technology is an integral part of everyone's life and more and more applications and systems are created with the support of machine learning and artificial intelligence.

This project intends to show how algorithms work today that allow artificial intelligence models to classify video into different classes.

It should be noted that, although it has not sought to obtain high performance in classification, a system implemented in Python3 has been designed, using the Keras library, which could be marked as a starting point to build a robust classification system.

Índice

1 Introducción

Miles de películas son producidas cada año, y por tanto la comprensión automática de contenidos es importante para desarrollar sistemas de recomendación de películas, almacenamiento eficiente, asistencia en el censurado, sistemas de extracción de escenas, etc. Es por ello que se ha comenzado a desarrollar algoritmos basados en Deep Learning para la extracción automática de etiquetas de vídeos, segmentación inteligente de escenas, y categorización automática de escenas y eventos ”beat”.

Durante este trabajo se mostrará: (1) cómo se definen este tipo de problemas, (2) como funcionan los algoritmos basados en el Aprendizaje Automático [1], en especial las redes convolucionales empleadas mediante ”transfer learning” y (3) cuales han sido los resultados de las evaluaciones del sistema y las conclusiones obtenidas.

1.1 Objetivo

El objetivo de este trabajo es desarrollar un programa capaz de clasificar la semántica del vídeo en determinadas categorías de películas. En otras palabras, crear un modelo basado en el aprendizaje profundo (DL) redes neuronales convolucionales (CNN), que pueda clasificar los materiales de vídeo y dar una respuesta en forma de un porcentaje de posible pertenencia de una película a una de las categorías.

Para ello, es necesario crear etiquetas con las que clasificaremos los materiales de vídeo. En nuestro caso, las etiquetas serán las siguientes categorías de películas: Drama (*Drama*), Persecución de coches (*Chase scenes*), Tiroteo (*Scenes of shooting*), Amor y alegría (*Scenes of love and joy*), Peleas de puños (*Scenes of fights*).

La finalidad de este proyecto ha sido exponer el funcionamiento de modelos basados en DL sobre problemas de clasificación del vídeo, donde se ha preferido ofrecer una robusta fundamentación teórica del sistema, destacando los aspectos matemáticos, a cambio de perseguir un alto rendimiento en las transcripciones.

Para ello se pretende crear un sistema capaz de tomar un vídeo, crear imágenes sobre este vídeo, en forma de tensor de entrada, y devolver el porcentaje de la relación del vídeo con una clase particular de modelo entrenado.

1.2 Sub-objetivo

Para alcanzar el objetivo principal de este trabajo, es necesario los siguientes subobjetivos (ver resumen en Figura 1):

1. Crear una base de datos a partir de cero que consiste en material de vídeo correspondiente a las categorías establecidas recortadas sucesivamente en fotogramas.

2. Crear tensores a partir de imágenes obtenidas y añadir de forma numéricas a un array multidimencional de NumPy.

3. Mostrar las diferentes posibilidades de creación de redes CNN y, por lo tanto, demostrar la diferencia en la eficiencia del modelo CNN creado desde cero en relación con tres modelos pre-entrenados de la red convolucional VGG-19[2] utilizando la clasificación Softmax.

4. Resumir la efectividad de cada modelo individual de CNN basado en los resultados obtenidos al reconocer materiales de vídeos completamente nuevos a través de un resumen de clasificación de fotogramas.

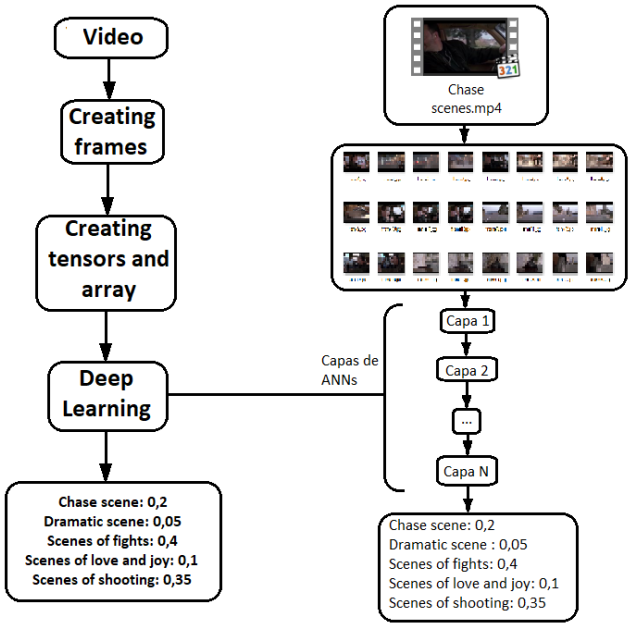


Figura 1. Abstracción del Modelo objetivo.

1.3 Metodología

El tipo de aprendizaje que utilizamos en este trabajo se conoce como aprendizaje supervisado y se trada de deducir una función a partir de datos de entrenamiento compuestos por pares *(X,Y)*.

Para diseñar este sistema es necesario asumir que existe una relación entre tensor *X*, que representa una imagen, y una secuencia *Y* de etiquetas, que representa las correspondientes clases de imágenes. Podemos pensar en una función *f(x)* que transforma el tensor *X* en la secuencia *Y* dentro de un conjunto *C*, formado por pares *(X,Y)*, que representa todas las posibles imágenes con palabras escritas:

$$Y = f(X), (X, Y) \in C$$

Ya como conocemos las categorias de cada imagen, podemos diseñar un sistema que "aprenda a conocer vídeo" mostrándole imágenes(fotogramas) y ajustando la salida hasta que llegamos a la salida ideal y conocida.

En este sentido, el sistema de reconocimiento de vídeo también se comporta como una funcion *f´(x)* que toma como entrada tensores *X* y unos parámetros δ, propios del

sistema, para dar como resultado una secuencia *Y´*, que tomaremos como salida del sistema:

$$Y´ = f´(X, \delta)$$

Nuestro objetivo es aproximar *Y´*(salida propia del sistema), a *Y*(salida real) de todo *C*. Para esto, escribiremos nuestro problema como una tarea de optimización y nuestro primer objetivo será:

$$Minimizar\ E = f(err)(Y', Y),\ \{X,Y\} \in C$$

Siendo *f(err)* una función de error que mide la diferencia entre dos secuencias, y que toma el valor 0 cuando ambas son iguales. Así el problema general de aprendizaje trata de minimizar este error y realizar distintas evaluaciones sobre los diversos modelos encontrados, partiendo de esta función objetivo y un subconjunto $S \subset \delta$.

La funcion *f´(x)* se obtiene mediante arquitecturas DL y nuestro objetivo será la optimización de este parámetro.

Para ello usaremos arquitecturas formadas por distintas capas de neuronas formando una red neuronal artificial (ANNs), donde podemos demostrar el uso de capas CNNs y modelos preentrenados, para el procesado de las imagenes. Los modelos han sido implementados en Python 3 usando TensorFlow y Keras como librerías para el diseño de arquitecturas y ha sido entrenadas con vídeo de películas(es decir, no de tipo doméstico o deportibo).

La metodología planteada en este proyecto se basa en distintas publicaciones donde se afronta este problema desde el DL [3, 4, 5].

1.4 Estado del Arte

Los grandes volúmenes de películas producidas hoy en día representan un gran problema para su procesamiento manual. Los metadatos humanos son generalmente insuficientes para describir el contenido principal de una película y/o son inexactos a causa de las dificultades para reproducir con precisión la información. Los experimentos realizados por U. Khan et al. lo confirman [5]. En el curso de estos experimentos, se pidió a algunos voluntarios que vieran varios vídeos de diferentes categorías y que señalaran la información clave contenida en ellos. Al comparar la información que ofrecían con la situación real recogida a través de un análisis exhaustivo de los vídeos, autores encontraron que las etiquetas semánticas creadas por el participantes no se correspondían entre sí y constituían una violación. Sus experimentos también demostraron que esta tarea necesita un análisis inteligente de vídeo para extraer automáticamente información importante de las películas.

Como este artículo [5] es similar a nuestro problema, fue tomado como base para nuestro trabajo. Las ideas propuestas por U. Khan et al. se superponen a lo que vamos a hacer en nuestro trabajo, pero también tienen muchas diferencias: vamos a utilizar tres modelos diferentes basadas en VGG-19, mientras que en este trabajo [5] sólo habla del modelo Inception-V3; este trabajo [5] incluye un gran número de categorías, mientras que en nuestro trabajo nos limitamos a sólo 5 que difieren de las categorías de películas

que se utilizan para clasificar en nuestro trabajo[5]; en nuestro trabajo los modelos de redesconvolucionales están implementados usando la biblioteca Keras.

El análisis semántico de las películas requiere un algoritmo de aprendizaje automático (ML) que le permita explorar las características de las escenas de las películas. Los cambios que han ocurrido recientemente en el ML han llevado a un cambio de paradigma en el análisis de datos complejos con una buena eficiencia que incluso excede la precisión humana. La computación paralela de propósito general que ofrecen las GPUs ha permitido que la enorme potencia de cálculo necesaria para el aprendizaje de la máquina se combine en un solo ordenador, allanando el camino para un análisis eficiente de imágenes y vídeo. Sin embargo, el uso del aprendizaje automático para estudiar las características tradicionales creadas manualmente es ineficaz para nuestro problema debido a los problemas asociados con: cambio de escala y rotación, formulación de los modelos matemáticos necesarios, falta de generalización, bajo rendimiento en diferentes entornos, etc.

Es por eso que resolveremos el problema del análisis de semántico de vídeo con la ayuda de Deep Learning [6], que no requiere del estudio de ingeniería de características. En cambio, este enfoque aprende a representar datos detectando estructuras complejas en conjuntos de datos con múltiples niveles de abstracción. Esto significa que nuestro problema de clasificación se basa en un conjunto de datos digitales.

Aunque las CNN son similares a las redes neuronales artificiales tradicionales, tienen una arquitectura mucho más profunda y son más adecuadas para estudiar patrones básicos en datos complejos. Una arquitectura típica de CNN (Figura 2) incluye capas convolution(eng), pooling(eng), activación, combinación y clasificación. Entrenar este modelo desde cero para realizar una nueva tarea requiere enormes recursos computacionales, mucho tiempo de aprendizaje y una gran cantidad de datos de entrenamiento. Investigaciones recientes [7] demuestran que las características aprendidas por una CNN pueden ser transferidas de un problema de aprendizaje a otro. Empezando por las capas inferiores de la CNN, que extraen características generales y de bajo nivel, la especificidad de las características aumenta con la transición a una capa superior de la CNN, haciendo que la capa final sea la más específica para resolver tareas específicas. Por lo tanto, los niveles inferiores de una CNN, entrenados para un gran conjunto de datos, contienen ponderaciones aprendidas para características de bajo nivel que pueden utilizarse para enseñar el modelo en una nueva tarea. En un modelo de CNN pre-entrenado como un elemento fijo, su capa final se puede modificar y volver a entrenar para realizar una nueva tarea. Este enfoque se llama "transferencia de aprendizaje" y lo utilizamos para modificar y preparar los niveles finales la red VGG-19, previamente entrenados en un gran conjunto de datos, para extraer etiquetas de película. Aunque

VGG-19 fue preparada para una tarea completamente diferente, sus características se transfieren efectivamente al estudio de nuestra nueva tarea.

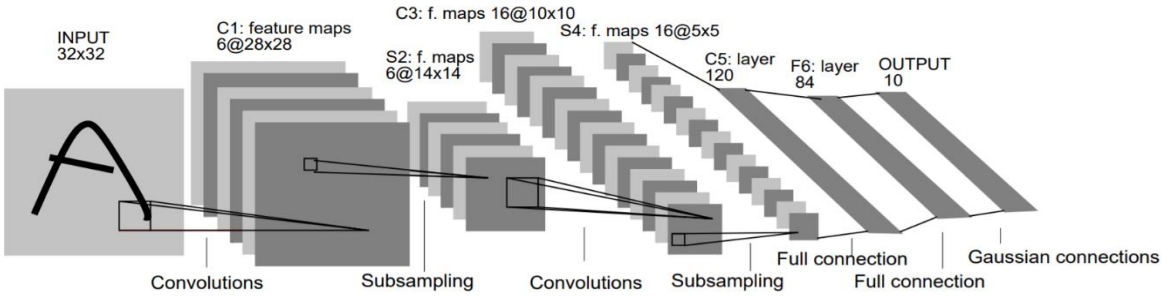


Figura 2. Ilustración de la CNN propuesta por LeCun et al. [6] para el reconocimiento de dígitos.

Esta información puede ser utilizada en una serie de tareas, incluyendo optimización de búsqueda, búsqueda de scripts, detección de objetos, traducción de películas al lenguaje natural, detección de eventos, reconocimiento de acciones, reconocimiento de comportamientos, sistemas de recomendación., etc.

2 Datos

Como ya se ha comentado en el capítulo anterior, es necesario un subconjunto de muestras $S \subset \delta$ para realizar el entrenamiento y la evaluación del sistema. En el problema que nos ocupa este subconjunto estará formado por un conjunto de imágenes, con sus correspondientes clasificaciones, en un format determinado.

2.1 Fuente

Los vídeos fueron descargados desde YouTube según los criterios adecuados para las siguientes clases: “Chase scenes”, “Dramatic scenes”, “Scenes of fights”, “Scenes of love and joy” y “Scenes of shooting”. Para crear un el dataset, primero se guardaron en la carpeta «/videodata/» todos estos vídeos descargados, con una longitud del tiempo total de todos los vídeos de 78,8 minutos. Para comprobar la exactitud de las respuestas de los modelos, los datos del conjunto de datos de la carpeta de vídeo se dividieron de la siguiente manera: 75% para entrenamiento, 15% para validación, 15% para prueba. Además, se realizó unos experimentos con nuevos datos de YouTube, que se encuentran en la carpeta «/new_data/video/». Se eligieron estos vídeos porque solo contienen escenas correspondientes a la clases que queremos representar.

Todos los vídeos tienen la misma resolución de 720p y la misma extensión de archivo .mp4.

2.2 Preprocesado de los Datos

Para cortar el vídeo en fotogramas, se ha desarrollado una función *make_frames()* que se escribió utilizando las bibliotecas “os” (para trabajar con el sistema de archivos del ordenador) y “cv2” para cortar el vídeo, respectivamente. La base de datos final para el entrenamiento consiste en 4887 imágenes en el formato .jpg.

Ya que las imágenes cortadas tienen una alta resolución y no están relacionadas con sus clases, se trata de reducir la resolución de la imagen y unirlas a las clases correspondientes. Por eso usamos la biblioteca «ImageDataGenerator». Esta librería nos permitió: reducir las imágenes a una resolución única de 224x224, añadir cada etiqueta de imagen de la clase correspondiente y hacer «data augmentation» a las imágenes, consiguiendo transformadas aleatoriamente mediante inversión y añadiendo zoom. La Figura 3 muestra imágenes después del usar ImageDataGenerator.

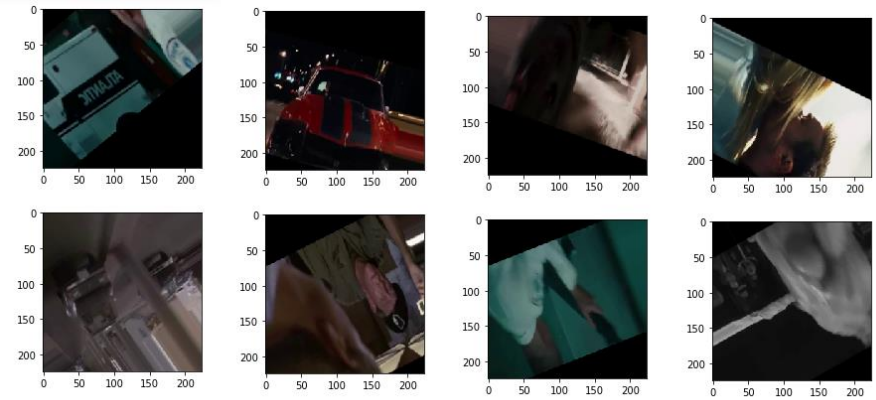
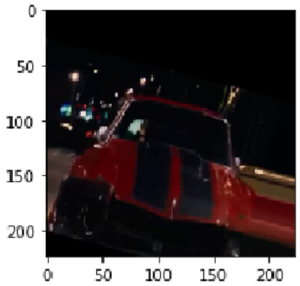


Figura 3. Imagenes despues del ImageDataGenerator

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0] [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2] [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2] [2, 2, 2, 2, 2,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3] [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

Figura 4. Ejemplo de listas con clases correspondientes a las imágenes



```
[[[0.         0.         0.        ]  [[0.         0.         0.        ]  [[0.         0.         0.        ]
  [0.         0.         0.        ]   [0.         0.         0.        ]   [0.         0.         0.        ]
  [0.         0.         0.        ]   [0.         0.         0.        ]   [0.         0.         0.        ]
  ...                                  ...                                  ...
  [0.7019608  0.7607844  0.6862745 ]   [0.70411754 0.76294106 0.68843126]   [0.708903   0.76772654 0.69321674]
  [0.7019608  0.7607844  0.6862745 ]   [0.7064544  0.7652779  0.6907681 ]   [0.7077346  0.7665581  0.6920483 ]
  [0.7019608  0.7607844  0.6862745 ]]  [0.7087913  0.76761484 0.69310504]]  [0.7065661  0.7653896  0.6908798 ]]

 [[0.5529412  0.58431375 0.5019608 ]  [[0.5529412  0.58431375 0.5019608 ]  [[0.5529412  0.58431375 0.5019608 ]
  [0.5529412  0.58431375 0.5019608 ]   [0.5529412  0.58431375 0.5019608 ]   [0.5521202  0.58349276 0.5011398 ]
  [0.5764514  0.60782397 0.52547103]   [0.56824523 0.5996178  0.51726484]   [0.55728877 0.5886613  0.5063084 ]
  ...                                  ...                                  ...
  [0.         0.         0.        ]   [0.         0.         0.        ]   [0.         0.         0.        ]
  [0.         0.         0.        ]   [0.         0.         0.        ]   [0.         0.         0.        ]
  [0.         0.         0.        ]]  [0.         0.         0.        ]]  [0.         0.         0.        ]]]
```

Figura 5. Imagen y su tensor asociado.

Establecer para todas las imágenes la misma escala permite diseñar la arquitectura del sistema conociendo en todo momento las dimensiones del flujo de datos a través de las distintas capas. Se podría diseñar el sistema para imágenes de tamaño variable, pero complicaría el código sin aportar valor añadido al proyecto. Por otro lado, al invertir los tonos en la escala de color se consigue que los píxeles correspondientes a partes escritas de la imagen sean representados con valores elevados dentro del tensor.

Como resultado final tenemos un dataset $S \subset \delta$ compuesto por pares *(X,Y)*, donde *X* es un tensor normalizado que representa una imgen de una película correspondiente a *Y* que es una secuencia de etiquetas, expresadas como números enteros.

2.3 Imágenes como Tensores

Los tensores son una herramienta útil para presentar los datos en forma de matríces (en su mayoría numéricas). Por tanto vamos a exponer una pequeña definición y el sentido de su uso.

Un tensor es un objeto de álgebra lineal que convierte linealmente elementos de un espacio lineal en elementos de otro como tensores de distinto orden. El orden de un tensor será el número de índices necesario para especificar cada una de sus componentes. La siguiente tabla se muestra un ejemplo de correspondencia:

Tabla 1. Casos especiales de Tensores.

Entidad Matemática	Ejemplo
Tensor de orden 0, Escalar.	255
Tensor de orden 1, Vector.	[28 41.6 12.3 9.24]
Tensor de orden 2, Matríz	$\begin{bmatrix} 51 & 46 \\ 34 & 18 \end{bmatrix}$

Tensor de orden 3, Matríz de matrices	$\begin{bmatrix}\begin{bmatrix}32 & 44\\19 & 56\end{bmatrix} & \begin{bmatrix}62 & 71\\101 & 23\end{bmatrix}\\\begin{bmatrix}48 & 96\\85 & 19\end{bmatrix} & \begin{bmatrix}37 & 12\\51 & 28\end{bmatrix}\end{bmatrix}$
...	...
Tensor de orden N	...

A menudo, el tensor se presenta como una tabla multidimensional $d_0 * d_1 * ... * d_n$, lleno de números - componentes tensores (donde d_n es la dimensión del espacio vectorial por encima del cual se especifica el tensor, y el número de multiplicadores coincide con el llamado orden de tensor).

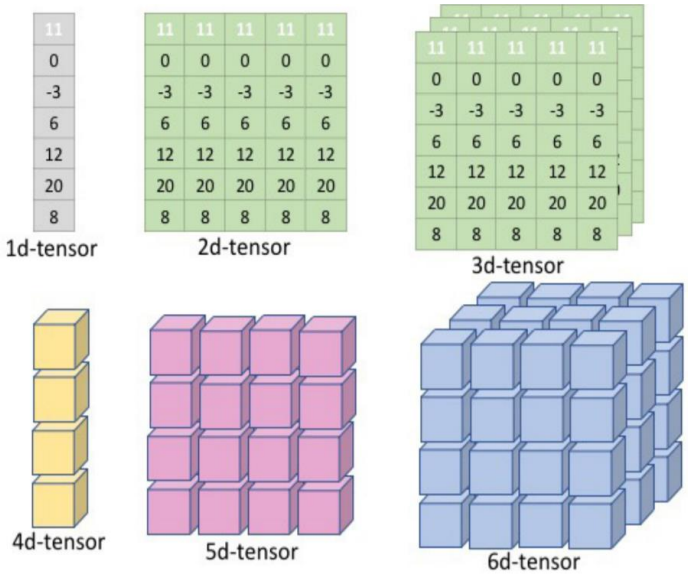


Figura 6. Ejemplos de tensores de distinto orden.

Este enfoque nos permite tratar las imágenes y vídeo como tensores de un determinado orden. Concretamente para problemas de visión por computador se suele tomar las imágenes como tensores de orden 3, donde las dimensiones representan la altura, la anchura y los tres canales RGB que forman el color.

2.4 El dataset creado

Después de preprocesar los datos, tenemos un conjunto de datos con las siguientes características: 3403 imágenes de entrenamiento, 741 imágenes de validación, 738 de prueba; todas las imágenes tienen el mismo formato 224x224; los tensores de imagen se normalizaron; cada imagen pertenece a su clase correspondiente. La duración de cada vídeo, de los cuales hemos sacado imágenes por separado, es la siguiente: vídeo "Chase" - 22:25, vídeo "Drama" - 13:45, vídeo "Fights" - 15:59, vídeo "Love and Joy" - 12:00, vídeo "Shooting" - 15:43.

También tenemos un conjunto de datos con vídeos descargados de YouTube con las mismas características pero solo con diferente cantidad de fotos, con el fin de hacer

experimentos una vez los modelos han sido entrenados. Están guardadas en la carpeta «/new_data/frames/»: 520 imágenes de “Chase Scenes”, 148 imagenes de “Dramatic Scenes”, 310 imagenes de “Scenes of Fights”, 146 imagenes de “Scenes of Love and Joy”, 468 imagenes de “Scenes of Shooting”. La Figura 7 muestra algunos ejemplos de fotogramas para poder tener una noción de qué tipo de imagenes pertenecen a cada clase.

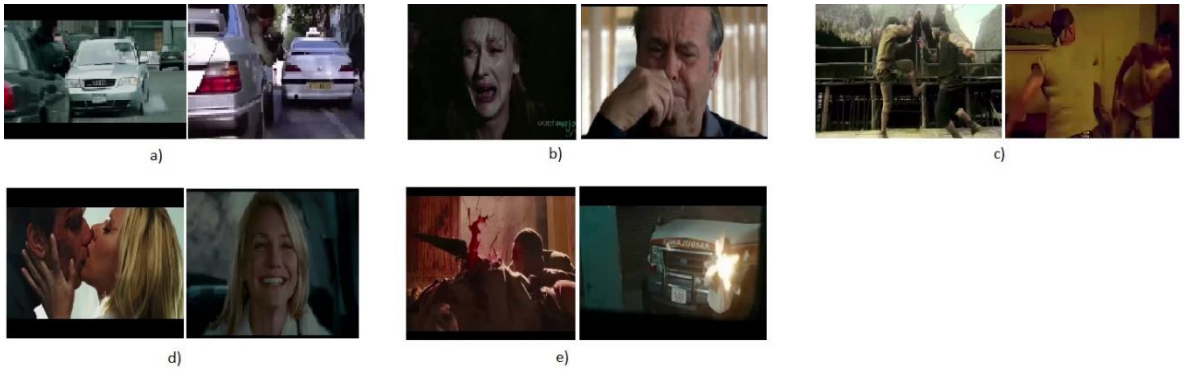


Figura 7. Fotogramas de ejemplo de cada clase: a) Chase Scenes, b) Dramatic Scenes, c) Scenes of Fighting, d) Scenes of Love and Joy, e) Scenes of Shooting

Las categorías por las que clasificamos los vídeos están entre las más populares, por eso los utilizamos en nuestro trabajo. Se puede ver una descripción de cada una en la Tabla 2:

Tabla 2. Categorías defenidas

Nombre de la categoría	Información de la categoría
Chase Scenes	Esta categoría incluye vídeos de persecución de vehículos. Donde un grupo de personas persigue a otro.
Dramatic Scenes	Esta categoría contiene vídeos que recogen momentos de tristeza, pena y sentimientos de los héroes.
Scenes of Fighting	En esta categoría, hemos añadido vídeos que contienen escenas de peleas.
Scenes of Love and Joy	Esta categoría contiene vídeos en los que el héroe se alegra, sonríe, así como momentos en los que los héroes se besan entre sí.
Scenes of Shooting	Aquí hay vídeos de tiroteos entre un grupo o entre varios héroes, escenas que muestran armas y disparos.

3 Redes Neuronales Artificiales

El presente capitulo comienza con un resumen sobre ANNs para explicar el conjunto de capas que forman la arquitectura de un sistema para Deep Learning. Se ha puesto especial énfasis en las redes CNN y redes pre-entrenadas. El objetivo es presentar los fundamentos teóricos que permitan entender la arquitectura diseñada y facilite la interpretación de los resultados obtenidos.

3.1 Redes Neuronales Artificiales multicapa (ANN)

La red neural artificial (ANN) es un - modelo matemático, así como su implementación software o hardware, construido sobre el principio de organización y funcionamiento de redes neurales biológicas - redes de células nerviosas de un organismo vivo. Estas redes están formadas por un conjunto de elementos simples, llamados nodos o neuronas, interconectados para procesar y transmitir señales de una forma dirigida. Generalmente, las arquitecturas DL de ANN consisten en varias capas de neuronas que trabajan en paralelo y transmiten la señal desde la capa de entrada a la de salida a través de varias capas ocultas. Este ripo de redes se denominan de Feed Forward. La Figura 8 muestra un ejemplo.

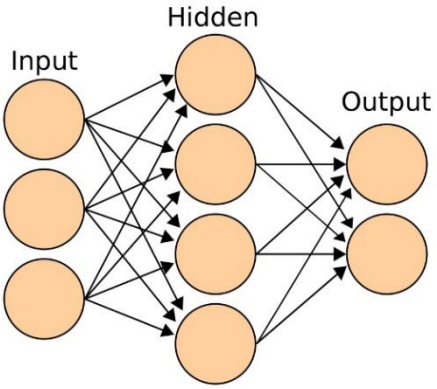


Figura 8. Ejemplo de una ANN con 3 capas.

Cada neurona consta de unas entradas, con pesos asociados a cada entrada, un umbral propio de cada neurona, y una función de activación que genera una salida a partir de un agregado de las entradas recibidas. La Figura 9 muestra la estructura de una neurona artificial.

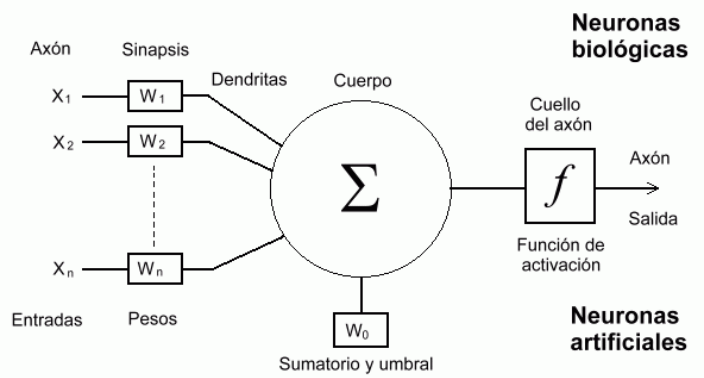


Figura 9. Neurona artificial inspirada en la neurona biológica.

El valor de salida consiste en tomar los valores de entrada(que recibe de otras neuronas o de la entrada del sistema), multiplicarlos por sus respectivos pesos y sumar estos valores con el umbral para aplicarles una función de activación. Podemos escribir la siguiente función:

$$y_i^{'} = f(X, W_i, b_i) = f_a\left(\sum_{p=0}^{N^X-1} w_{p.i.} x_p b_i\right)$$

Donde X es el conjunto de entrada $[x_0, x_1, \dots, x_{N^X-1}]$, W_i es el conjunto de pesos asociados a cada entrada x_p de la neurona i, b_i representa un umbral, que actúa directamente sobre la neurona y es independiente de las entradas, f_a es una función de activación, que suele considerarse derivable y no lineal para adaptarse a un número mayor de problema, con la que se genera salida $y_i^{'}$.

Una solo neurona aislada no es capaz de resolver problemas de una dificultad elevada, pero la interconexión de ellas siguiendo determinadas topologías permite modelar funciones que son capazes de trabajar con problemas de una dificultad elevada.

El teorema de Aproximación Universal [8] establece que una sola capa intermedia es suficiente para aproximar, con una precisión arbitraria, cualquier función con un número finito de discontinuidades, siempre y cuando las funciones de activación de las neuronas ocultas sean no lineales. El teorema establece que las redes multicapa no añaden capacidad a menos que la función de activación de las capas sea no lineal.

El resultado de este tipo, donde la salida de la capa l es la entrada de la capa l + 1(como muestra la Figura 9), se puede expresar como:

$$Y^{'} = f(H^L, W^Y, B^Y)$$

$$H^L = f(H^{L-1}, W^L, B^L)$$

$$H^{L-1} = f(H^{L-2}, W^{L-1}, B^{L-1})$$

...

$$H^0 = f(X, W^0, B^0)$$

Donde $Y^{'}$ es la salida del sistema, W^Y los pesos, B^Y bias asociados a la capa de salida, mientras que H^L la salida, W^L los pesos y B^L los bias de la capa l, siendo L número total de capas ocultas.

3.2 Redes Neuronales Convolucionales (CNN)

Una CNN es una arquitectura especial de redes neuronales artificiales, propuesta por Yan LeCunn en 1988 y dirigida al reconocimiento efectivo de imágenes, es parte de la tecnología del aprendizaje profundo. Utiliza algunas de las características de la corteza visual, en la que se descubrieron las llamadas células simples, reaccionando a líneas rectas en diferentes ángulos, y células complejas, cuya reacción está asociada a la activación de un determinado conjunto de células simples. Así, la idea de las redes neuronales de convolución consiste en la alternancia de capas de convolución y capas

subdiscretizantes. A continuación, y con el fin de exponer su funcionamiento vamos a definir más detalladamente esta operación.

3.2.1 La arquitectura de la red neural convolucional

En un perceptrón normal, que representa una red neuronal completamente conectada, cada neurona está conectada a todas las neuronas de la capa anterior, y cada conexión tiene su propio coeficiente de peso personal. En la red neuronal convolucional en la operación de convolución sólo se utiliza una matriz limitada de escalas de pequeño tamaño, que se "mueve" a lo largo de toda la capa procesada (al principio - directamente sobre la imagen de entrada), formando después de cada cambio una señal de activación para la neurona de la siguiente capa con una posición similar. Es decir, para las diferentes neuronas de la capa de salida se utilizan los pesos comunes - la matriz de pesos, que también se llama un conjunto de pesos o el núcleo (kernel) de la convolución. Está construido de tal manera que codifica gráficamente cualquier característica, por ejemplo, la presencia de una línea inclinada en un cierto ángulo. La siguiente capa, resultante de la operación de convolución de dicha matriz de pesos, muestra la presencia de una determinada línea oblicua en la capa procesada y sus coordenadas, formando el llamado mapa de características. Naturalmente, en la red neural de convolución, un conjunto de escalas no es uno, sino toda una gama que codifica todo tipo de líneas y arcos en diferentes ángulos. Al mismo tiempo, tales núcleos de convolución no son definidos por el investigador, sino que se forman de forma independiente mediante el aprendizaje de la red por el método clásico de propagación de errores. El paso de cada conjunto de escalas forma su propia copia del mapa de características, haciendo que la red neuronal sea multidimensional (muchos mapas de características independientes en una capa).

También hay que tener en cuenta que durante la búsqueda de una capa de una matriz de escalas se suele mover no sobre un paso completo (el tamaño de esta matriz), sino sobre una pequeña distancia (stride). Por ejemplo, cuando la matriz es 5×5, es desplazada por una o dos neuronas (píxeles) en lugar de cinco, para no "pasar por encima" de la característica deseada. La operación de submuestreo (subsampling) reduce la dimensionalidad de los mapas de características generados.

En esta arquitectura de red se considera que la información sobre la presencia de la característica deseada es más importante que el conocimiento exacto de sus coordenadas, por lo que de varias neuronas vecinas del mapa de características se selecciona la máxima y se toma como una neurona del mapa de características de reducida dimensionalidad. La operación de encontrar el promedio entre neuronas adyacentes también se utiliza a veces. Debido a esta operación, además de acelerar otros cálculos, la red se vuelve más invariable a la escala de la imagen de entrada. Así, repitiendo varias capas de convolución y submuestreo una tras otra, se construye una red neuronal convolucional. A la salida de la red, varias capas de la red neural totalmente conectada (perceptrón multicapa, MLP) se instalan a menudo adicionalmente. Si en la primera capa el núcleo de la convolución pasa sólo una imagen inicial, entonces en las capas internas el mismo núcleo pasa en paralelo en todos los mapas de atributos de esta capa, y el resultado de la convolución se

resume, formando (después de pasar la función de activación) un mapa de atributos de la siguiente capa correspondiente a este núcleo de convolución.

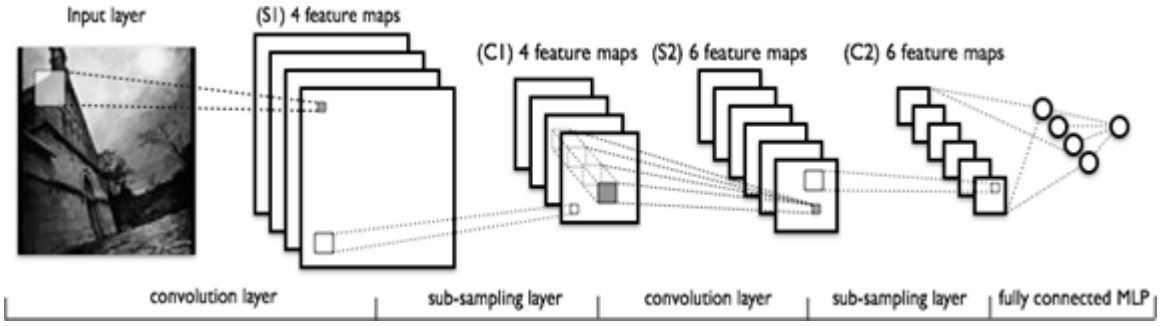


Figura 10. Topología de una red convolucional

La operación de convolución puede describirse mediante la siguiente fórmula:

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l] * g[k,l]$$

Aqui *f* - matriz de imagen original, *g* - núcleo de convolución.

Informalmente, esta operación se puede describir de la siguiente manera: con la ventana del tamaño del núcleo *g* se recorre con un paso dado (normalmente 1) la imagen completa *f*, y en cada paso multiplicamos por elementos el contenido de la ventana por el núcleo *g*, el resultado se resume y se escribe en la matriz de resultados.

3.2.2 Capa de convolución

La capa de convolución implementa la idea de campos receptivos locales, es decir, cada neurona de salida está conectada sólo a una cierta (pequeña) área de la matriz de entrada y así simula algunas características de la visión humana. Esto se puede anotar en la siguiente fórmula:

$$x^l = f(x^{l-1} * k^l + b^l)$$

Donde x^l - salida de capa *l*, *f()* - función de activación, *b* - coeficiente de paso, con el simbolo * se describe operacion de convolución de la salida *x* con el núcleo *k.*

Al mismo tiempo, debido a los efectos de borde, el tamaño de las matrices iniciales disminuye.

$$x_j^l = f\left(\sum_i x_j^{l-1} * k_j^l + b_j^l\right)$$

Aqui x_j^l - mapa de características *j*(salida de capa *l*), *f()* - función de activación, b_j^l - es el coeficiente de umbral para el mapa de características *j*, k_j^l - número *j* de núcleo de la circunvolución, x_j^{l-1} - mapas de características de la capa anterion.

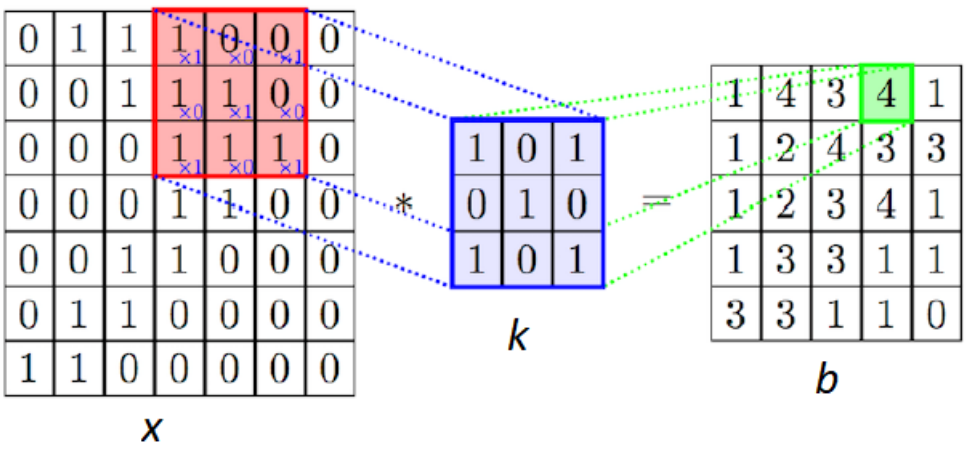


Figura 11. Ejemplo de convolución entre tensores de orden 2, con b = [1,1]

3.2.3 Capa subsampling

Las capas de este tipo reducen el tamaño del mapa de características de entrada (normalmente en un factor de 2). Esto se puede hacer de diferentes maneras, en este caso consideraremos el método de selección del elemento máximo (max-pooling) - todo el mapa de características se divide en celdas 2x2 del elemento, de las cuales se selecciona el valor máximo. Formalmente, la capa se puede describir de la siguiente manera:

$$x^l = f(a^l * subsample(x^{l-1}) + b^l)$$

Donde x^l - salida de capa *l*, *f()* - función de activación, *a,b* - coeficientes, subsample() - operación de muestreo de valores máximos locales.

El uso de esta capa le permite mejorar el reconocimiento de las muestras redimensionadas (reducidas o aumentadas).

3.2.4 Capa MLP

Las neuronas de cada mapa de la capa anterior de la submuestra están relacionadas con una sola neurona de la capa oculta. Así, el número de neuronas de esta capa oculta *l* es igual al número de mapas de la capa oculta anterior *l-1*, pero las conexiones pueden no ser necesariamente tales, por ejemplo, sólo una parte de las neuronas de cualquiera de los mapas de capa oculta puede estar conectada con la primera neurona de la capa oculta *l*, y el resto con la segunda *l-1*, o todas las neuronas del primer mapa están conectadas con neuronas de *l* y *l-1* capas ocultas. El cálculo de los valores de neuronas se puede describir mediante la fórmula:

$$x_j^l = f\left(\sum_i x_i^{l-1} * w_{ij}^{l-1} + b_j^{l-1}\right)$$

Donde x_j^l - salida de capa *l*, *f()* - función de activación, *b* - coeficiente de paso, *w* - matriz de coeficientes de pesos.

El objetivo de esta capa es mejorar la calidad del reconocimiento mediante la optimización de una función no lineal compleja.

3.2.5 Función de activación

Una de las etapas del desarrollo de la red neuronal es la elección de la función de activación neuronal. El tipo de función de activación determina en gran medida la funcionalidad de la red neuronal y el método de enseñanza de la red. El algoritmo clásico de propagación inversa de errores funciona bien en redes neuronales de dos y tres capas, pero con el aumento de la profundidad comienza a experimentar problemas. Una de las razones es el llamado atenuación del gradiente. A medida que el error se propaga de la capa de salida a la capa de entrada, cada capa se mezcla con una función de activación derivada para producir el resultado actual.

3.2.5.1 ReLU

Una unidad lineal rectificada (una unidad que emplea el rectificador también se denomina unidad lineal rectificada ReLU) tiene salida 0 si la entrada es inferior a 0, y salida bruta en caso contrario. Es decir, si la entrada es mayor que 0, la salida es igual a la entrada. La operación de ReLU es más cercana a la forma en que funcionan nuestras neuronas biológicas.

$$f(x) = \max(x, 0)$$

ReLU es no lineal y tiene la ventaja de no tener errores de retropropagación(backpropagation) a diferencia de la función sigmoide, también para grandes redes neuronales, la velocidad de construcción de modelos basados en ReLU es muy rápida y esta es una de las razones para utilizar esta función de activación en capas convolucionales.

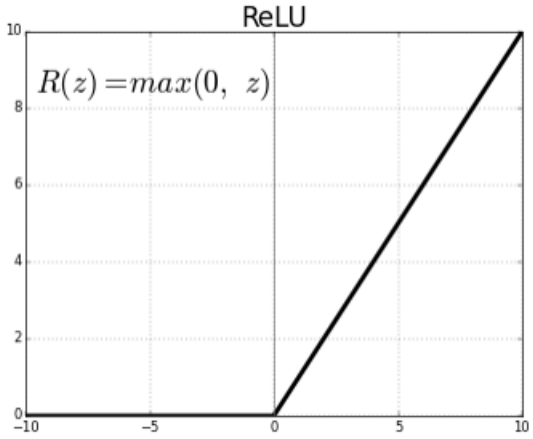


Figure 12. Función de activación ReLU.

3.2.5.2 Softmax

Softmax es una función de activación muy interesante porque no sólo mapea nuestra salida a un rango de[0,1] sino que también mapea cada salida de tal manera que la suma total es 1. La salida de Softmax es por lo tanto una distribución de probabilidad. La función softmax se utiliza a menudo en la capa final de un clasificador basado en redes neuronales como CNN para clasificación multiclase. Como nuestras redes están entrenadas bajo un régimen de "categorical cross-entropy", dando una variante no lineal

de regresión logística multinomial, utilizamos la multiclasificación de Softmax en esta situación.

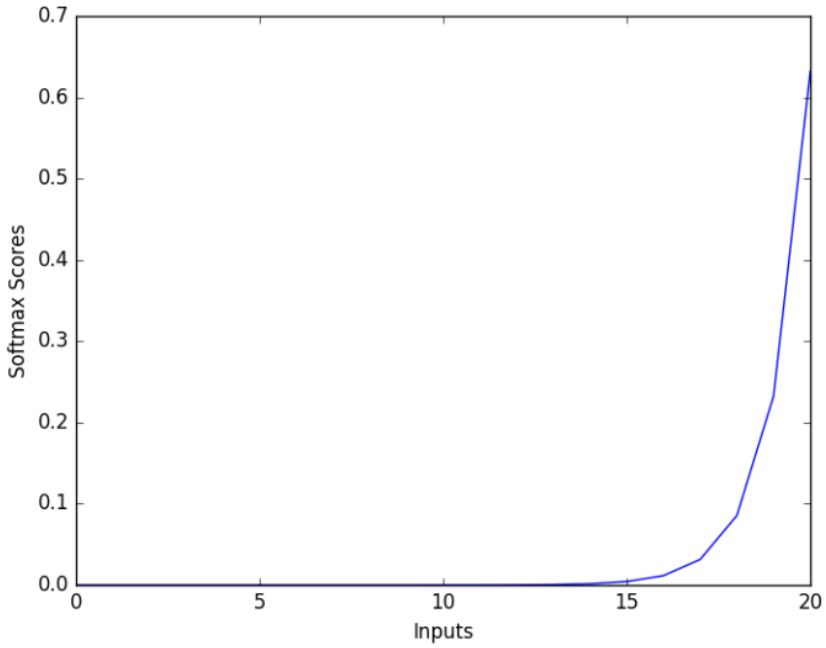


Figura 13. Función de activación Softmax.

Si tomamos la salida de la última capa H^L como entrada de la capa Softmax, obtenemos la salida Y' mediante:

$$y_i^l = Softmax(h_i^L) = \frac{e^{h_i^L}}{\sum_{k=0}^{N^{L-1}} e^{h_k^L}}$$

Donde N^L expresa el numero de neuronas de la última capa, que corresponde con el número de salidas de la capa Softmax.

3.2.6 Medida de Error

Hasta ahora hemos descrito cómo una ANN con una capa de Softmax se puede utilizar para la clasificación en secuencias de etiquetas, pero es necesaria una función objetivo que permita optimizar los parámetros de la red.

Generalmente para entrenar redes se utilizan funciones derivables que miden el error entre la salida dada por el sistema y salida ideal, como hemos ido observado a lo largo del trabajo. En nuestro caso esta medida obtiene mediante uso de una funcion “categorical cross-entropy”.

3.2.6.1 Categorical Cross-Entropy Loss

También llamado Softmax Loss, se asicoa una pérdida de entropía cruzada a la activación Softmax. Si usamos esta pérdida, entrenaremos a una CNN para que emita una probabilidad sobre las clases *C* para cada imagen. Se utiliza para la clasificación multiclase.

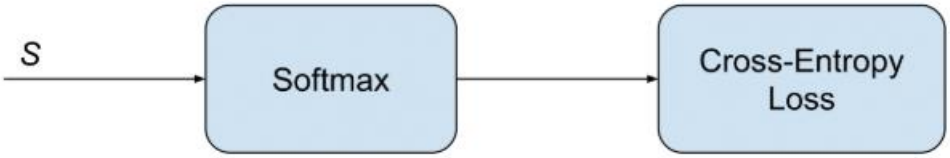


Figura 14. Representación esquemática de la conexión Softmax y funcion de pérdida.

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i \log\left(f(s)_i\right)$$

En nuestro caso, clasificación multi-clase, las etiquetas están hechas usando *One-Hot encoding*, por lo que sólo la clase positiva C_p mantiene su término en pérdida. Solo hay un elemento del vector de destino *t* que no es cero $t_i = t_p$. Por lo tanto, descartando los elementos de la suma que son cero debido a las etiquetas de destino, podemos escribir:

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Donde s_p es la puntuación de la CNN para la clase positiva.

3.3 Transfer Learning/ Fine-tuning

La transferencia de aprendizaje (o "transfer learning", TL) es un problema de investigación en el aprendizaje automático que se centra en almacenar el conocimiento adquirido mientras se resuelve un problema y aplicarlo a otro problema diferente pero relacionado [9].

La definición de transferencia de aprendizaje se da en términos de dominio y tarea. El dominio *D* consiste en: un espacio de característica *X* y una distribución de probabilidad marginal *P(X)*, donde

$$X = \{x_1 \dots x_n\} \in X$$

Dado un dominio específico, *{D={X,P(X)}}*, una tarea consta de dos componentes: un espacio de etiqueta *Y* y una función predictiva objetiva *f(*)* (denotada por *T={Y,f(*)}*), que se aprende a partir de los datos de entrenamiento que consisten en pares *{x_{i},y_{i}}*, donde *x_{i}* pertenece a *X* y *y_{i}* pertenece a *Y*. La función *f(*)* puede usarse para predecir la etiqueta correspondiente, *f(x)*, de una nueva instancia *x*, que puede ser reescrito por la forma probabilística de la distribución condicional de probabilidad *P(Y|X)*. Una tarea puede definirse como *T={Y,P(Y|X)}*. Con las notaciones de dominio y tarea, el TL se define como se indica a continuación:

Dado un dominio de origen D_S y tarea de aprendizaje T_S, y un dominio de destino D_T y tarea de aprendizaje T_T, TL pretende ayudar a mejorar el aprendizaje de la función predictiva de destino f(*) en D_T utilizando los conocimientos en D_S y T_S, donde $D_S \neq D_T$, o $T_S \neq T_T$.

En la definición anterior, la condición $D_S \neq D_T$ se refiere a $X_S \neq X_T$ o $P_S(X) \neq P_T(X)$, es decir, los dominios fuente y destino tienen diferentes espacios de características o distribuciones de probabilidad marginales, mientras que la condición $T_S \neq T_T$ significa $Y_S \neq Y_T$, o $P(Y_S|X_S) \neq P(Y_T|X_T)$, es decir, los dominios fuente y destino tienen diferentes espacios de etiquetas o distribuciones de probabilidad condicional. Tenga en cuenta que cuando los dominios de destino y de origen son los mismos, es decir, $D_S = D_T$, y sus tareas de aprendizaje son las mismas, es decir, $T_S = T_T$, el problema de aprendizaje se convierte en un problema tradicional de aprendizaje automático. Si el TL mejora el rendimiento al utilizar únicamente D_T y T_T, el resultado se refiere a una transferencia positiva. De lo contrario, el deterioro del TL conduce a una transferencia negativa [10].

El " transfer learning " a través de la " fine-tuning " es otra herramienta poderosa que puede emplearse utilizando la representación de características. Mientras que el entrenamiento de una red con un conjunto de datos muy grande puede llevar un día, un modelo pre-entrenado puede ser ajustado con precisión para adaptarse a la aplicación en cuestión. Un modelo de pre-entrenado aprende un mapeo desde los píxeles hasta el espacio de características. Y como resultado, la mayoría de la red no tiene que ser reciclada. En su lugar, la última capa o capas pueden ser modificadas y reaprendidas con un conjunto de datos más pequeño con el fin de ajustar la red para que se ajuste a la aplicación deseada. El “fine tuning” de una red es un procedimiento basado en el concepto de “transfer learning” [11, 12]. Comenzamos a entrenar una CNN para aprender características para un dominio amplio con una función de clasificación dirigida a minimizar el error en ese dominio. Luego, reemplazamos la función de clasificación y optimizamos la red nuevamente para minimizar el error en otro dominio más específico. Bajo esta configuración, estamos transfiriendo las características y los parámetros de la red del dominio amplio al dominio específico.

La función de clasificación en la CNN original es un clasificador softmax que calcula la probabilidad de 1000 clases del conjunto de datos de ImageNet. Para iniciar el procedimiento de ajuste fino(“fine tuning”), eliminamos este clasificador softmax e inicializamos uno nuevo con valores aleatorios. El nuevo clasificador softmax es entrenado desde cero utilizando el algoritmo de retropropagación con nuestros datos, con categorías diferentes.

Para iniciar el algoritmo de retropropagación para el ajuste fino, es fundamental establecer adecuadamente los factores de aprendizaje de cada capa. La capa de clasificación, es decir, el nuevo clasificador softmax, necesita una gran velocidad de aprendizaje porque se ha inicializado con valores aleatorios. El resto de las capas necesitan una velocidad de aprendizaje relativamente pequeña porque queremos preservar los parámetros de la red anterior para transferir ese conocimiento a la nueva red. Sin embargo, tenga en cuenta que la velocidad de aprendizaje no es cero en el resto de las capas: se optimizarán de nuevo a un ritmo más lento.

3.4 Modelos para el uso del transfer learning

3.4.1 Inception-V3

Los modelos Inception son tipos de redes neuronales convolucionales diseñados por Google principalmente para la clasificación de imágenes. La principal diferencia entre

los modelos Inception y las CNN regulares son los bloques inception. Esto implica convolucionar el mismo tensor de entrada con múltiples filtros y concatenar sus resultados. Este bloque se representa en la imagen de abajo.

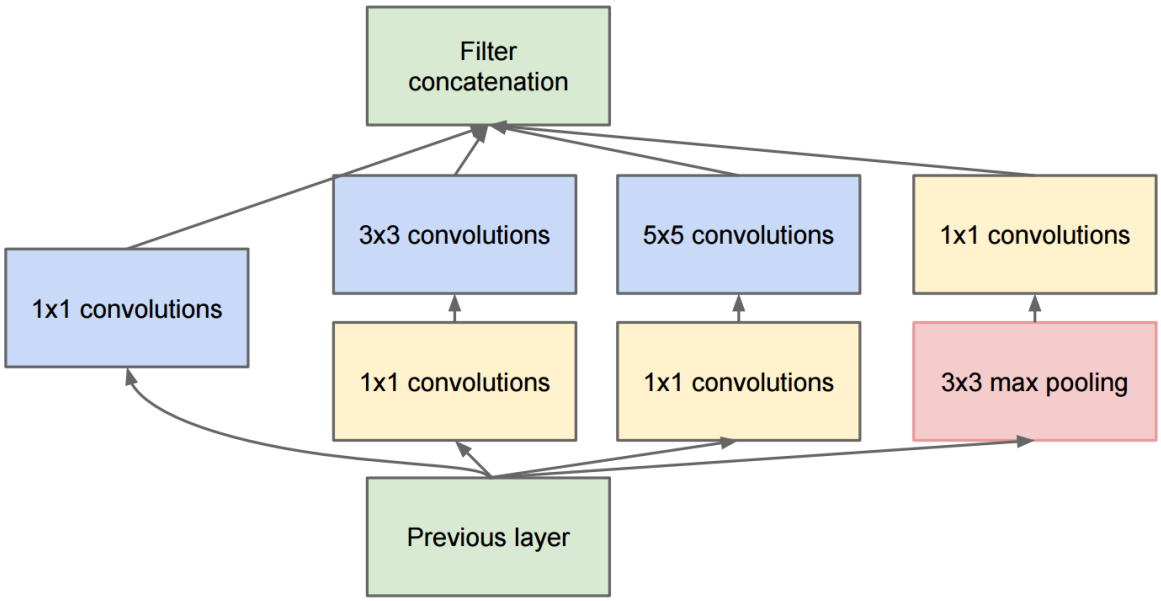


Figura 15. Bloques de inception.

Por el contrario, las CNN regulares realizan una sola operación de convolución en cada tensor.

Inception-V3[13] es una arquitectura de redes neuronales profundas que utiliza bloques inception como el que se describió anteriormente. Su arquitectura se ilustra en la siguiente figura.

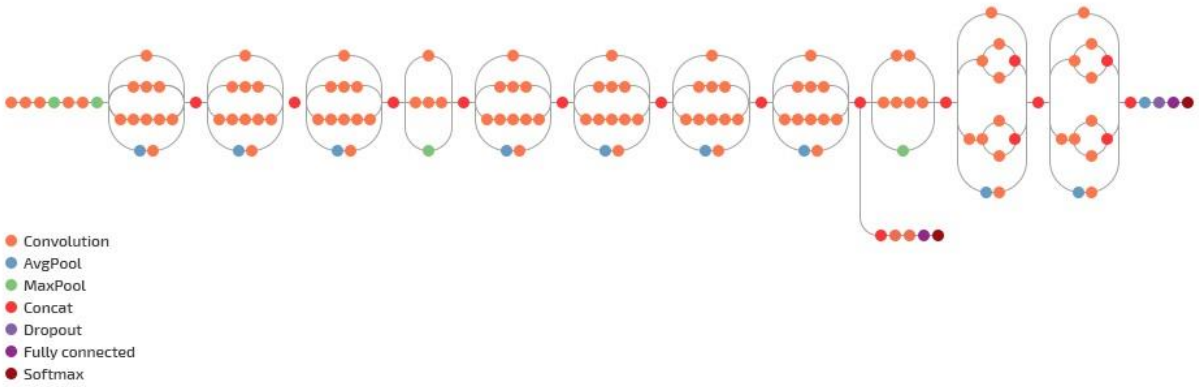


Figura 16. Arquitectura Inception-V3.

Las partes en las que las capas "se ramifican" y luego se fusionan de nuevo son los bloques inception descritos anteriormente.

3.4.2 Modelo ResNet

Uno de los problemas que ResNet resuelve es el famoso desvanecimiento del gradiente. Este problema ocurre cuando intentamos entrenar un modelo de red neural usando técnicas de optimización basadas en Gradiente, y es conocido en realidad, desde

hace 10 años para entrenar modelo de red neural profundas debido al largo proceso de entrenamiento y a la degradación de la precisión del modelo.

Lo que sucede es que a medida que añadimos más y más capas ocultas en el modelo, la velocidad de aprendizaje de las siguientes capas ocultas en el modelo se hace cada vez más rápida. El problema viene cuando empezamos a usar backpropagation. El gradiente es mucho menor en las capas anteriores. Como resultado, las primeras capas de la red son las más lentas de entrenar. Esto puede resultado en que los valores de los pesos nunca se actualicen y por lo tanto, no se esté realizando ningún aprendizaje.

Con ResNet, los gradientes pueden fluir directamente a través de las conexiones de salto hacia atrás desde las capas posteriores a los filtros iniciales.

Como ResNets puede tener tamaños variables, dependiendo del tamaño de cada una de las capas del modelo, y de cuántas capas tenga, seguiremos lo descrito por los autores en el artículo[14] - ResNet 34 - para explicar la estructura.

En la Figura 17 podemos ver que ResNet consiste en un paso de convolución y pooling (en naranja) seguido de 4 capas de comportamiento similar.

Cada una de las capas sigue el mismo patrón. Realizan una convolución de 3x3 con una dimensión de mapa de características fijas (F)[64, 128, 256, 512] respectivamente, pasando por alto la entrada cada 2 convoluciones. Además, las dimensiones de anchura (W) y altura (H) permanecen constantes durante toda la capa.

La línea de puntos está ahí precisamente porque ha habido un cambio en la dimensión del volumen de entrada (por supuesto una reducción debido a la convolución). Nótese que esta reducción entre capas se consigue mediante un aumento del stride, de 1 a 2, en la primera convolución de cada capa; en lugar de mediante una operación de pooling.

En la Figura 18, hay un resumen del tamaño de salida en cada capa y la dimensión de los núcleos convolucionales en cada punto de la estructura.

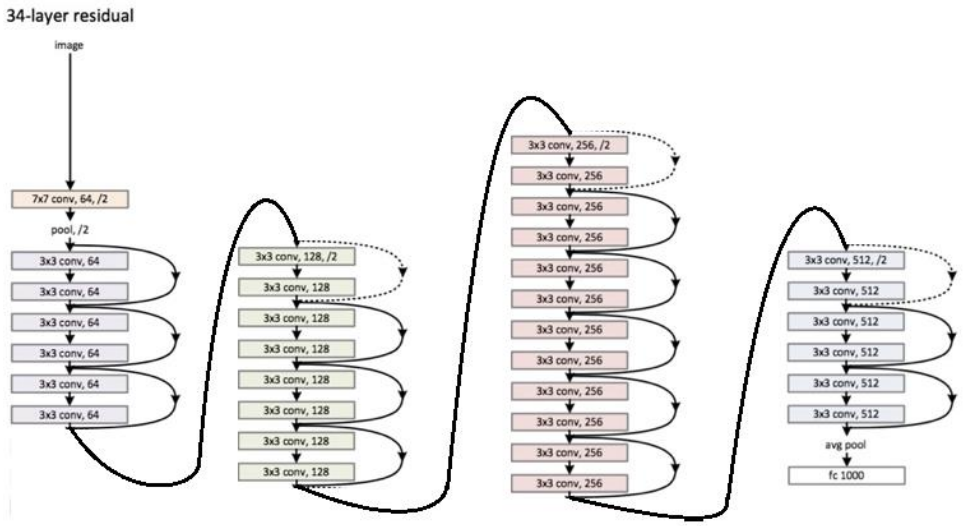


Figura 17. Arquitectura ResNet-34.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 18. Tamaños de salidas y núcleos convolucionales para ResNet 34.

3.4.3 VGG19

El modelo pre-entrenado VGG19 es el modelo que elegimos para implementar la técnica de “transfer learning”. Este modelo fue elegido porque es uno de los mejores cinco modelos de reconocimiento de imágenes de ImageNet es el más grande disponible actualmente, y por si sencillez para ser usado como extractor de características. El modelo VGG19 incluye 19 capas de aprendizaje profundo y pesa 574 mb. De todos los modelos existentes, era el único que podía usar en el ordenador portátil donde se realizó este trabajo. El mejor modelo para el entrenamiento sobre el conjunto de datos presentado, es ResNet, pero para el uso de este modelo es necesaria la décima versión de CUDA para GPUs de NVIDIA o superior, que desafortunadamente no tiene el equipo donde se realizó el trabajo.

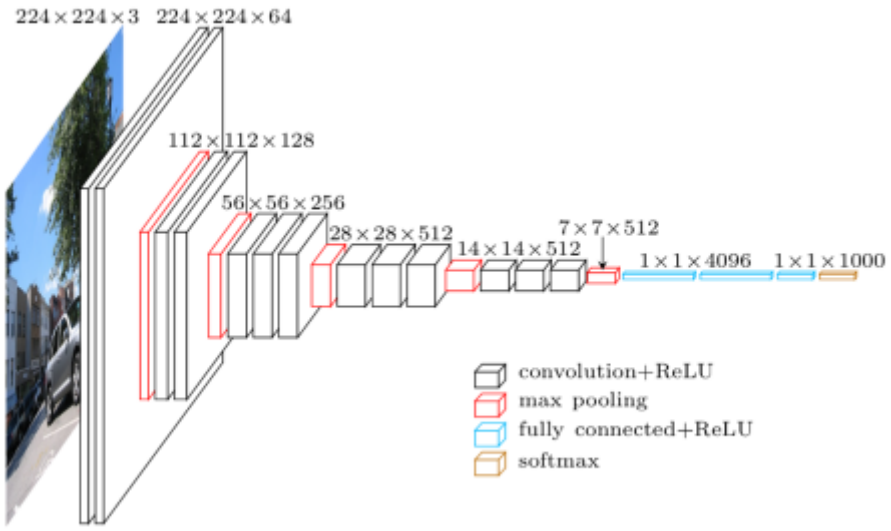


Figura 19. Arquitectura modelo VGG19.

4 Modelos desarrollados

El sistema de DL presentado en este proyecto está implementada utilizando diferentes arquitecturas de modelos CNN, para así ver diferencias en las eficiencias conseguidas en el reconocimiento de las categorías propuestas. Todo el sistema está implementado en Python haciendo uso de la librería TensorFlow y Keras, y su código y datasets queda expuesto en el siguiente link de Google drive (el Anexo A explica la relación de carpetas):

https://drive.google.com/open?id=1-I8PNd2ubaU4GyXxNVVaPNvSloZ8vcm3

La arquitectura global del sistema está basada en dos ideas proncipales: obtener la mayor cantidad de información posible de la imagen, y ayudarnos de esta información para calcular la secuencia de etiquetas más probable.

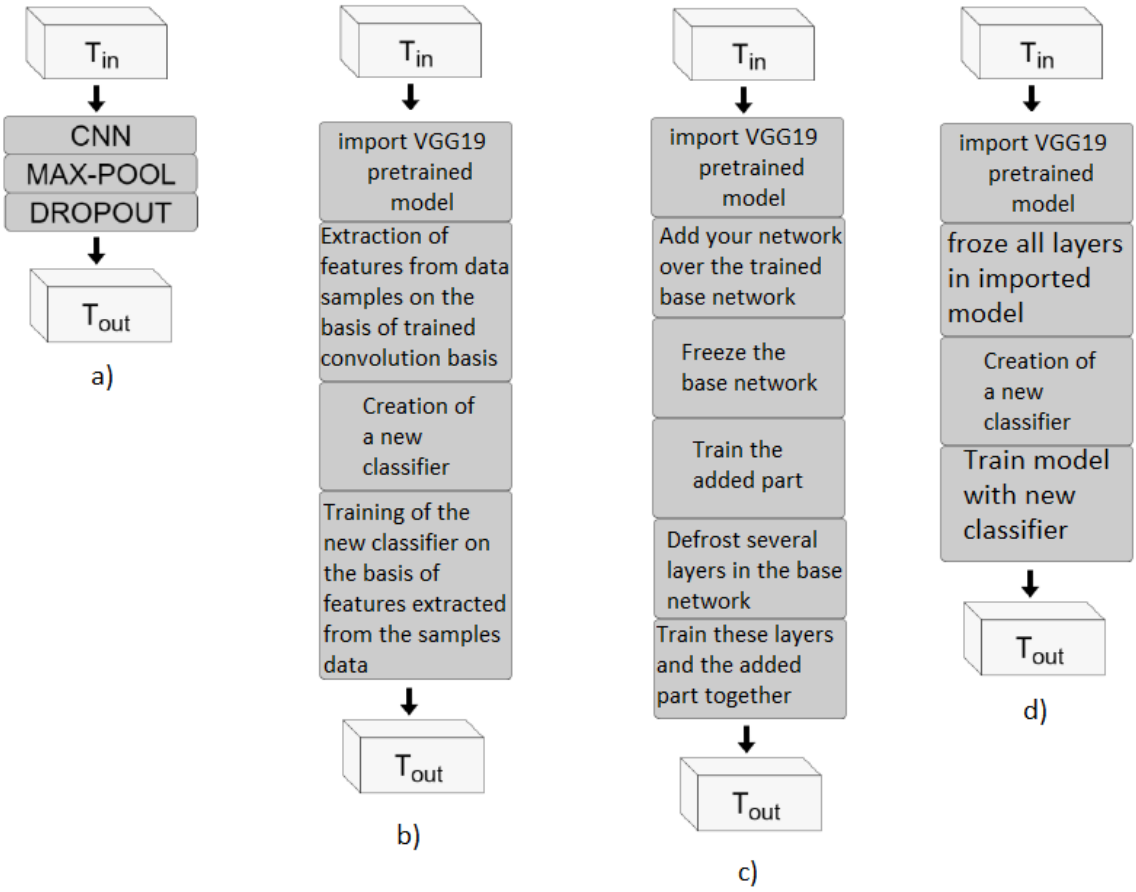


Figura 20. Arquitecturas que componen los modelos entrenados en el trabajo: a) modelo CNN creado desde cero, b) modelo de Transfer Learning entrenado con extracción de caracteristicas, c) modelo de Fine-Tuning, d) modelo de Transfer Learning con todas las capas congeladas

Para cumplir el primer objetivo y procesar la imagen en busca de información se han propuesto 4 arquitecturas por separado. Para todos los modelos presentados, se utilizó el mismo optimizador y la misma función de pérdida: RMSprop - optimizador con learning_rate - 1e-4 y 'categorical_crossentropy' - función de pérdida. El tamaño de batch para los modelos fue el siguiente: modelo CNN creado desde cero - 4, modelo Transfer Learning entrenado con extracción de caracteristicas - 8, modelo Fine-Tuning - 4, modelo

Transfer Learning con todas las capas congeladas - 4. Todos los modelos fueron entrenados utilizando GPU de NVIDIA GTX 960m con dos gigabytes de RAM.

Usamos estos modelos de CNN para entender cómo funcionan las redes convolucionales y qué características tienen, así como para mostrar la diferencia entre una red convolucional creada desde cero y una pre-entrenada.

Dado que una de nuestras principales tareas era crear un modelo basado en otro pre-entrenado, para entender qué modelo es el más adecuado para el reconocimiento de vídeo, se decidió crear tres modelos diferentes para ver la diferencia entre ellos y elegir el modelo que mostrara los mejores resultados.

Las configuraciones del modelo fueron elegidas de acuerdo a la información del libro[3].

4.1 Arquitectura del modelo CNN

Como ya hemos comentado en el punto 3.2 sobre el funcionamiento del modelo CNN, añadamos una descripción en que conciste nuestro modelo:

- CNN: Capa convolucional que permite establecer una relación entrada-salida mediante unos parámetros.
- Max-Pool[15]: Capa para reducir la dimensionalidad basada en el muestro de la entrada. El objetivo es muestrear la entrada tomando subconjuntos de elementos y seleccionando como salida el valor más elevado.
- Dropout[16]: El objetivo de esta última capa es "añadir ruido" durante el entrenamiento de la ANN, anulando, bajo una cierta probabilidad, las salidas de la capa anterior. Es una técnica de regularización que evita en cierta medida el problema de sobreajuste, cuando el sistema se adapta muy bien a las muestras que ya ha procesado, pero responde muy mal a las muestras desconocidas

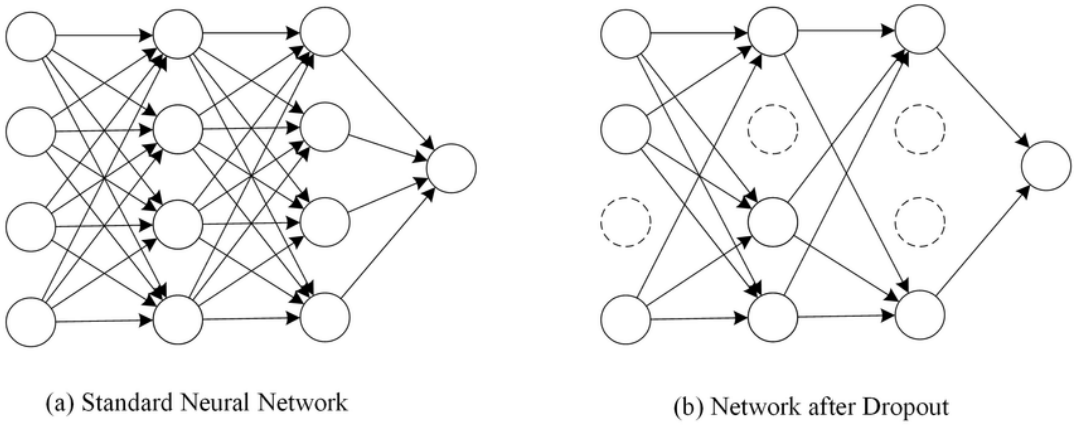


Figura 21. Ejemplo de Dropout.

Esta arquitectura es frecuentemente usada para el procesamiento de imágenes y generalmente obtiene buenos resultados en tareas de visión artificial.

4.2 Arquitectura basada en modelo preentrenado con extracción de características

Como ya se ha comentado, una red pre-entrenada es una red almacenada, previamente entrenada en un gran conjunto de datos, usualmente dentro del marco de una tarea de clasificación de imágenes a gran escala.

Debido a que este modelo de aprendizaje profundo permite la transferencia de los parámetros aprendidos entre diferentes modelos, se ha convertido en una gran ventaja del aprendizaje profundo sobre muchos métodos más antiguos de aprendizaje superficial, lo que hace del aprendizaje profundo una herramienta muy efectiva para resolver problemas con pequeñas cantidades de datos. En este trabajo usamos el modelo pre-entrenado VGG19, que es una red neuronal convolucional entrenada sobre más de un millón de imágenes de la base de datos de ImageNet[17]. La red tiene 19 capas de profundidad y puede clasificar imágenes en 1000 categorías de objetos. Como resultado, la red ha aprendido una gran cantidad de representaciones de características para una amplia gama de imágenes. La red tiene un tamaño de entrada de imagen de 224 x 224.

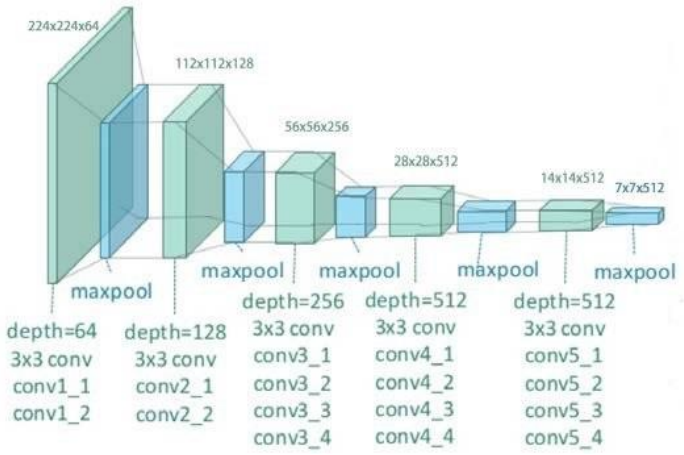


Figura 22. Arquitectura de modelo VGG19, mostrando nombre de capas y tamaños.

Resaltar que la técnica consiste en utilizar las características aprendidas por la red anterior para identificar las características de las nuevas muestras, que luego pasan por el nuevo clasificador, el cual se entrena desde cero.

Como se ha mostrado anteriormente, las CNNs utilizadas para clasificar las imágenes constan de dos partes: comienzan con una secuencia de capas de selección de valores y convolución y terminan con un clasificador totalmente coherente. La primera parte se llama la base convolucional del modelo. En el caso de las redes neuronales de convolución, el proceso de identificación de características consiste en tomar la base de convolución de la red pre-entrenada, pasar a través de ella nuevos datos y entrenar al nuevo clasificador sobre la base de la salida.

Esta arquitectura se basa en una pequeña red de dos capas densas, de 512 la primera y 5 la última (para clasificación), con una capa de Dropout entre ellas. Esta red se entrena no sobre las imágenes del dataset, sino sobre el mapa de características extraídas por VGG19 en su última capa (conv5_4) sobre las imágenes del dataset.

4.3 Arquitectura basada en el Ajuste Fino(Fine Tuning)

Otro método ampliamente utilizado de reutilización de modelos que complementa la selección de características es el ajuste fino (Figura 23). El reentrenamiento consiste en descongelar varias capas superiores del modelo congelado, que se utilizó para identificar las características, y el entrenamiento en su conjunto de la parte recién añadida del modelo (en este caso, el clasificador) y estas capas superiores descongeladas. Esta técnica se llama reentrenamiento porque corrige ligeramente las representaciones más abstractas en el modelo reutilizable para hacerlas más relevantes para la tarea.

Para el ajuste fino los siguientes pasos son necesarios:

1. Agregue su propia red encima de la red principal entrenada.

2. Congelar la red principal.

3. Añadir el nuevo clacificador.

4. Descongelar varias capas de la red principal.

5. Entrenar estas capas y la parte añadida.

Reentrenaremos las tres últimas capas de convolución, es decir, todas las capas por encima del block4_pool deben estar congeladas, y las capas del bloque block5_conv1, block5_conv2 y block5_conv3 deben estar disponibles para la entrenamiento.

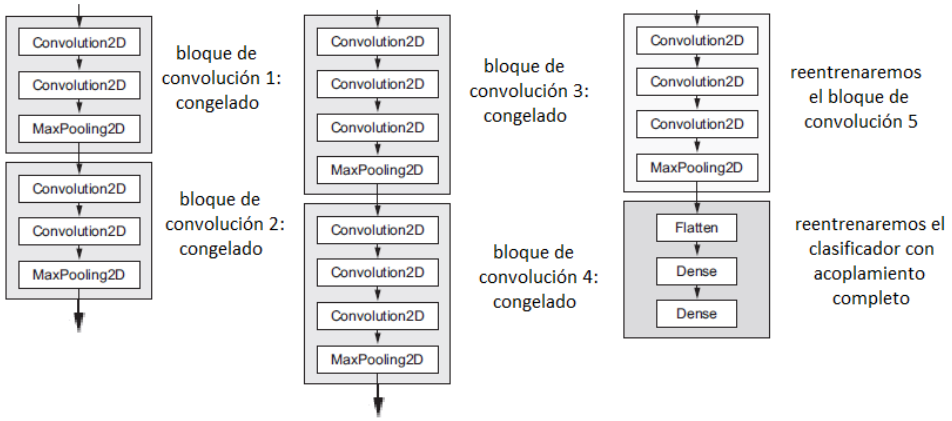


Figura 23. Esquema que muestra los cambios en la arquitectura de la VGG19 usando Ajuste Fino.

Después de realizar los pasos anteriores, se puede comenzar a volver a entrenar a la red. Para ello utilizamos el optimizador RMSProp con una velocidad de aprendizaje muy baja, en concreto, un learning rate de 1e-5. La razón de utilizar la baja velocidad de aprendizaje es la necesidad de limitar la cantidad de cambios realizados en las vistas de las tres capas recicladas. Demasiados cambios pueden dañar estos puntos de vista.

4.4 Arquitectura basada en el modelo preentrenado entrenado con todas capas congeladas

La arquitectura del modelo se muestra en la Figura 24. Nuestro modelo es uno que consiste en VGG19 importado sin su clasificador y con capas que hemos congelado previamente para que no se vuelvan a entrenar. También hemos añadido un nuevo

clasificador que consiste en una capa densa con 512 neuronas y una capa densa con 5 neuronas para clasificación con softmax, lo que a la salida da como resultado la pertenencia de la foto a una de las categorías definidas por nosotros. No se emplea Dropout.

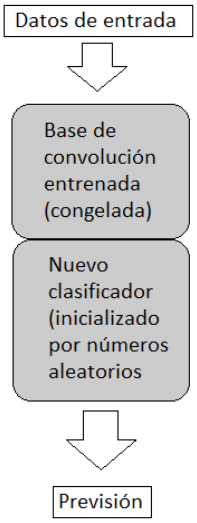


Figura 24. Arquitectura del modelo preentrenado con todas capas congeladas

La principal diferencia entre este modelo y el modelo con la extracción de características es que aquí los datos a la entrada de la red final no son las características de extracción de los fotogramas, sino los fotogramas en sí, y no utilizamos capa de Dropout. La capa Dropout no la hemos utilizado para ver cuando el modelo empieza a sobreajustarse.

5 Pruebas y Resultados

Sobre el sistemas se ha realizado dos pruebas para evaluar su rendimiento en el reconocimiento de las características de las imágenes: (1) datos especialmente elegidos(datos que han sido recortados del conjunto de datos principal utilizado en el entrenamiento del modelos) y (2) nuevos datos sacados de Youtube (un conjunto de datos compuesto por tráileres de películas, cada uno de los cuales está dominado por una de las clases seleccionadas de las 5 clases seleccionadas por nosotros al principio del entrenamiento de los modelos).

Para determinar la respuesta del modelo sobre un fotograma se utilizó la función *argmax*, que seleccionará el valor de una lista de cinco valores con la mayor probabilidad de la lista, donde cada valor es una posible pertenencia de un fotograma a cada clase.

5.1 Datos especialmente elegidos

Para esta prueba se ha propuesto el dataset creado para este trabajo que está disponible en el enlace de Google drive, enlace adjuntado arriba y en el Anexo A. Se han realizado una evaluación de cada modelo, con las mismas particiones para entrenamiento, validación y prueba. Cada modelo se ha entrenado con el 75% de las muestras del dataset, se ha probado con 15% y se ha validado sobre 15% de los datos de entrenamiento (que fueron separados antes de entrenar modelos, enviando en todo momento el solapamiento entre ellos).

Figura 25. Representación del dataset con imágenes mezcladas.

El etrenamiento del sistemas se ha realizado como sigue: los modelos se entrenaron durante 20 épocas, se llegó a hacer pruebas con 30 épocas pero el resultado era básicamente el mismo. Se ha tomado un lote de 3403 imágenes del conjunto de

entrenamiento, 741 imágenes para realizar validación, y 738 imágenes para pruebas sobre los datos que el modelo no ha visto antes. El número de epocas fue elegido para evitar sobreajuste, aunque esté en todos los modelos, pero de forma aceptable. Los resultados se muestran en los siguientes capítulos. Estas fotos se obtuvieron utilizando la función que corta el vídeo en fotogramas, cada una de las cuales representa un segundo del vídeo.

5.1.1 Resultados

5.1.1.1 Modelo CNN creado desde cero

La función de pérdida obtiene un resultado 1,32 y precisión 0,60. En el apartado 3.2.6.1. hemos descrito qué es la función de pérdida. Sobre la base de la información de este punto, podemos concluir que este valor de pérdida es demasiado alto, lo que resulta en un gran número de errores, es decir, las imágenes son poco reconocidas. Esto se puede ver en la siguiente Figura 26, donde se representada como matriz de confusión, que muestra cuántas imágenes de cada clase fueron reconocidas correctamente y cuántas fueron reconocidas como otra clase.

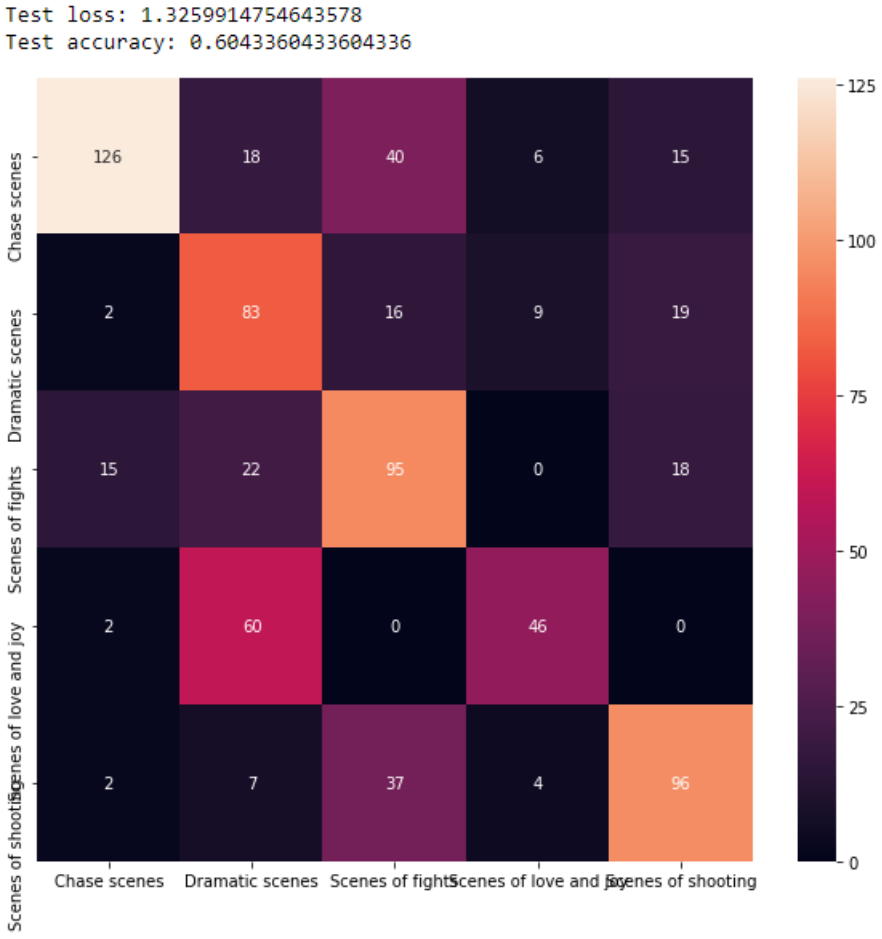


Figura 26. Matriz de confusión con resultados de uso del modelo CNN.

Se puede ver que las categorías de “Fight”, las escenas de “Love and Joy” y escenas de “Shooting” son más reconocidas que las otras. Este resultado demuestra que el modelo no estaba lo suficientemente capacitado para dar una evaluación precisa de todas las clases de materiales de vídeo. Este resultado se debe principalmente a la falta de datos utilizados para enseñar el modelo, así como a la contaminación de datos. Por ejemplo,

hay escenas de ¨Shooting” en escenas de “Chase”. En las escenas de “Shooting”, hay escenas de “Fight” entre los personajes de la película. En escenas caracterizadas como “Drama”, buscamos escenas en las que el héroe se encuentra en estado de ansiedad (pérdida de un familiar, traición por parte de amigos, etc.) y esta característica tiene como objetivo reconocer las imágenes de rostro principalmente triste, pero en la mayoría de los casos en las películas tal estado puede ser observado en el héroe en los momentos de peleas y tiroteos.

5.1.1.2 Modelo preentrenado VGG19 usando Ajuste Fino(Fine-Tuning)

En este caso podemos ver que la función de pérdida tampoco nos da un resultado bueno, pero hemos mejorado en la precisión un 20%, así que podemos decir que este modelo es mejor entrenado que el modelo anterior. En el Figura 27 se presenta una matriz de confusión que nos da la mejor manera de ver en qué momentos esta red es mejor que una CNN entrenada desde cero.

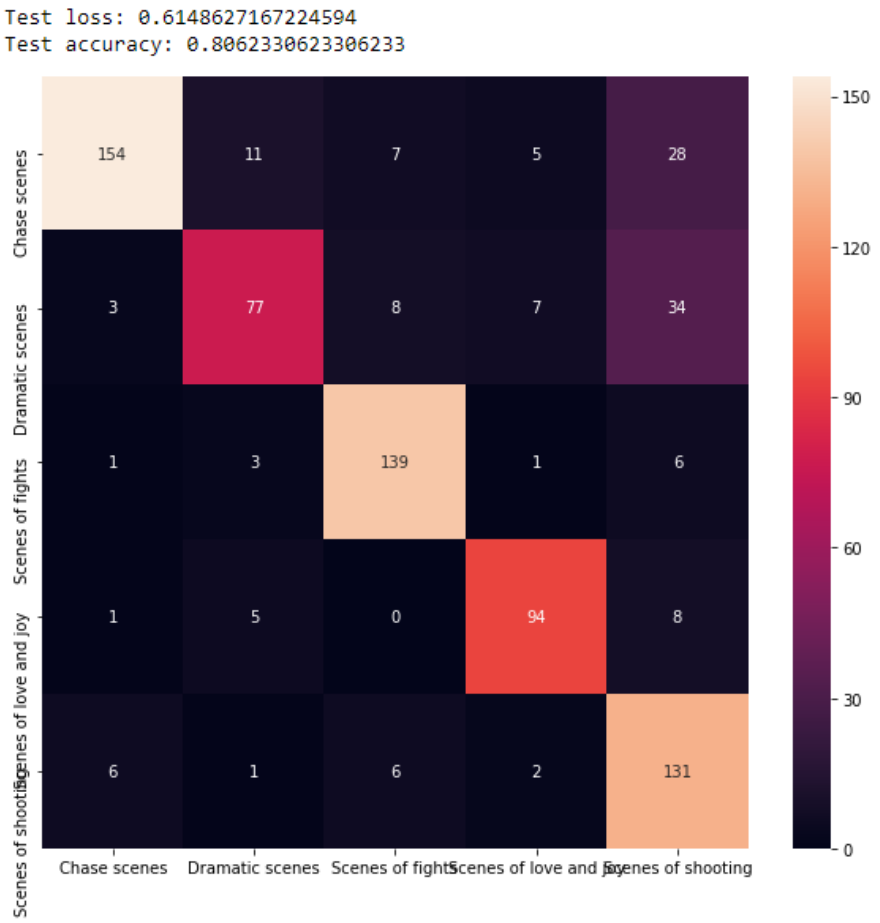


Figura 27. Matriz de confusión con resultados de uso del modelo VGG19 con Ajuste Fino.

Aquí podemos ver claramente que el modelo reconoce bien los vídeos que pertenecen a las clases de “Chase”, "Fights", "Love and Joy" y "Shooting", pero muestra un mal resultado en el reconocimiento del clase de vídeo "Drama". Esto se debe a que el vídeo Drama presenta momentos en los que un héroe de la película amenaza a otro héroe con

un arma, y por lo tanto estas escenas son clasificadas por el modelo como escenas de "Shooting".

5.1.1.3 Modelo pre-entrenado VGG19 entrenado con caracteristicas de fotogramas

En primer lugar, hemos mejorado la pérdida reduciendo su valor, así como aumentando la precisión en un 2%. En el transcurso de los experimentos, este modelo demostró ser el mejor en el muestreo de datos de entrenamiento. Los resultados se muestran claramente en la Figura 28.

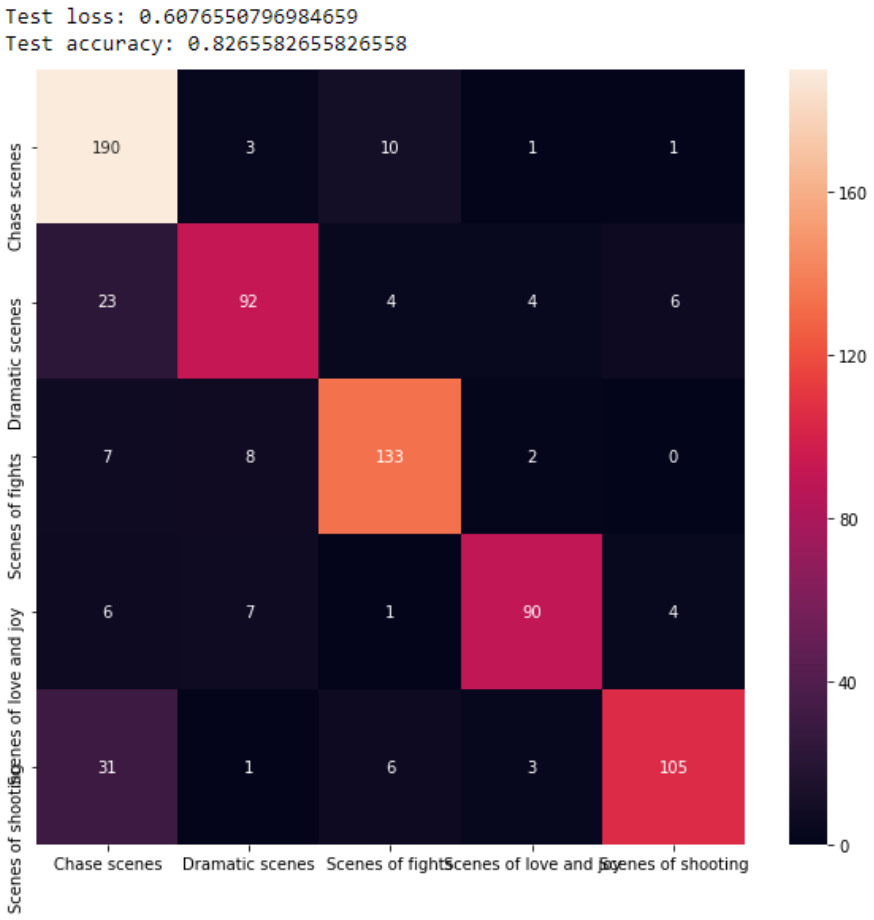


Figura 28. Matriz de confusión con resultados de uso del modelo VGG19 pasando por el modelo caracterícticas sacadas del fotogramas.

Las inexactitudes en las decisiones tomadas por el modelo se pueden atribuir al hecho de que en algunas fotos hay un gran número de objetos que interfieren con el proceso de toma de decisiones del modelo. Aquí vemos que el modelo lo ha hecho muy bien. El único problema son las escenas de clase "Drama" y “Shooting”. Algunas fotogramas fueron definidos como escenas de clase "Chase". En la clase "Drama" y "Shooting" sucedió porque en los fotogramas del conjunto de datos para estas clases hay escenas en las que el héroe conduce el coche.

5.1.1.4 Modelo pre-entrenado VGG19 entrenado con todas capas congeladas

A partir de los resultados obtenidos, se puede concluir que este modelo es el peor determinante de las clases de materiales de vídeo. En la Figura 29 se pueden ver los resultados más detallados.

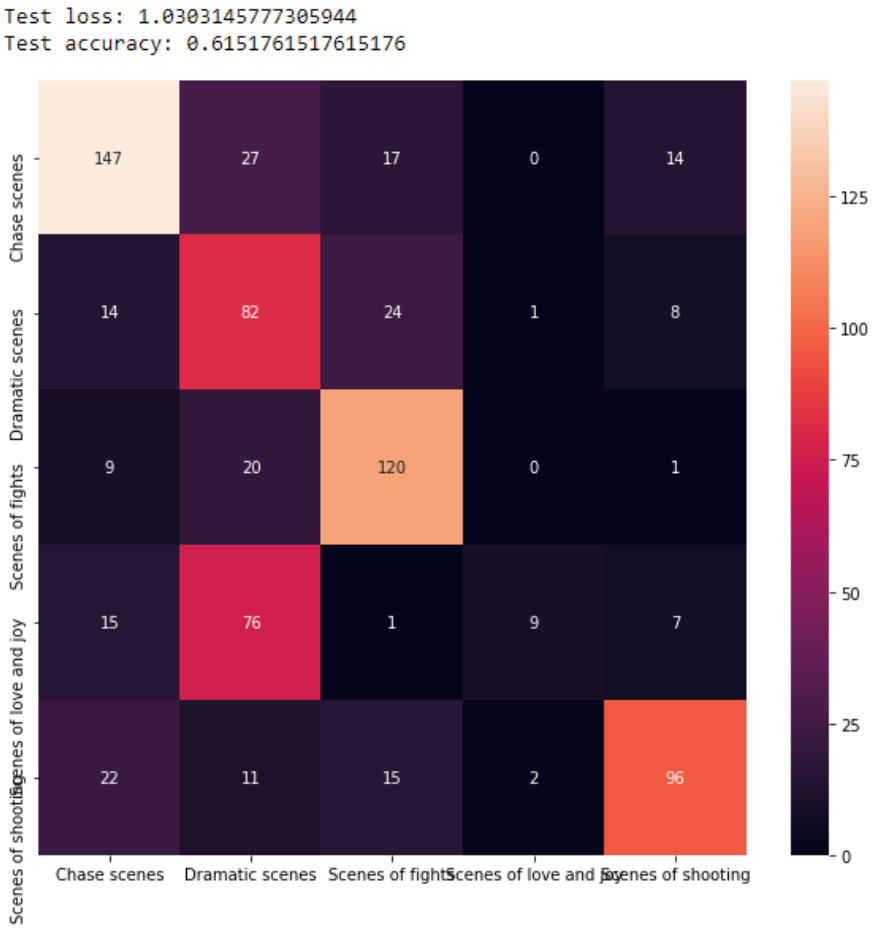


Figura 29. Matriz de confusión con resultados de uso del modelo VGG19 con capas congeladas.

Este resultado muestra que al entrenar un modelo pre-entrenado VGG19 sin extraer primero las características de las fotogramas, es decir, al entrenar con imágenes en lugar de sus características, y no usar las capas del Dropout el modelo no se adapta bien para tomar las decisiones correctas.

5.2 Nuevos datos sacados de Youtube

Hemos cogido 10 vídeos de youtube, dos vídeos por categoría. Los datos seleccionados no están limpiados de ninguna manera, y hay segmentos en cada vídeo que pueden pertenecer a categorías diferentes. Más detalles sobre los vídeos se pueden encontrar en el enlace de Google Drive descrito en el Anexo A, en las carpetas «new_data/video/». La primera carpeta «new_data/frames/video data1» contiene fotogramas de la primera carpeta con vídeo, divididos en 5 clases y con un número total de 870 fotogramas; la segunda «new_data/frames/video data2» carpeta contiene

fotogramas de la segunda carpeta con materiales de vídeo, divididos en 5 clases y con un número total de 722 fotogramas.

Para ver el resultado con más detalle, he creado un collage que contienen fotogramas elegidos por cada modelo para cada categoría. Estos collages se encuentran en la carpeta «Collage results/» donde “Collage 1” - collage de la categoría “Chase”, “Collage 2” - collage de la categoría “Drama”, “Collage 3” - collage de la categoría “Fights”, “Collage 4” - collage de la categoría “Love and Joy”, “Collage 5” - collage de la categoría “Shootings”. Cada collage pertenece a un modelo y se encuentra en la carpeta correspondiente a este modelo: modelo CNN creado desde cero - «Collage results/ cnn_video data1 y Collage results/ cnn_video data2», modelo con Ajuste Fino - «Collage results/vgg_ft_video data1 y Collage results/vgg_ft_video data2», modelo con extracción del características - «Collage results/vgg_tl2_video data1 y Collage results/vgg_tl2_video data2», y modelo preentrenado VGG19 con capas congeladas - « Collage results/vgg_tl1_video data1 y Collage results/vgg_tl1_video data2».

5.2.1 Modelo CNN creado desde zero

Los resultados sobre primer conjunto del videos (un video en cada clase) usando modelo CNN demuestran un resultado aceptable en categorias "Chase" y "Fights", pero un mal resultado en resto del categorias. Explicacion de resultados con mas detalles:

- En el primer video clase "Chase", hay escenas de persecución y robo a mano armada del banco, lo que remite esta parte del video a las escenas de la clase "Shootings", pero no hay escenas de la clase "Fights" en este caso, concluimos que el modelo se equivocó al determinar 159 segundos de vídeo en la escena "Fights".

- En la segunda clase de vídeo "Drama", hay escenas cortas que pertenecen a la clase "Fights" y "Chase", pero no hay escenas de la clase " Love and Joy ", llegamos a la conclusión de que el modelo dio el resultado incorrecto.

- En la tercera clase de vídeo "Fights" contiene principalmente sólo escenas de esta clase y no hay escenas que puedan pertenecer a la clase "Shootings". La conclusión es que el modelo estaba medio equivocado.

- En el cuarto vídeo de la clase "Love and Joy " hay varias escenas que se pueden atribuir a la clase "Fights", pero no hay escenas que puedan pertenecer a la clase "Shootings". El problema de la falta de reconocimiento de este vídeo también está relacionado con su corta duración.

- En el quinto vídeo de la clase "Shootings" hay escenas en las que los héroes se mueven sobre los coches, pero no hay escenas de la clase "Fights". La conclusión es que el resultado del modelo es incorrecto.

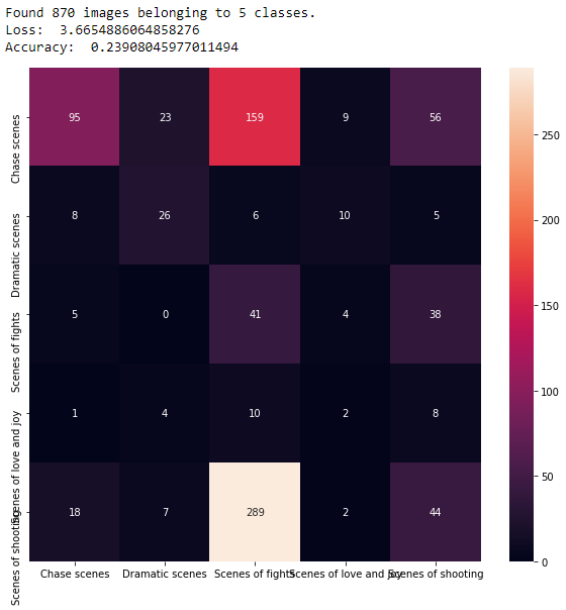


Figura 30. Matriz de confucion con resultados del primer conjunto del videos

Los resultados sobre segundo conjunto del videos (un video en cada clase) usando modelo CNN demuestran un resultado aceptable en categorias "Chase" y "Fights", pero un mal resultado en resto del categorias. Explicacion de resultados con mas detalles:

- En el primer vídeo de la clase "Chase" hay escenas que se pueden clasificar como "Fighting" y "Shooting", pero no contiene escenas de las clases "Drama" y "Love and Joy". Concluimos que en este caso el modelo reconoció bien este video, ya que la mayoría de las escenas fueron reconocidas correctamente.

- En el segundo vídeo de la clase "Drama" hay varias escenas en las que el héroe conduce un coche, y varias escenas de agresión, que podrían ser reconocidas por el modelo como clases "Chase" y "Fighting", pero no hay escenas con el contenido de la clase "Shooting". Conclusión - el vídeo no se reconoce correctamente porque no hay persecución real en las escenas y no hay escenas con peleas.

- El tercer vídeo de la clase "Fights" contiene escenas dramáticas y escenas en las que los héroes sonríen, pero en este caso se puede ver que el modelo no distingue entre las escenas "Fight" y "Drama" y reconocido casi todos las fotogramas como clase de “Drama”. Se puede concluir que el vídeo fue reconocido incorrectamente.

- El cuarto vídeo de la clase "Love and Joy" contiene escenas en las que los héroes viajan en coche por todo el país, pero no hay escenas de persecución. Esta categoría fue elegida incorrectamente por el modelo. Tampoco hay escenas de “Shooting” en el vídeo. Conclusión general - el video fue reconocido incorrectamente.

- En el quinto vídeo de la clase "Shootings" hay escenas de persecución y peleas, pero no hay escenas de las clases "Drama" y "Love and Joy". La conclusión general es que el vídeo se reconoce correctamente sólo parcialmente.

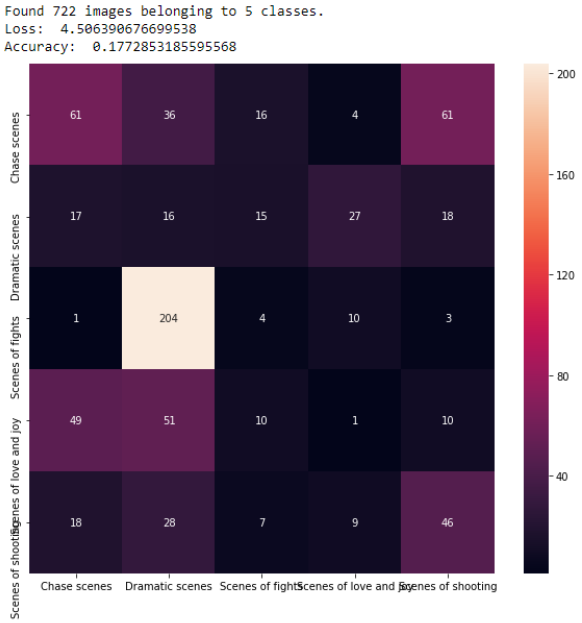


Figura 31. Matriz de confusión con resultados del segundo conjunto del videos

5.2.2 Modelo preentrenado VGG19 usando Ajuste Fino(Fine-Tuning)

Los resultados sobre primer conjunto del vídeos usando el modelo entrenado usando la técnica de Ajuste Fino demuestra un resultado aceptable en categorias "Chase" y "Shooting", pero un mal resultado en resto del categorias. Explicacion de resultados con mas detalles:

- En el primer vídeo de la clase “Chase”, como se describe en el párrafo anterior, hay escenas que se pueden atribuir a la clase de "Shootings" y "Drama", pero no contiene escenas de las clases de "Love and Joy" y "Fights". El modelo identificó muy bien las escenas de persecución y las escenas de tiroteo, pero fue un error clasificar 58 segundos de vídeo como clase "Love and Joy" y 11 segundos como clase "Fights". Concluimos que en este caso el modelo reconoció bien este vídeo, ya que la mayoría de las escenas fueron reconocidas correctamente.

- En el segundo vídeo de la clase de “Drama” hay varias escenas que podrían ser reconocidas por el modelo como clases de "Fights" y "Shootings", pero no hay escenas con el contenido de la clase "Chase" y " Love and Joy". Conclusión - el vídeo no se reconoce correctamente, porque las escenas definidas por el modelo como "Fights" deberían pertenecer a la clase "Drama".

- El tercer vídeo de la clase "Fights" contiene escenas dramáticas, pero no hay escenas de persecución y tiroteos. Estas escenas no fueron correctamente reconocidas por el modelo.

- El cuarto vídeo de la clase "Love and Joy" contiene varias escenas de peleas, pero no hay escenas de persecución, tiroteos y dramas. La conclusión general es que el vídeo no se reconoce correctamente.

- En el quinto vídeo de la clase "Shootings" hay escenas de persecución y peleas. Este vídeo fue reconocido por el modelo correctamente.

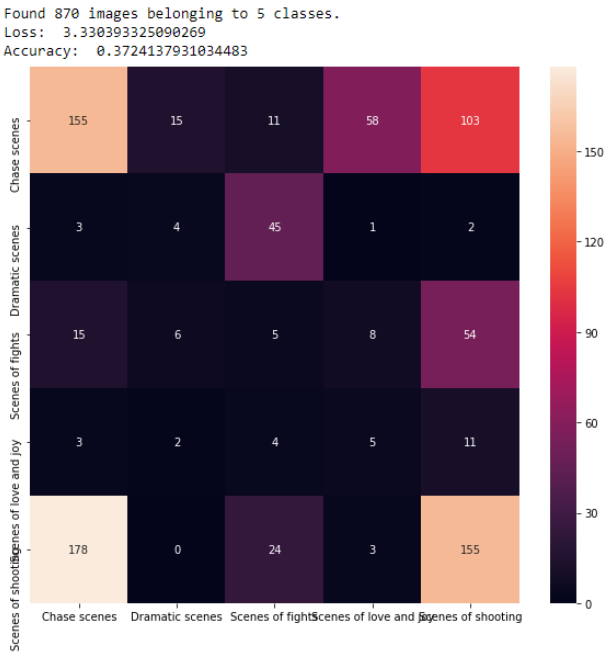


Figura 32. Matriz de confusión con resultados del primer conjunto del vídeos

Los resultados sobre el segundo conjunto del vídeos usando el modelo entrenado con técnica de Ajuste Fino demuestra un resultado aceptable en categorias "Chase" y "Fights", pero un mal resultado en resto del categorias. Explicación de resultados con más detalles:

- El primer vídeo fue reconocido correctamente, ya que hemos comentado que este vídeo contiene escenas que pertenecen a los clases "Chase", "Drama" y "Shooting".

- Al reconocer el segundo vídeo, el modelo cometió un error al definir las escenas en las categorías "Love and Joy " y "Chase". La conclusión general es que el vídeo no fue reconocido correctamente.

- El tercer vídeo de la clase "Fights" contiene escenas dramáticas y una escena en la que el autobús se mueve por la carretera, pero no hay escenas de persecuciones y tiroteos. El hecho de que el modelo clasificara la escena del autobús en la categoría "Chase" está justificado, ya que no tenía otra alternativa. En general, el vídeo fue reconocido correctamente por el modelo.

- El cuarto vídeo de la clase "Love and Joy " contiene varias escenas de peleas, pero no hay escenas de persecución, tiroteo y dramas. La conclusión general es que el vídeo no se reconoce correctamente.

- En el quinto vídeo de la clase "Shootings" hay escenas de persecución y tiroteos. El modelo llegó a una conclusión equivocada al definir la mayor parte del vídeo en las categorías "Drama" y " Fights".

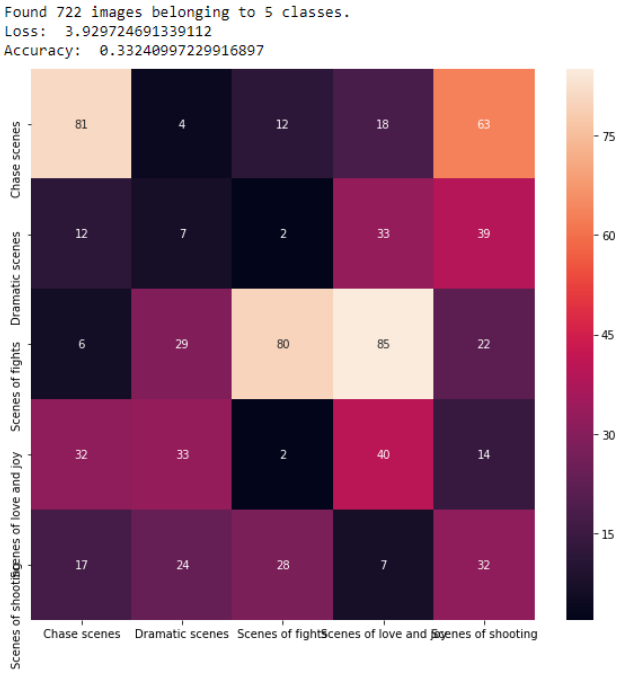


Figura 33. Matriz de confusión con resultados del segundo conjunto del vídeos

5.2.3 Modelo pre-entrenado VGG19 entrenado con caractericticas de fotogramas

Los resultados sobre primer conjunto del vídeos usando modelo entrenado con extracción de caractericticas demuestra un resultado aceptable en categoria "Chase", pero un mal resultado en resto del categorias. Explicación de resultados con más detalles:

- En el primer vídeo de la clase “Chase”, como se describe en el párrafo anterior, hay escenas que se pueden atribuir a la clase de " Shootings" y "Drama", pero no contiene escenas de las clases de " Love and Joy " y " Fights". El modelo identificó muy bien las escenas de persecución y las escenas de tiroteo, pero fue un error clasificar 30 segundos de vídeo como clase "Love and Joy" y 43 segundos como clase " Fights". Concluimos que en este caso el modelo reconoció bien este vídeo, ya que la mayoría de las escenas fueron reconocidas correctamente.

- Al reconocer el segundo vídeo, el modelo cometió un error al definir casi todos los vídeos como vídeos de lucha. La conclusión general es que el vídeo fue reconocido incorrectamente.

- El tercer vídeo de la clase "Fights" contiene escenas dramáticas, pero no hay escenas de persecución y tiroteo. El modelo determinó que el vídeo pertenece a las categorías "Chase" y "Shootings" - esta conclusión no es correcta.

- El cuarto vídeo de la clase " Love and Joy " contiene varias escenas de peleas, pero no hay escenas de persecución, tiroteos y drama. La conclusión general es que el vídeo no fue reconocido correctamente.

- En el quinto vídeo de la clase "Shootings" hay escenas de persecución y tiroteo. El modelo llegó a una conclusión equivocada al definir todos los fotogramas como "Chase".

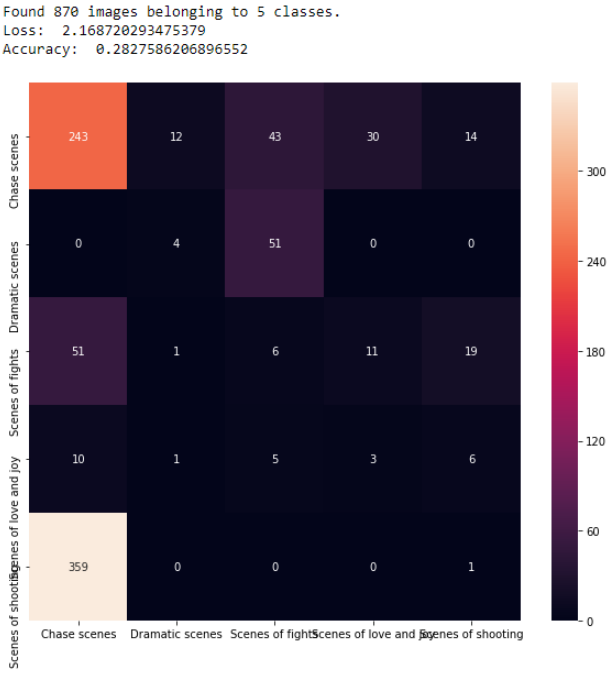


Figura 34. Matriz de confusión con resultados del primer conjunto del videos

Los resultados sobre segundo conjunto del vídeos usando el modelo entrenado con extracción de características demuestra un resultado aceptable en categoria "Drama" y "Love and Joy", pero un mal resultado en resto del categorías. Explicación de resultados con más detalles:

- En el primer vídeo de la clase "Chase" hay escenas que se pueden clasificar como "Shootings", "Drama" y "Fights", pero no contiene escenas de las clases "Love and Joy". La conclusión general es que el modelo determinó el resultado correcto sólo parcialmente.

- En el reconocimiento del segundo vídeo, el modelo llegó básicamente a las conclusiones correctas al incluir el vídeo en las categorías "Drama" y " Love and Joy ".

- El tercer vídeo de la clase "Fights" contiene escenas de drama, amor y alegría, el modelo dió la respuesta correcta clasificando 66 segundos de vídeo en la categoría "Fights". El modelo cometió un error al definir algunas peleas en la categoría "Drama", y tambien se ve que el modelo no aprendió la diferencia entre “Drama” y “Love and Joy”. La conclusión general es que el modelo dio un resultado correcto solo parcialmente.

- En el cuarto vídeo de la clase " Love and Joy", el modelo dio respuestas correctas clasificando vídeo en la categoría "Drama" y " Love and Joy", pero no en todos los fotogramas, en el “Collage 2” y “Collage 4” se ve, que el modelo se reconoce como “Drama” varias escenas que son del clase “Love and Joy”. En general, el vídeo fue bien reconocido.

En el quinto vídeo de la clase "Shooting" hay escenas dramáticas, pero ni la escena de tiroteo, ni la escena del drama fueron definidas por el modelo en las categorías correctas.

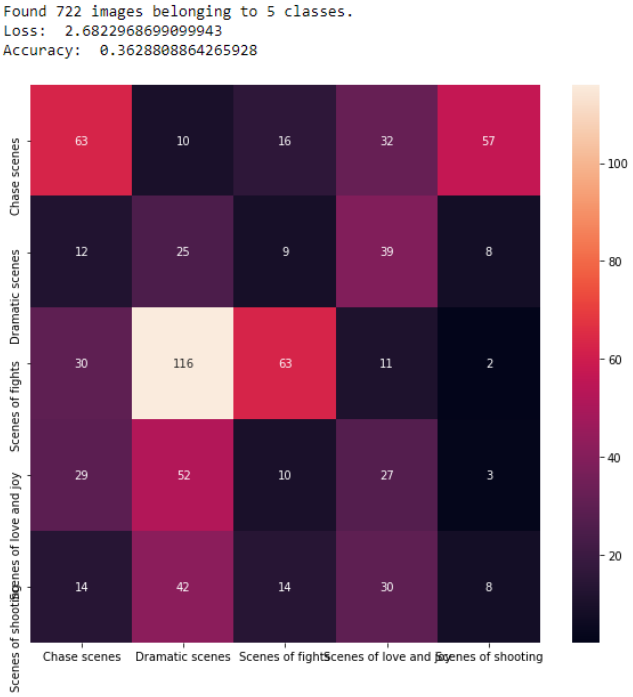


Figura 35. Matriz de confusión con resultados del segundo conjunto del vídeos

5.2.4 Modelo pre-entrenado VGG19 entrenado con todas capas congeladas

Los resultados sobre primer conjunto del vídeos usando el modelo entrenado con todas capas del modelo pre-entrenado congeladas demuestra un resultado aceptable en categoria "Chase", pero un mal resultado en resto del categorias.

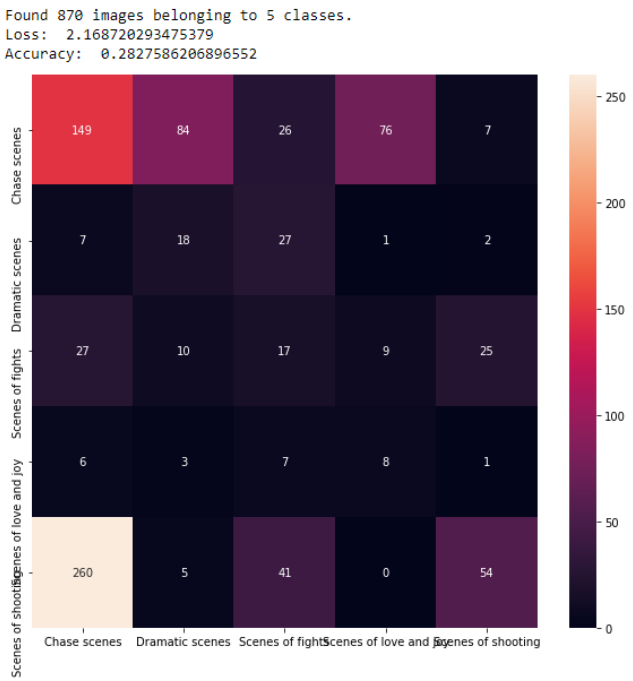


Figura 36. Matriz de confusión con resultados del primer conjunto del vídeos

Los resultados sobre el segundo conjunto del vídeos usando el modelo entrenado con todas capas del modelo pre-entrenado congeladas demuestra un resultado aceptable en categoria "Fights" y "Drama ", pero un mal resultado en resto del categorias.

El modelo no está bien entrenado para reconocer escenas de “Sgooting”, “Love and Joy” y “Chase”. Aunque el modelo identificó bien el vídeo de las categorías "Drama" y “Fights”, no aprendió a distinguir entre “Drama” y “Love and Joy”.

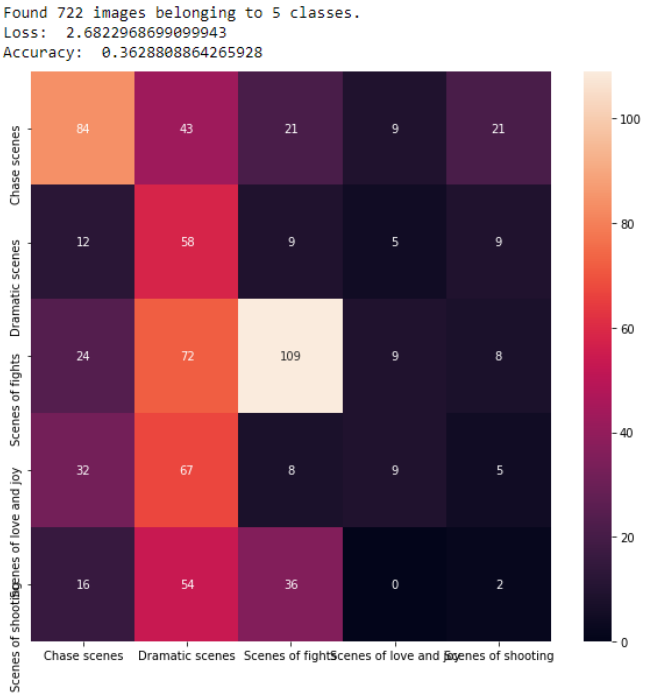


Figura 37. Matriz de confusión con resultados del segundo conjunto del vídeos

6 Conclusiones y Trabajos Futuros

Una vez expuesto el procedimiento seguido para abordar el problema y obtenidos los resultados de las pruebas realizadas es hora de considerar las conclusiones obtenidas y proponer posibles trabajos futuros relacionados, que o bien buscan solventar las carencias y errores que a lo largo de nuestra aproximación hemos podido realizar, o bien proponen vías alternativas para abordar el mismo problema.

Para llegar a las primeras conclusiones, hay que recordar que en el presente proyecto no se ha pretendido construir un sistema que resuelva el problema con una eficiencia del 100%, sino mostrar de una manera clara pero precisa cómo se afrontan este tipo de problemas desde el Aprendizaje Automático, desde la construcción de un dataset desde cero a presentar sistemas implementados con Tensoflow y Keras, las librerías de DL que mayor popularidad y porcentaje de uso tiene actualmente. Es por ello que se ha hecho un especial hincapié en el apartado teórico dejando en un segundo plano el apartado técnico y más relacionado con la implementación real del sistema, que no obstante queda incluido en los Anexos.

En el proceso de diseño del sistema se implementaron y probaron varios modelos. Empezamos con un modelo básico para familiarizarnos con el principio de las redes neuronales de tipo convolucional. Tal modelo da buenos resultados en una gran cantidad de datos, pero en nuestro caso no mostró muy buenos resultados. Dado que la ANNs permite el libre diseño de topologías, se han añadido varias variaciones del modelo básico, dando más sentido a la utilización de los modelos CNN.

Por ello, quiero destacar la universalidad de la ANNs, donde podemos empezar con una arquitectura muy sencilla y cambiarla estudiando el comportamiento de los sistemas de AA. Hay muchas topologías con funciones específicas y en combinación con ellas hay adaptabilidad de los algoritmos.

A medida que se modificó el modelo, los resultados han mejorado. Los estudios se realizaron a una velocidad(batch) de 4-8 y utilizando la GPU NVIDIA GTX 960m con dos gigabytes de RAM, sin duda aumentando este parámetro de velocidad de aprendizaje y utilizando una tarjeta gráfica más potente aumentará la velocidad de aprendizaje de los modelos, y para obtener los mejores resultados podemos utilizar los modelos pre-entrenados ResNet50.

El mejor resultado en el reconocimiento de vídeo, utilizando parte del conjunto de datos utilizado para el entrenamiento de los modelos, fue el modelo pre-entrenado VGG19 entrenado con extraccion de caracteristicas de fotogramas, que dio un resultado del 82,5% de las respuestas correctas. En el momento de prueba con nuevos datos descargados de YouTube, el mejor resultado se obtiene con dos modelos: (1) el modelo pre-entrenado VGG19 entrenado con extraccion de características de fotogramas reconoce bien los fotogramas del segundo conjunto de vídeo “new_data/frames/video data2”, que dio un resultado de 36% de las respuestas correctas con pérdida 2,68 y (2) el

modelo con Ajuste Fino reconoce bien los fotogramas del primer conjunto del vídeo “new_data/frames/video data1” con un resultado de 37% de las respuestas correctas con un valor de pérdida de 3,33. También vale la pena añadir que la prueba se basa en el número de vídeos correctamente reconocidos. El modelo pre-entrenado VGG19 entrenado con extraccion de características de fotogramas reconoció la mayoría de los vídeos: "Drama", "Fights", "Love and Joy", dando el resultado equivocado sólo para las escenas de "Shooting" y “Chase” y el modelo con Ajuste Fino reconoce bien las escenas que pertenecen a clases “Chase” y “Shootings”.

Las imágenes en nuestra base de datos que hemos usado para entrenamiento están bien emparejadas, como se puede ver en los resultados de los modelos, pero todavía hay algunas soluciones posibles que pueden mejorar este resultado: (1) ampliar la base de datos añadiendo nuevas imágenes, (2) realizando un fuerte preprocesamiento de las nuevas imágenes de los nuevos vídeos para hacerlos similares a los del conjunto de datos de entrenamiento. Tampoco podemos ignorar el hecho de que las categorías de películas en nuestros datos de entrenamiento no son todas las categorías posibles de películas que existen. Es posible que partiendo alguna categoría o añadiendo nuevas se pueda refinar mejor los resultados, ya que las clasificaciones sería sobre conceptos semánticos más reducidos. Además, se podría combinar varios modelos en forma de ensemble para mejorar los resultados finales, ya que como se ha comentado en el párrado anterior, el modelo sobre la extracción de características y el de ajuste fino son más precisos en clases distintas.

Como se puede observar este tipo de proyectos no tienen una aplicación práctica directa, es más bien una prueba de concepto algo elaborada, que puede suponer un punto de partida para construir modelos robustos de DL que sean capaces de resolver este tipo de problemas.

Y dejando a un lado los cambios y mejoras posibles para el sistema que concierne al proyecto, también se podrían plantear este tipo de modelos para otros problemas como el reconocimiento del los vídeos domésticos, deportivos o de vigilancia.

Referencias

[1] K. P. Murphy, Machine Learning: A Probabilistic Perspective, Cambridge, Massachusetts y London, England: The MIT Press, 2012.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv, vol. 1409, pp. 1–8, 2014.

[3] Francois Chollet, Deep Learning with Python, vol. 317, Manning Publications, 2017.

[4] N. Ejaz, U.A Khan, M.A. Martínez-del-Amor, H. Sparenberg. Deep Learning based Beat Event Detection in Action Movie Franchises, Moving Picture Technologies, Fraunhofer Institute for Integrated Circuits, 2018.

[5] U.A. Khan, N. Ejaz, M.A. Martínez-del-Amor, H. Sparenberg, "Movies Tags Extraction Using Deep Learning", IEEE AVSS, 2017.

[6] LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature, vol. 521, pp. 436–444, 2015.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In Advances in neaural information processing systems, p. 3320-3328, 2014.

[8] K. Hornik. Multilayer Feedforward Networks are Universal Approximatiors, Pergamon Press plc, 1989.

[9] J. West, D. Ventura. "Spring Research Presentation: A Theoretical Foundation for Inductive Transfer", Warnick, Sean, 2007.

[10] Y. Lin, T. Jung, Improving EEG-Based Emotion Classification Using Conditional Transfer Learning. Front Hum Neurosci, vol. 11, 2017.

[11] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. Unsupervised and Transfer Learning Challenges in Machine Learning, p. 19, 2012.

[12] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell. Deep convolutional activation feature for generic visual recognition, arXiv preprint, 2013.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. "Rethinking the Inception Architecture for Computer Vision", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, p. 2818-2826, 2016.

[14] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in CVPR, 2016.

[15] D. Scherer, A. Müller, S. Behnke, Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, Thessaloniki, Greece: 20th International Conference on Artificial Neural Networks (ICANN), 2010.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15, 2014.

[17] ImageNet. http://www.image-net.org.

Anexos

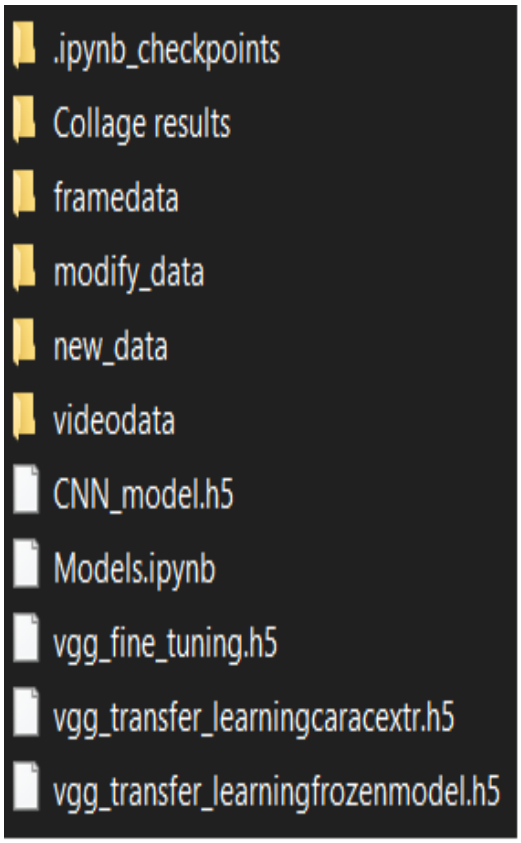
Anexo A. Código y datos de la implementación.

El código y el dataset desarrollado para este trabajo se puede encontrar en el siguente enlace de Google Drive:

https://drive.google.com/open?id=1-I8PNd2ubaU4GyXxNVVaPNvSloZ8vcm3

A continuación, se detallan las carpetas que componen el directorio compartido.

Directorio del Proyecto.



framedata: Directorio que contiene imágenes cortadas(frames).

new_data: Directorio que contiene nuevos dados descargados del YouTube (10 vídeos, 2 en cada clase).

videodata: Directorio con vídeo que he utilizado para crear dataset en la carpeta framedata.

Ficheros *.h5: Modelos guardados después del entrenamiento.

Models.ipynb: Contiene el código de todo el proyecto en forma de Jupyter Notebook.

Collage results: Esta carpeta almacena collages de fotos que el modelo ha definido en una de las clases (más detalles sobre la estructura de carpetas en la sección 5.2).