Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Área de Deep Learning

Deep Learning para la percepción y clasificación de sentimientos en imágenes

---

Autor: Jose María Bernet Fernández

Tutor: Anna Bosch Rué

Profesor: Albert Solé Ribalta

Sevilla, 8 de enero de 2020

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Percepción y clasificación de sentimientos en imágenes
Nombre del autor:	Jose María Bernet Fernández
Nombre del colaborador/a docente:	Anna Bosch Rué
Nombre del PRA:	Albert Solé Ribalta
Fecha de entrega (mm/aaaa):	01/2020
Titulación o programa:	Máster Universitario en Ciencia de Datos
Área del Trabajo Final:	Deep Learning
Idioma del trabajo:	español
Palabras clave	análisis, sentimientos, imágenes

Abstract

Sentiment analysis in text allow corporations and political parties to know if an user or voter has a positive, negative or neutral approach. However, media content has become much important nowdays and definetly more important than text, thus, image sentiment analysis is getting more popular. This work is focused on the study of the current context and what has been done in the last scientific papers and also a convolutional neural network is trained in order to predict sentiment in two different datasets.

Keywords: convolutional neural network, sentiment, analysis, deep learning

Resumen

El análisis de sentimientos en textos permite a corporaciones o partidos políticos conocer si el posicionamiento de un usuario o votante es positivo, neutral o negativo. No obstante, el contenido que se comparte en los diferentes medios de comunicación cada vez es más multimedia y menos textual, es por ello que ha empezado a tomar relevancia esta vertiente de análisis de sentimientos, esta vez en imágenes.En este trabajo se estudia el contexto actual, donde se han quedado los últimos artículos científicos y se define una red neuronal convolucional para el análisis de sentimientos en dos datasets diferentes.

Palabras clave: convolutional neural network, sentiment, analysis, deep learning

Índice general

Índice de figuras

Índice de Código

Capítulo 1

Introducción

En este primer capitulo se presentará por un lado la introducción al trabajo junto a las motivaciones y justificación del mismo así como la definición de objetivos y metodología.

1.1. Contexto y justificación del Trabajo

El análisis de sentimientos ha sido una de las aplicaciones de la ciencia de datos que más ha crecido en los últimos años, la posibilidad de poder adivinar en cierta medida la actitud (positiva o negativa) que puede tomar un usuario sobre un tema concreto es altamente utilizables por compañías de diversos sectores (marketing, atención al cliente) o incluso por partidos políticos.

Sin embargo, el uso de palabras para comunicarse por internet ha pasado a un segundo plano, el contenido multimedia (imágenes y video) ocupan gran parte del tráfico de cualquier red social y por tanto, se deben de modificar las estrategias de análisis de sentimientos para adaptarnos a las necesidades actuales.

1.2. Motivación personal

El análisis visual de sentimiento es un área que esta empezando a explorarse pero que aun sigue siendo poco común, los trabajos e investigaciones realizadas toman diversos puntos de vista e intentan resolver el mismo problema pero usando diferentes métodos, lo que no ayuda a fortalecer un mismo conocimiento sino que fragmenta la investigación.

Mi motivación con este proyecto, no es solo la de aprender y profundizar con el uso de redes neuronales lo aprendido en el máster sino la de intentar aportar mis ideas y descubrir posibles nuevas aplicaciones de esta tecnología, la cual estoy seguro llegará a estandarizarse y a tener una mayor relevancia.

1.3. Definición de los objetivos

El objetivo principal del proyecto es el de clasificar imágenes automáticamente en función de los sentimientos que provoquen al usuario que las visualiza. Para conseguir el objetivo final se divide en otros objetivos más pequeños:

- Investigar el estado del arte.
- Recopilar información de las diferentes tecnologías y procedimientos usados.
- Encontrar un dataset válido y suficientemente grande para el entrenamiento de la red.
- Desarrollar un modelo de red neuronal usando Python y Keras (TensorFlow).
- Comparar los resultados así como las ventajas sobre otros modelos.

1.4. Descripción de la metodología

Dado que se trata de un proyecto de minería de datos y las peculiaridades de este área, usar una metodología enfocada a otro tipo de proyectos pueden suponer un sobre-esfuerzo innecesario, por ello y tras un proceso de búsqueda se decide utilizar la metodología CRISP-DM, la cual es ampliamente usada en este tipo de proyectos.

1.5. Planificación

La planificación inicial del proyecto viene determinada por la planificación propia de la asignatura, es decir por la entrega de las diferentes PECS, a saber:

Fase	**Descripción**	**Entrega**
1	Definición y planificación del trabajo final	29/09/2019
2	Estado del Arte	29/09/2019
3	Diseño e implementación del trabajo	21/12/2019
4	Redacción de la memoria	08/01/2020
5	Presentación y defensa	14/01/2020

Capítulo 2

Estado del Arte

Este capitulo empezará haciendo una breve introducción de lo que son las emociones y los sentimientos, después pasaremos al análisis clásico de sentimientos en textos y continuaremos con el salto al análisis de imágenes para terminar hablando del estado del arte actual, donde se ha llegado y que se ha conseguido.

2.1. Origen Análisis de Sentimientos

Para empezar a tratar el análisis de sentimientos empezaremos por su propia definición oficial, según la RAE el sentimiento, es, el "Estado afectivo del ánimo" (RAE) dicho de un modo más técnico, podemos definir una emoción como el resultado (o output) a una serie de estímulos externos (o inputs) y estos estímulos a su vez son procesados por nuestro cerebro transformándolos en sentimientos.

En realidad y fijándonos desde un punto de vista práctico, las emociones son la respuesta inconsciente, de nuestro organismo frente a diversas situaciones de nuestro día a día, son incontrolables pues se crean a raíz de un proceso de aprendizaje que tiene el ser humano durante toda una vida, sin embargo, los sentimientos son la interpretación que hacemos de la emociones después de procesarlas, somos conscientes de ellos y nos condicionan a la hora de tomar decisiones.

Tal y como acabamos de definir los sentimientos, debemos de prestar especial atención al poder de los sentimientos sobre el condicionamiento humano, no solo a la hora de tratar las emociones introspectivamente sino sobre todo de como estos sentimientos nos hace interactuar con el medio, si nos sentimos felices es probable que tomemos acciones diferentes a cuando nos sintamos tristes.

De este modo podemos interpretar los sentimientos cómo el motor metafísico de nuestras acciones, por consiguiente, intentar comprender por ejemplo un texto como una opinion o

una publicación puede afectarnos de un modo u otro es algo que las grandes compañías así como partidos politicos están empezando a usar para conocer mucho más a los consumidores o votantes.

2.2. Sentimientos en Imágenes

El uso de las redes sociales y de los diferentes sitios de opinión sigue creciendo año tras año, casi el 42 % de la población mundial hace uso de estas tecnologías, invirtiendo casi una hora y media cada día, además, la tendencia esta en continuo aumento, con el tiempo cambia la forma en la que nos comunicamos, pero no la necesidad de hacerlo. Esto no es solo beneficioso para los usuarios por la facilidad y rapidez de comunicación sino que la información que generan con su uso es un valor muy importante para muchas grandes compañías quienes están transformándolo en un activo más.

El análisis de la opinión o del sentimiento, no es algo novedoso [13] [1] ya hablaban de él hace 15 años y fue cogiendo fuerza en esta última década, con aplicaciones en el mundo del consumo, la política etc.

Sin embargo, por todos es sabido el dicho "Una imagen vale más que mil palabras" y es que, el contenido multimedia es cada vez más frecuente en las redes, hasta el punto de que muchísimas redes sociales basan su mecanismo en compartir imágenes (Instagram, Flickr) es por esto que el siguiente paso a dar era el del análisis visual de sentimientos. Esta no era una tarea fácil ya que se empezó con el análisis a bajo nivel de las imágenes, esto es el uso de características como GIST, LBP, BoW o el histograma las cuales aunque alcanzaban un nivel medio de precisión estaban muy lejos de ser perfectas.

En 2013 [2] da un paso más el análisis visual del sentimiento y abre la puerta a futuros estudios con la creación de SentiBank, logrando una representación a medio nivel con el uso de ANP (Adjetive Noun Pairs) es decir, pares de adjetivos y sustantivos que expresan 24 emociones diferentes extraídas de la rueda de las emociones de Plutchik . En una primera instancia se logro superar hasta en un 13 % la precisión de la predicción usando regresiones logísticas y SVM con SentiBank frente a los métodos de bajo nivel convencionales (Histograma, GIST, LBP).

A raíz de SentiBank, diversas investigaciones trataron de mejorar los resultados, por ejemplo [6] que intenta mejorar la relación de conceptos visuales con pares de adjetivos y sustantivos que son estadísticamente más probables de unirse, por ejemplo, "coche feliz" no sería candidato y si lo sería "perro feliz", además de una mejor enfoque en la localización de objetos prestando especial atención en la parte de la imagen que contiene información y no en el conjunto.

El siguiente paso importante lo dieron parte del mismo equipo que en 2013 creo SentiBank, con la creación de DeepSentiBank [5] un nuevo giro en él método de clasificación, esta vez

usando Redes Neuronales Convolucionales (CNN) que habían cogido especial repercusión tras el uso de ellas en el reto de reconocimiento de objetos ImageNET de 2012 donde la red AlexNET logro superar con creces a su antecesor. La utilización del modelo CNN basado en el framework de deep learning Caffe, consiguio mejorar aun más la precisión del modelo y reducir el numero de errores.

Durante los siguientes años, se emplearon diferentes técnicas cómo CNN progresivas (PCNN) [14], redes neuronales de adjetivo y sustantivos profundamente acopladas [12] ademas de diversos estudios de fine-tunning, para lograr el ajuste perfecto. [4][3]

Un estudio que se salió un poco de lo que estaba siendo la senda convencional pero que igualmente es destacable fue [7] quienes salieron del tan repetitivo uso de las CNN y sus limitaciones a la hora de predecir los sentimientos y propusieron un nuevo modelo donde se predicen dos valores continuos diferentes Valencia (Negativo/Positivo) - Excitación (Calmado, Exitado) de este modo pueden expresar un mayor rango de sentimientos. El modelo resultado usaba diferentes características: Color, Local, Objetos y Segmentación semántica. De modo que usando características de bajo y alto nivel se complementen y hace mejorar enormemente el rendimiento al reconocer emociones.

En 2018 hubo un nuevo giro con la llegada de un nuevo concepto WSCNet (Weakly Supervised Coupled Networks) [11] el cuál resuelve el gran problema que supone el etiquetado de imágenes. El modelo presentado es una CNN formada por dos ramas, la rama de detección y la rama de clasificación, en primer lugar la rama de detección genera un mapa de sentimientos el cual es acoplado en la rama de clasificación con el fin de poder obtener el sentimiento. El resultado fue una mejora de precisión en 7 datasets distintos en comparación con otros modelos anteriormente presentados como el DeepSentibank.

Un último articulo presentado en octubre de 2019 basado en el método anterior WSCNet [Weakly Supervised Learning of Image Emotion Analysis Based on Cross-spatial Pooling] [10] intenta mejorar los resultados obtenidos.

Capítulo 3

Especificaciones

3.1. Hardware

Dada la naturaleza de este proyecto, al igual que en todos los proyectos de machine learning, el Hardware es un factor clave y fue el primer obstáculo que se tuvo que sortear pues mi equipo personal, con las siguientes características:

- CPU: 2,3 GHz Intel Core i5
- RAM: 8 GB 2133 MHz LPDDR3
- Almacenamiento: Intel Iris Plus Graphics 655 1536 MB
- GPU: 250gb SSD

No llegaba a lo mínimo exigido para el desarrollo de un proyecto como este en el que se entrenarían cientos de imágenes y se probarían diferentes configuraciones de hiperparámetros lo que supondría un coste elevado en tiempo.

Con esta premisa, se buscaron soluciones alternativas en la red, de todas las disponibles elegí solo 4 para analizar en más profundidad, la primera característica a mirar era que tuvieran una versión gratuita, estas 4 cumplían esta característica y además tenían más renombre en la comunidad data scientist.

- Azure Notebooks
- Kaggle
- Amazon Sagemaker
- Google Colab

El siguiente paso fue comprobar las limitaciones que una version gratuita pudiera tener en cada uno de los entornos y tuvimos que descartar Azure y Amazon ya que los planes gratuitos limitaban las specs de la maquina. Solo nos quedaban dos alternativas Google Colab y Kaggle, me decante por Colab por encima de Kaggle por la fácilidad de integración con el resto de productos de Google como por ejemplo Google Drive para almacenar los Datasets y guardar los modelos entrenados.

Las características de Google Colab son las siguientes:

- GPU: 1xTesla K80 , having 2496 CUDA cores, compute 3.7, 12GB(11.439GB Usable) GDDR5 VRAM
- CPU: 1xsingle core hyper threaded i.e(1 core, 2 threads) Xeon Processors @2.3Ghz (No Turbo Boost) , 45MB Cache
- RAM: 12.6 GB Available
- Disk: 320 GB Available

Aunque contaba con limitaciones temporales, el tiempo máximo de ejecución son 12 horas continuas, a partir de esas 12 horas tu maquina asignada se reinicia y debes de lanzar de nuevo, además, debes de estar trabajando con el notebook en primer plano pues a los 90 minutos en segundo plano se reinicia también, no obstante lo mejor es ir haciendo copias de seguridad del modelo con el entrenamiento conseguido y partir de ese modelo en la próxima iteración.

3.2. Software

La búsqueda de las librería, framework o paquete a usar fue más sencilla, quería aplicar los conocimientos aprendidos en la asignatura de Deep Learning, para ello decidí usar Python como lenguaje ya que tengo experiencia laboral extensa a parte de la educativa.

Como framework para el desarrollo de la red neuronal quería usar Keras con backend TensorFlow ya que, no solo es con la que tenía más experiencia sino que además lo consideraba un mayor reto ya que la gran mayoría de artículos encontrados basaban su trabajo en Pytorch, de este modo también abro una nueva rama en el desarrollo del análisis de sentimientos usando esta configuración, la cual espero que pueda servir de ayuda en trabajos futuros míos y de cualquiera que se interese.

Una vez elegido el software se establece la versión más estable posible para evitar problemas de compatibilidad, las versiones que se eligieron son:

- TensorFlow Version: 1.15.0

- Keras Version: 2.2.4-tf

Con especial atención en que la version de Keras es la que viene integrada en TensorFlow y no es un paquete diferente. La version de Python usada es la 3.6.9.

Capítulo 4

Arquitectura

Este capitulo hace un repaso de la arquitectura CNN para después explicar la arquitectura elegida en base a las investigaciones realizadas sobre el estado del arte.

4.1. Breve historia de las Redes Convolucionales

La primera vez que se mencionaron fue en 1998 en un paper de Yann Leun's con su red LeNet [9] , originalmente fue usada por bastantes bancos para digitalizar los números escritos a mano en los cheques y aunque fue bastante pionera tenía varias limitaciones, la principal era que el aumento de resolución en las imágenes de entrada aumentaba el numero de capas convolucionales de red, esto la hacia computacionalmente muy costosa, aun así, supuso las bases de lo que hoy conocemos como redes convolucionales.

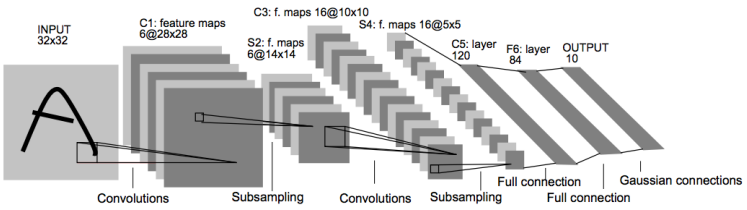


Figura 4.1: Arquitectura red LeNet.

No fue hasta 2012 cuando la red AlexNet [8] consiguió el mejor resultado en la competición ImageNET(ILSVRC) y supuso el renacimiento de estas redes, ahora sí, con una estructura más versátil y simple. Algunas diferencias con LeNet era el uso de MaxPooling y ReLU en vez de Average Pooling y Sigmoid o Tanh, un diseño con más capas y unidades (60m parámetros aprox.) y por su puesto el uso de GPU que aumenta considerablemente la rapidez de los entrenamientos que usando CPU.

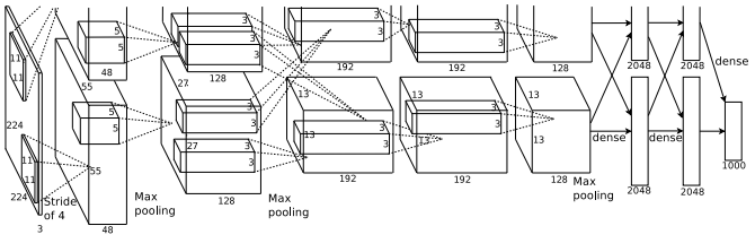


Figura 4.2: Arquitectura red AlexNet.

4.2. ¿Cómo funcionan?

Las redes convolucionales se han convertido en los últimos años en la arquitectura ideal para los problemas de machine learning con imágenes, la forma en la que se conectan las diferentes capas y la utilización de los filtros (kernels) se asemeja a la del córtex visual.

De manera general, las CNN están formadas principalmente por: una capa de entrada (input), varias capas de convolución con sus funciones de activación, varias capas capa de pooling (max, avg ...) que reducen la información de la imagen dejando solo la información más relevante y una Fully Connected Layer (FC) que traduce la salida de la CNN a un array de probabilidades para las diferentes etiquetas.

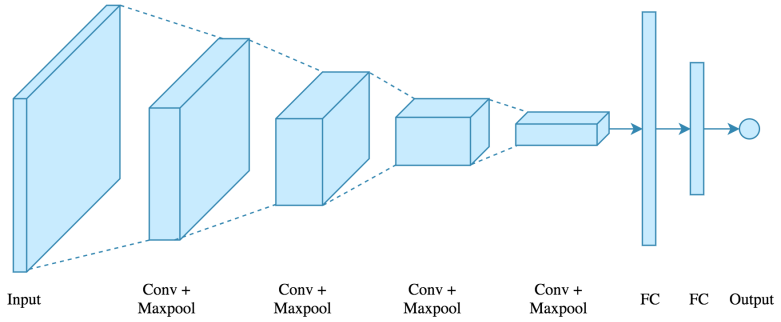


Figura 4.3: Estructura red convolucional..

4.3. Image Augmentation

Al tener un conjunto de entrenamiento finito, que además no es muy extenso, es muy probable que exista overfitting, uno de los métodos más comunes para reducir el overfitting cuando se trabaja con imágenes es el de Image (o data) Augmentation. Las imágenes que se usan para entrenar son sometidas a una serie de alteraciones (rotaciones, recortes, zoom, brillo) que dan como resultado a una imagen aparentemente nueva que ya esta etiquetada, eso sí, con

control y cuidado de no perder la característica identificativa que nos interesa para nuestro aprendizaje.

4.4. Arquitectura de la red

Tras la investigación realizada en la etapa estado del arte y las sucesivas iteraciones sobre la misma era más que evidente que la utilización de CNN iba a ser el punto de partida de nuestra red, sirviendo de especial referencia los últimos avances en esta rama por parte de los artículos [11] [10] donde ambos hablan de la unión de una FCN (Fully Convolutional Network) con una capa de Pooling y la Dense con activación softmax para la clasificación.

Sabiendo que debíamos elegir una red FCN como entrada a nuestro modelo, elegimos las posibilidades que teníamos dentro del paquete applications de Keras, encontramos las siguientes:

- Xception
- VGG16
- VGG19
- ResNet, ResNetV2
- InceptionV3
- InceptionResNetV2
- MobileNet
- MobileNetV2
- DenseNet
- NASNet

Al buscar información sobre ellas dimos con el siguiente gráfico en el que se puede comparar fácilmente la precisión (ranking imagenet) frente al número de operaciones y el número de parámetros (tamaño). Lo ideal era una red ligera pero efectiva, descartamos las VGG y las Inception por el número de operaciones ya que no teníamos recursos ilimitados y finalmente elegimos una red ResNET, en concreto la de 101 capas, no tan costosa como la de 152 pero bastantes puntos de precisión por encima de las de 34 y 50.

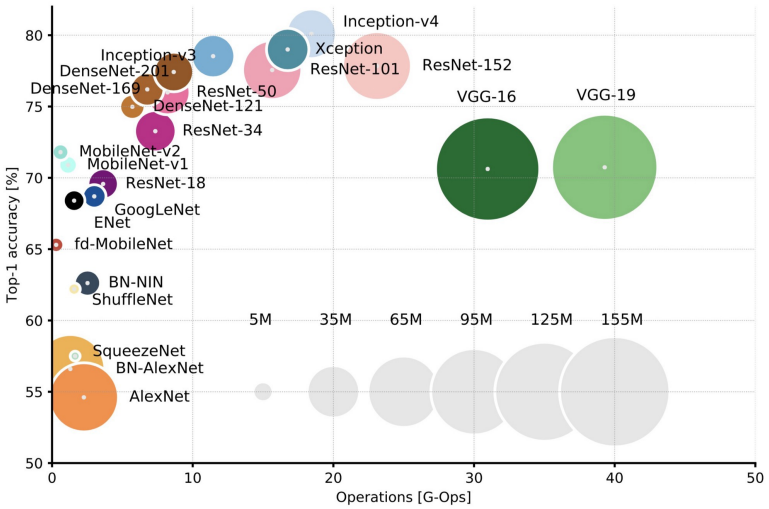


Figura 4.4: Comparación convolucionales.

El paquete applications de Keras tiene la opción de usar el modelo ResNET preentrenado con ImageNET por lo que nos ahorramos su entrenamiento y nos centramos en entrenar el resto del modelo, además el parámetro include_top nos permitiría añadir una fully connected layer (FCL) en la salida de ResNET aunque para nuestro modelo definiremos el resto de capas nosotros mismos.

El siguiente elemento de nuestro modelo será añadir una capa de pooling, en concreto una capa GAP (Global Avarage Pooling) que será la encargada de encontrar las diferentes zonas de la imagen que activarán una categoría o otra. Por último una capa Dense con el mismo tamaño que categorías a predecir.

4.5. Código

A continuación vamos a desglosar las diferentes partes del código que componen la aplicación. Se ira representando el trozo de código seguido de una explicación de su utilidad, además de sus parámetros de entrada y salida.

```
1 def load_and_crop_img(path, grayscale=False, color_mode='rgb', target_size=
    None, interpolation='nearest'):
2     """Wraps keras_preprocessing.image.utils.loag_img() and adds cropping.
3     Cropping method enumarated in interpolation
4 
5     # Arguments
6         path: Path to image file.
7         color_mode: One of "grayscale", "rgb", "rgba". Default: "rgb".
```

```
8             The desired image format.
9         target_size: Either `None` (default to original size)
10            or tuple of ints `(img_height, img_width)`.
11        interpolation: Interpolation and crop methods used to resample and
    crop the image
12            if the target size is different from that of the loaded image.
13            Methods are delimited by ":" where first part is interpolation
    and second is crop
14            e.g. "lanczos:random".
15            Supported interpolation methods are "nearest", "bilinear", "
    bicubic", "lanczos",
16            "box", "hamming" By default, "nearest" is used.
17            Supported crop methods are "none", "center", "random".
18
19    # Returns
20        A PIL Image instance.
21
22    # Raises
23        ImportError: if PIL is not available.
24        ValueError: if interpolation method is not supported.
25    """
26
27    # Decode interpolation string. Allowed Crop methods: none, center,
    random
28    interpolation, crop = interpolation.split(":") if ":" in interpolation
    else (interpolation, "none")
29
30    if crop == "none":
31        return keras_preprocessing.image.utils.load_img(path,
32                                                        grayscale=grayscale,
33                                                        color_mode=color_mode,
34                                                        target_size=target_size,
35                                                        interpolation=interpolation)
36
37    # Load original size image using Keras
38    img = keras_preprocessing.image.utils.load_img(path,
39                                                   grayscale=grayscale,
40                                                   color_mode=color_mode,
41                                                   target_size=None,
42                                                   interpolation=interpolation)
43
44    # Crop fraction of total image
45    crop_fraction = 0.875
46    target_width = target_size[1]
```

```
47      target_height = target_size[0]
48
49      if target_size is not None:
50          if img.size != (target_width, target_height):
51
52              if crop not in ["center", "random"]:
53                  raise ValueError('Invalid crop method {} specified.', crop)
54
55              if interpolation not in keras_preprocessing.image.utils.
    _PIL_INTERPOLATION_METHODS:
56                  raise ValueError(
57                      'Invalid interpolation method {} specified. Supported '
58                      'methods are {}'.format(interpolation,
59                          ", ".join(keras_preprocessing.image.utils.
    _PIL_INTERPOLATION_METHODS.keys())))
60
61              resample = keras_preprocessing.image.utils.
    _PIL_INTERPOLATION_METHODS[interpolation]
62
63              width, height = img.size
64
65              # Resize keeping aspect ratio
66              # result shold be no smaller than the targer size, include crop
    fraction overhead
67              target_size_before_crop = (target_width/crop_fraction,
    target_height/crop_fraction)
68              ratio = max(target_size_before_crop[0] / width,
    target_size_before_crop[1] / height)
69              target_size_before_crop_keep_ratio = int(width * ratio), int(
    height * ratio)
70              img = img.resize(target_size_before_crop_keep_ratio, resample=
    resample)
71
72              width, height = img.size
73
74              if crop == "center":
75                  left_corner = int(round(width/2)) - int(round(target_width
    /2))
76                  top_corner = int(round(height/2)) - int(round(target_height
    /2))
77                  return img.crop((left_corner, top_corner, left_corner +
    target_width, top_corner + target_height))
78              elif crop == "random":
79                  left_shift = random.randint(0, int((width - target_width)))
```

```
80                  down_shift = random.randint(0, int((height - target_height))
    )
81                  return img.crop((left_shift, down_shift, target_width +
    left_shift, target_height + down_shift))
82
83      return img
84
85 # Monkey patch
86 keras_preprocessing.image.iterator.load_img = load_and_crop_img
```

Código 4.1: load_and_crop

En Keras existe una limitación importante a la hora de realizar técnicas de Image Augmentation y es que, no existen una función out of the box para el cropping cuando se le pasa un generador como flow_from_directory o flow_from_dataset para ello, existen diversas alternativas creadas por la comunidad, como por ejemplo esta función load_and_crop_img (créditos a rstml). Esta función sobreescribe la función base load_img del paquete keras_preprocessing y le realiza un crop a elegir entre center o random además de tener diversas configuracones para el modo de color y tamaño del cropping que en este caso no vamos a usar, dejaremos por defecto una fracción de corte del 0.875 de la imagen. Este código puede ser encontrado aqui.

```
1 def mount_drive():
2   mounted = drive.mount('/content/drive')
3   return mounted
```

Código 4.2: mount_drive

Tal y como se comento anteriormente, al trabajar con Google Colab usaremos directamente Google Drive para alojar nuestros Datasets, esta función se encarga de montar nuestro drive para poder acceder a ellos.

```
1 def generate_params():
2   params = {'c_categories': 6,'k_feature_maps':4,'n_epochs':20,'batch_size'
     :32,
3             'data_dir': '/content/drive/My Drive/Datasets/EmotionROI'}
4
5   return params
```

Código 4.3: generate_params

Función auxiliar para generar los diferentes parámetros que usaremos en nuestras ejecuciones.

```
1
2 def load_base_model():
3   tf.get_logger().setLevel(logging.ERROR)
4   input_tensor = Input(shape=(256, 256, 3))
5   base_model = applications.ResNet101(input_tensor=input_tensor,weights='
     imagenet',include_top=False)
6
7   for layer in base_model.layers:
8     layer.trainable=False
9
10  return base_model
```

Código 4.4: load_base_model

Esta función descarga el modelo preentrenado ResNet101, además marcamos las capas como no entrenables.

```
1 def generate_model(base_model, params):
2   c_categories = params.get('c_categories')
3   k_feature_maps = params.get('k_feature_maps')
4   x = base_model.output
5   x = Conv2D(c_categories*k_feature_maps, kernel_size=(1,1),activation="relu
     ", padding="same", use_bias=True)(x)
6   x = GlobalAveragePooling2D()(x)
7   x = Dense(c_categories, activation='softmax')(x)
8
9 return Model(base_model.input, x)
```

Código 4.5: generate_model

Pasando como entrada nuestros parámetros y nuestro modelo base, la función generate_model termina de añadirle las capas que faltan a la salida del modelo. Primero se recoge la salida del modelo, se pasa por una capa Conv2D para que pueda ser tratado por la capa GAP y más tarde la salida a través de una capa Dense con activación por softmax y de tamaño igual al número de categorías a etiquetar.

```
1 def prepare_dataset(params):
2
3   batch_size = params.get('batch_size')
4   data_dir = params.get('data_dir')
5
6   num_train_samples = sum([len(files) for r, d, files in os.walk(data_dir+'/
     train')])
```

```
7   num_valid_samples = sum([len(files) for r, d, files in os.walk(data_dir+'/
     val')])
8
9   num_train_steps = math.floor(num_train_samples/batch_size)
10  num_valid_steps = math.floor(num_valid_samples/batch_size)
11
12  train_datagen = ImageDataGenerator(
13        horizontal_flip=True,
14        rotation_range=20,
15        preprocessing_function=preprocess_input)
16
17  val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
18  test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
19
20  train_it = train_datagen.flow_from_directory(data_dir+'/train', class_mode
     ='categorical',shuffle=True, batch_size=batch_size, interpolation = '
     lanczos:random')
21  val_it = val_datagen.flow_from_directory(data_dir+'/val', class_mode='
     categorical', shuffle=False, batch_size=batch_size, interpolation = '
     lanczos:center')
22  test_it = test_datagen.flow_from_directory(data_dir+"/test",class_mode='
     categorical',shuffle=False,batch_size=1, interpolation = 'lanczos:center'
     )
23
24  return train_it,val_it,test_it
```

Código 4.6: prepare_dataset

Uno de los puntos más importantes al entrenar una red con imágenes es el de preparación de las mismas, nuestro dataset se organiza en Drive en tres carpetas diferentes /Train, /Val y /Test con una división del 80 %, 15 % y 5 % del total de imágenes respectivamente. La función ImageDataGenerator se encarga de definir las funciones de aumentación y deformación pertinentes, además de añadirle preprocess_input que se encargará de realizar el cropping mediante la función explicada anteriormente. Una vez definidos nuestros generadores, definiremos los iteradores con el método flow_from_directory que serán los pasados como parámetros a la función fit .

```
1 def prepare_dataset_twitter(params):
2   data_dir = params.get('data_dir')
3   batch_size = params.get('batch_size')
4
5   images_df = pd.read_csv(data_dir+'/twitter_five_agrees.txt',
     delim_whitespace=True, header=None)
6   images_df.columns = ['filename','label']
```

```
7   images_df.loc[images_df.label == 0, 'label'] = 'Negativo'
8   images_df.loc[images_df.label == 1, 'label'] = 'Positivo'
9
10  _custom_datagen = ImageDataGenerator(
11      horizontal_flip=True,
12      rotation_range=20,
13      preprocessing_function=preprocess_input,
14      validation_split=0.2)
15
16  train_it = _custom_datagen.flow_from_dataframe(directory=data_dir,
     dataframe=images_df,x_col='filename',y_col='label', class_mode='
     categorical',shuffle=True, target_size=(256,256),batch_size=batch_size,
     interpolation = 'lanczos:random', subset='training')
17  val_it = _custom_datagen.flow_from_dataframe(directory=data_dir,dataframe=
     images_df,x_col='filename',y_col='label', class_mode='categorical',
     shuffle=False,target_size=(256,256), batch_size=batch_size, interpolation
      = 'lanczos:center', subset='validation')
18
19  num_train_steps=train_it.n//train_it.batch_size
20  num_valid_steps=val_it.n//val_it.batch_size
21
22  return train_it, val_it
```

Código 4.7: prepare_dataset_twitter

Para trabajar con el dataset de Twitter hay que modificar una serie de parámetros en nuestra función de preparación de las imágenes, esto es por que las etiquetas no se encuentran como nombre de la carpeta sino que están almacenadas en un archivo txt por lo que hay que cruzar los datos y tratarlos como un Dataframe, después se usará flow_from_dataframe.

```
1 def plot_acc_loss(history):
2   plt.plot(history.history['acc'])
3   plt.plot(history.history['val_acc'])
4   plt.title('model accuracy')
5   plt.ylabel('accuracy')
6   plt.xlabel('epoch')
7   plt.legend(['train', 'val'], loc='upper left')
8   plt.show()
9
10  plt.plot(history.history['loss'])
11  plt.plot(history.history['val_loss'])
12  plt.title('model loss')
13  plt.ylabel('loss')
14  plt.xlabel('epoch')
```

```
15   plt.legend(['train', 'val'], loc='upper left')
16   plt.show()
```

Código 4.8: plot_acc_loss

Esta función auxiliar nos dará una gráfica comparativa entre los valores de precisión y pérdida para los conjuntos de entrenamiento y validación durante las diferentes epochs.

```
1  def do_test(test_it, model):
2    num_test_steps=test_it.n//test_it.batch_size
3    test_it.reset()
4    Y_pred= model.predict_generator(test_it, steps=num_test_steps,verbose=1)
5    y_pred = np.argmax(Y_pred, axis=1)
6    print('Confusion Matrix')
7    cm = confusion_matrix(test_it.classes, y_pred)
8    print(cm)
9    print('Classification Report')
10   target_names = list(test_it.class_indices.keys())
11   print(classification_report(test_it.classes, y_pred, target_names=
      target_names))
12
13   labels = target_names
14   fig = plt.figure()
15   ax = fig.add_subplot(111)
16   cax = ax.matshow(cm)
17   plt.title('Confusion matrix')
18   fig.colorbar(cax)
19   ax.set_xticklabels(['']  + labels)
20   ax.set_yticklabels(['']  + labels)
21   plt.xlabel('Predicted')
22   plt.ylabel('True')
23   plt.show()
```

Código 4.9: do_test

Esta función realizará una predicción con el conjunto de test y nos mostrará los resultados obtenidos a través de diferentes informes.

```
1  def do_train(model, params, train_it, val_it):
2    n_epochs = params.get('n_epochs')
3    num_train_steps = params.get('num_train_steps')
4    num_valid_steps = params.get('num_valid_steps')
5
6    sgd = optimizers.SGD(lr=0.001, decay=0.0005, momentum=0.9, nesterov=True)
```

```
7   model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['
     accuracy'])
8   history = model.fit_generator(train_it,steps_per_epoch=num_train_steps,
     epochs=n_epochs,validation_data=val_it,validation_steps=num_valid_steps)
9
10  return history
```

Código 4.10: do_train

Esta función se encarga de compilar el modelo y realizar el entrenamiento.

```
1 if __name__== "__main__":
2   params = generate_params()
3   mount_drive()
4   base_model = load_base_model()
5   model = generate_model(base_model, params)
6
7   print(model.summary())
8
9   train,val,test = prepare_dataset(params)
10  history = do_train(model, params, train, val)
11  plot_acc_loss(history)
12  do_test(test, model)
```

Código 4.11: __main__

Función orquestadora de los diferentes experimentos.

Capítulo 5

Experimentos

Una vez definida la arquitectura, se enumeran los diferentes experimentos que se han realizado con sus diferentes configuraciones. Se analizan los resultados de los experimentos.

5.1. Datasets elegidos

Tras la investigación realizada en el capitulo de estado del arte, se eligieron dos de los datasets más conocidos entre los artículos de análisis de sentimientos, por un lado tenemos EmotionROI y por otro Twitter 1, pasamos a comentar sus características:

- EmotionROI: Cuenta con 1,980 imágenes distribuidas entre 6 categorías diferentes: anger, disgust, fear, joy, sadness y surprise.
- Twitter 1: En este caso son 1,269 imágenes clasificadas con solo 2 etiquetas, positivo y negativo.

5.2. Hiperparámetros

Los parámetros que tendremos en cuenta para realizar los diferentes experimentos son:

- batch_size: Número de imágenes que se propagarán a la vez en la red .En nuestros experimentos usaremos diferentes tamaños 16,32 y 64.
- learning_rate: Ratio de aprendizaje de nuestro modelo, elegir un valor demasiado pequeño conllevará un aumento en el tiempo de procesamiento pero puede conseguir mejores resultados, nosotros usaremos 0,001 y 0,0001.
- n_epochs: Número de ciclos de entrenamiento. Usaremos 20, 40 y 60.

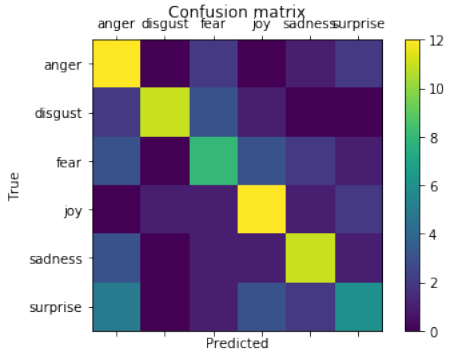
5.3. Resumen de los experimentos

5.3.1. Experimento 1

Dataset	batch	leaning_rate	epochs
EmotionROI	32	0.001	20

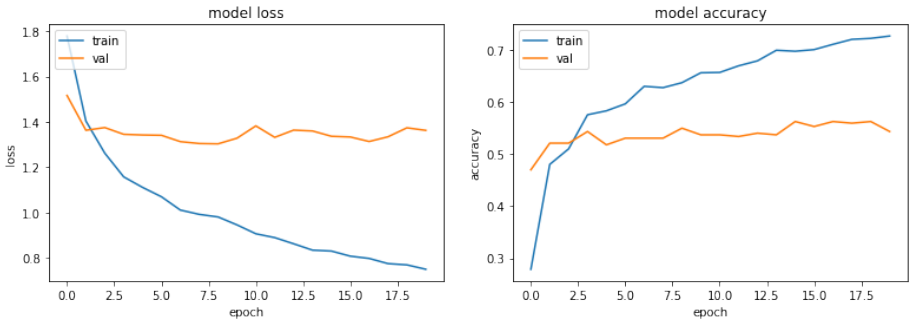
Classification Report

	precision	recall	f1-score	support
anger	0.48	0.71	0.57	17
disgust	0.92	0.65	0.76	17
fear	0.50	0.47	0.48	17
joy	0.60	0.71	0.65	17
sadness	0.65	0.65	0.65	17
surprise	0.50	0.35	0.41	17
accuracy			0.59	102
macro avg	0.61	0.59	0.59	102
weighted avg	0.61	0.59	0.59	102



(a) Figure A (b) Figure B

Figura 5.1: Resultados experimento 1: Resumen / Matriz Confusión



(a) Figure A (b) Figure B

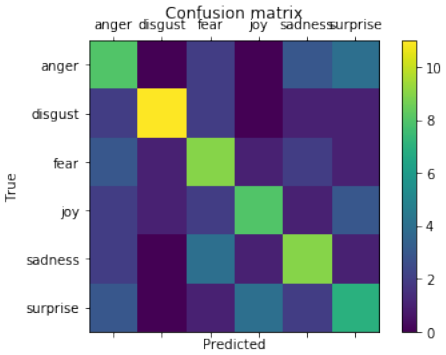
Figura 5.2: Resultados experimento 1: acc / loss

5.3.2. Experimento 2

Dataset	batch	leaning_rate	epochs
EmotionROI	64	0.001	20

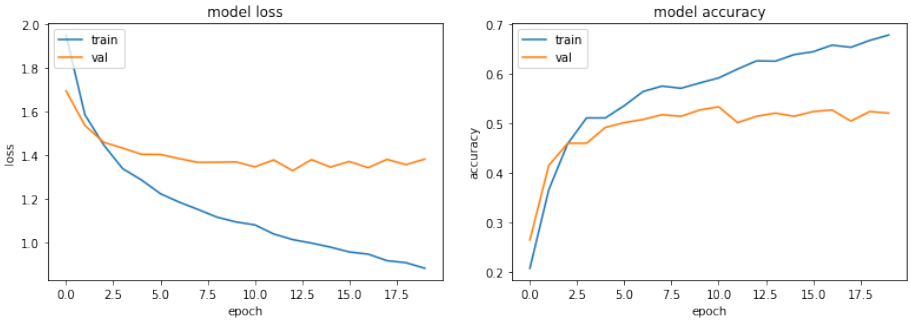
Classification Report

	precision	recall	f1-score	support
anger	0.40	0.47	0.43	17
disgust	0.85	0.65	0.73	17
fear	0.45	0.53	0.49	17
joy	0.57	0.47	0.52	17
sadness	0.50	0.53	0.51	17
surprise	0.41	0.41	0.41	17
accuracy			0.51	102
macro avg	0.53	0.51	0.52	102
weighted avg	0.53	0.51	0.52	102



(a) Figure A (b) Figure B

Figura 5.3: Resultados experimento 2: Resumen / Matriz Confusión



(a) Figure A (b) Figure B

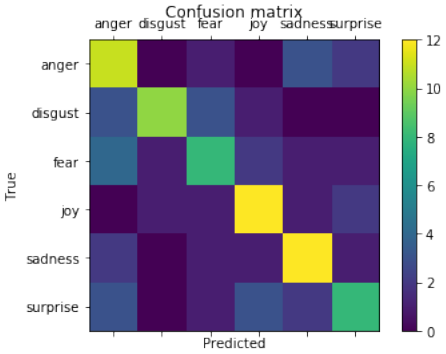
Figura 5.4: Resultados experimento 2: acc / loss

5.3.3. Experimento 3

Dataset	batch	leaning_rate	epochs
EmotionROI	16	0.001	20

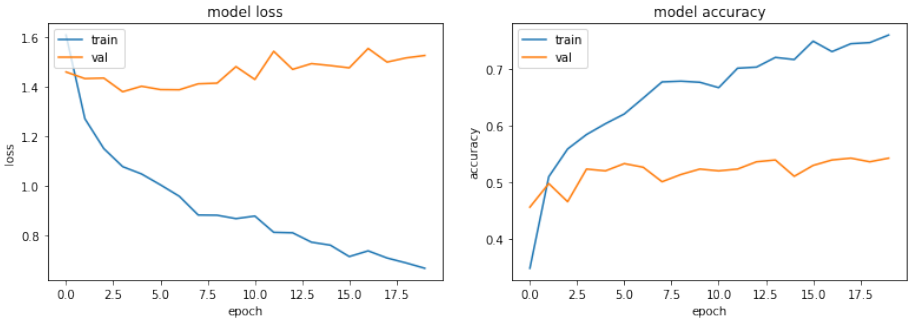
Classification Report

	precision	recall	f1-score	support
anger	0.48	0.65	0.55	17
disgust	0.83	0.59	0.69	17
fear	0.53	0.47	0.50	17
joy	0.63	0.71	0.67	17
sadness	0.63	0.71	0.67	17
surprise	0.57	0.47	0.52	17
accuracy			0.60	102
macro avg	0.61	0.60	0.60	102
weighted avg	0.61	0.60	0.60	102



(a) Figure A (b) Figure B

Figura 5.5: Resultados experimento 3: Resumen / Matriz Confusión



(a) Figure A (b) Figure B

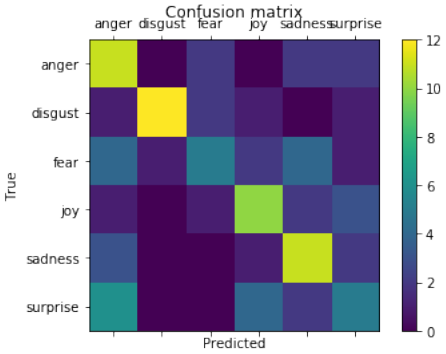
Figura 5.6: Resultados experimento 3: acc / loss

5.3.4. Experimento 4

Dataset	batch	leaning_rate	epochs
EmotionROI	32	0.001	40

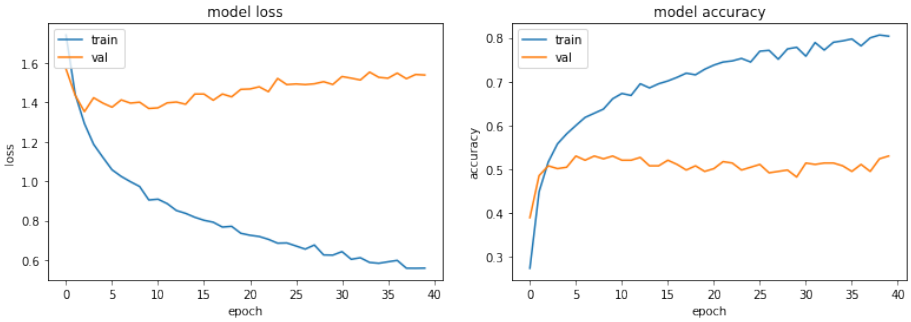
Classification Report

	precision	recall	f1-score	support
anger	0.42	0.65	0.51	17
disgust	0.92	0.71	0.80	17
fear	0.50	0.29	0.37	17
joy	0.56	0.59	0.57	17
sadness	0.52	0.65	0.58	17
surprise	0.36	0.29	0.32	17
accuracy			0.53	102
macro avg	0.55	0.53	0.53	102
weighted avg	0.55	0.53	0.53	102



(a) Figure A (b) Figure B

Figura 5.7: Resultados experimento 4: Resumen / Matriz Confusión



(a) Figure A (b) Figure B

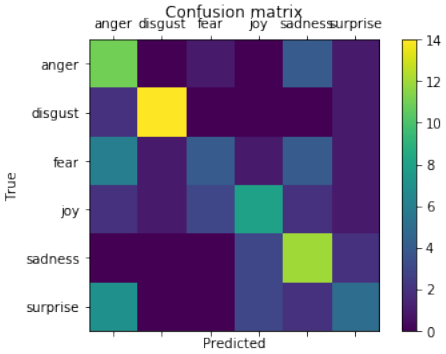
Figura 5.8: Resultados experimento 4: acc / loss

5.3.5. Experimento 5

Dataset	batch	leaning_rate	epochs
EmotionROI	32	0.0001	20

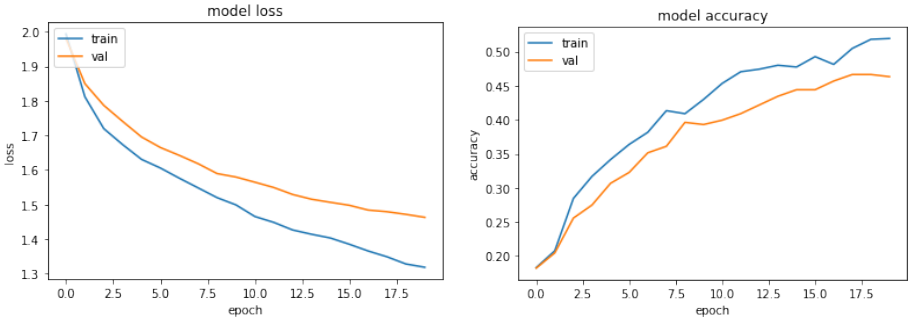
Classification Report

	precision	recall	f1-score	support
anger	0.39	0.65	0.49	17
disgust	0.88	0.82	0.85	17
fear	0.50	0.24	0.32	17
joy	0.53	0.47	0.50	17
sadness	0.50	0.71	0.59	17
surprise	0.45	0.29	0.36	17
accuracy			0.53	102
macro avg	0.54	0.53	0.52	102
weighted avg	0.54	0.53	0.52	102



(a) Figure A (b) Figure B

Figura 5.9: Resultados experimento 5: Resumen / Matriz Confusión



(a) Figure A (b) Figure B

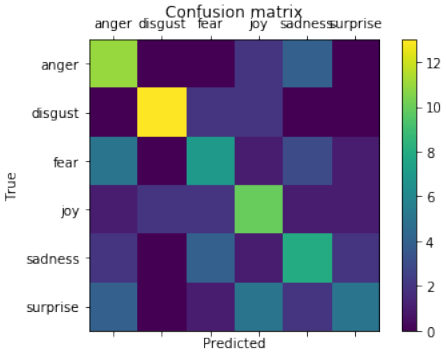
Figura 5.10: Resultados experimento 5: acc / loss

5.3.6. Experimento 6

Dataset	batch	leaning_rate	epochs
EmotionROI	32	0.0001	60

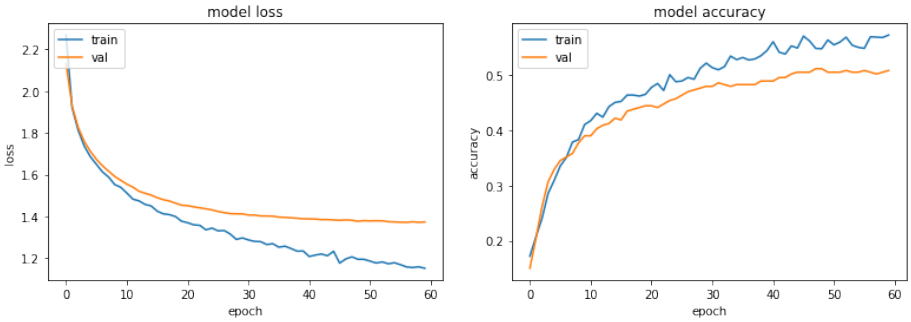
Classification Report

	precision	recall	f1-score	support
anger	0.48	0.65	0.55	17
disgust	0.87	0.76	0.81	17
fear	0.44	0.41	0.42	17
joy	0.48	0.59	0.53	17
sadness	0.44	0.47	0.46	17
surprise	0.56	0.29	0.38	17
accuracy			0.53	102
macro avg	0.54	0.53	0.53	102
weighted avg	0.54	0.53	0.53	102



(a) Figure A (b) Figure B

Figura 5.11: Resultados experimento 6: Resumen / Matriz Confusión

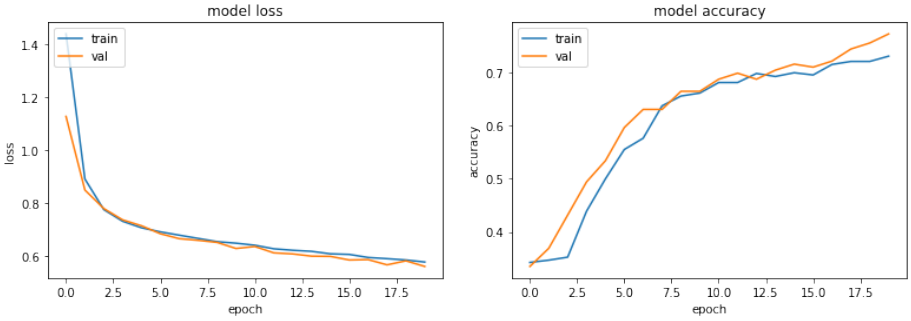


(a) Figure A (b) Figure B

Figura 5.12: Resultados experimento 6: acc / loss

5.3.7. Experimento 7

Dataset	batch	leaning_rate	epochs
Twitter	32	0.0001	20

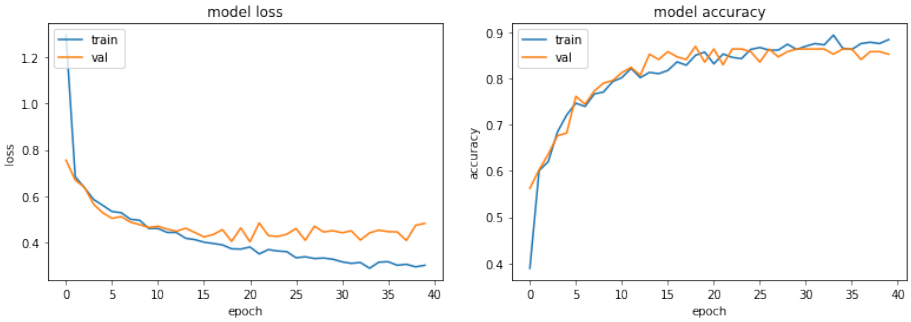


(a) Figure A (b) Figure B

Figura 5.13: Resultados experimento 7: acc / loss

5.3.8. Experimento 8

Dataset	batch	leaning_rate	epochs
Twitter	32	0.0001	40

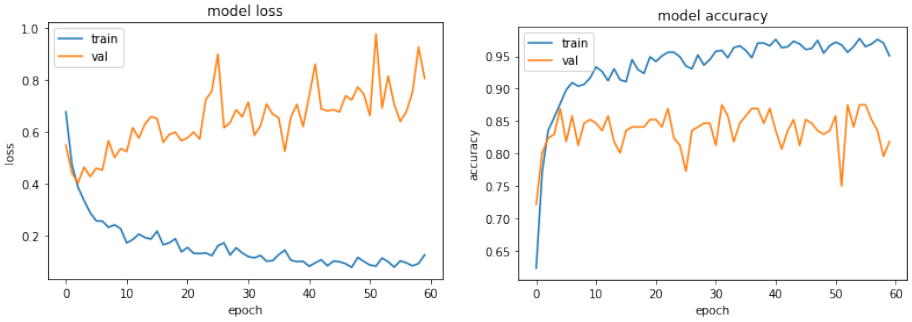


(a) Figure A (b) Figure B

Figura 5.14: Resultados experimento 8: acc / loss

5.3.9. Experimento 9

Dataset	batch	leaning_rate	epochs
Twitter	32	0.001	60



(a) Figure A (b) Figure B

Figura 5.15: Resultados experimento 9: acc / loss

5.4. Resultados

Tras los diferentes experimentos realizados con nuestra red convolucional sobre los dos datasets se deduce lo siguiente:

- EmotionROI: En este primer caso se alcanzó la máxima precisión de 54 % en el experimento número 1, con un batch size de 32, learning rate de 0,001 y 20 epochs. Todos los experimentos han dado un resultado cercano, la diferencia notable es que con un learning rate de 0,0001 se ha conseguido un overfitting inferior con unos resultados entre los conjuntos de entrenamiento y validación parejos. Las matrices de confusión nos dejan ver que los sentimientos positivos (joy, surprise) y los negativos (sadness, anger, disgust, fear) son más complicados de diferenciar entre ellos por nuestra red convolucional, ya que las imágenes tienen características similares.

- Twitter 1: En este caso, teniendo un dataset donde solo 2 etiquetas son valoradas, positivo y negativo, nuestra red consigue una precisión del 81 % en el experimento número 8 con un batch size de 32, learning rate de 0,0001 y 40 epochs. El tener solo dos etiquetas a beneficiado a encontrar un porcentaje de precisión mayor al tener las imágenes características tan diferentes entre ellas.

Capítulo 6

Conclusiones

6.1. Repaso de lo aprendido

Antes de elegir proyecto de final de Máster solo sabia que quería que estuviese relacionado con deep learning, durante el Máster aprendí las nociones básicas para poder empezar casi cualquier proyecto o al menos poder buscarme las herramientas correctas, pero sobre todo descubrí una ventana a este mundo tan extenso y sorprendente. La elección de la temática fue en gran medida por la curiosidad, quería sentirme capaz de poder lograr analizar algo tan humano como son los sentimientos pero a través de imágenes, dotar de la inteligencia suficiente a una maquina para que pudiera contar en la medida de lo posible lo que le transmitía.

Durante este camino he aprendido mucho sobre las redes convolucionales y su funcionamiento, además de las diferentes tipos de capas usados en el experimento. He tenido que leer y releer muchísimos artículos científicos desde que se empezará a hablar del concepto allá por 2013 hasta la actualidad, realizando una tarea de organización documental y estructuración del ciclo de vida de la idea bastante extensa hasta lograr comprender lo máximo posible la situación en la que se encuentran las investigaciones actuales.

Sin duda ha sido un camino apasionante, aunque frustrante a veces, no he tenido quizás las herramientas adecuadas para poder haber sacado más partido a los conocimientos y sobre todo no he tenido el tiempo necesario para poder haber logrado avanzar más, aun con eso, me quedo con haber desarrollado un modelo basado en la tecnología aprendida en el Máster Keras, a pesar de que la mayoría de los últimos trabajos de esta materia han usado un framework a más bajo nivel cómo es PyTorch.

A mis espaldas queda una herramienta perfectamente ampliable que espero que pueda servir a más de uno a comprender el estado actual del análisis de sentimientos en imágenes y que por su puesto sirva también para que más personas se animen a participar en este gran proyecto de investigación.

6.2. Trabajo futuro

Es evidente que aun queda mucho trabajo por hacer, el poder diferenciar entre dos sentimientos (positivo y negativo) no supone ningún reto con las herramientas actuales, sin embargo, el poder predecir sentimientos más elaborados como los que se han utilizado en nuestros experimentos con el dataset EmotionROI si presenta un verdadero reto, no solo desde el punto de vista de la red o el modelo que sea capaz de predecirlos sino también de su previo etiquetado, y es que, no hay nada tan especial en los sentimientos que los ojos con los que se ven.

Bibliografía

[1] Philip Beineke, Trevor Hastie, and Shivakumar Vaithyanathan. The sentimental factor: Improving review classification via human-provided information. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 263–270, Barcelona, Spain, July 2004.

[2] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 223–232, New York, NY, USA, 2013. Association for Computing Machinery.

[3] Victor Campos, Brendan Jou, and Xavier Gir'o i Nieto. From pixels to sentiment: Fine-tuning cnns for visual sentiment prediction. *CoRR*, abs/1604.03489, 2016.

[4] Victor Campos, Amaia Salvador, Brendan Jou, and Xavier Gir'o i Nieto. Diving deep into sentiment: Understanding fine-tuned cnns for visual sentiment prediction. *CoRR*, abs/1508.05056, 2015.

[5] Tao Chen, Damian Borth, Trevor Darrell, and Shih-Fu Chang. Deepsentibank: Visual sentiment concept classification with deep convolutional neural networks. *CoRR*, abs/1410.8586, 2014.

[6] Tao Chen, Felix X. Yu, Jiawei Chen, Yin Cui, Yan-Ying Chen, and Shih-Fu Chang. Object-based visual sentiment concept analysis and application. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 367–376, New York, NY, USA, 2014. Association for Computing Machinery.

[7] H. Kim, Y. Kim, S. J. Kim, and I. Lee. Building emotional machines: Recognizing image emotions through deep neural networks. *IEEE Transactions on Multimedia*, 20(11):2980–2992, Nov 2018.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference*

on Neural Information Processing Systems - Volume 1, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[9] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[10] Guoqin Peng and Dan Xu. *Weakly Supervised Learning of Image Emotion Analysis Based on Cross-spatial Pooling*, pages 116–125. 10 2019.

[11] D. She, J. Yang, M. Cheng, Y. Lai, P. L. Rosin, and L. Wang. Wscnet: Weakly supervised coupled networks for visual sentiment classification and detection. *IEEE Transactions on Multimedia*, pages 1–1, 2019.

[12] Jingwen Wang, Jianlong Fu, Yong Xu, and Tao Mei. Beyond object recognition: Visual sentiment analysis with deep coupled adjective and noun neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 3484–3490. AAAI Press, 2016.

[13] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[14] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. Robust image sentiment analysis using progressively trained and domain transferred deep networks. *CoRR*, abs/1509.06041, 2015.