Trabajo final RNA

Victoria Beltrán Domínguez

Enero 2021

Contents

1 Introducción

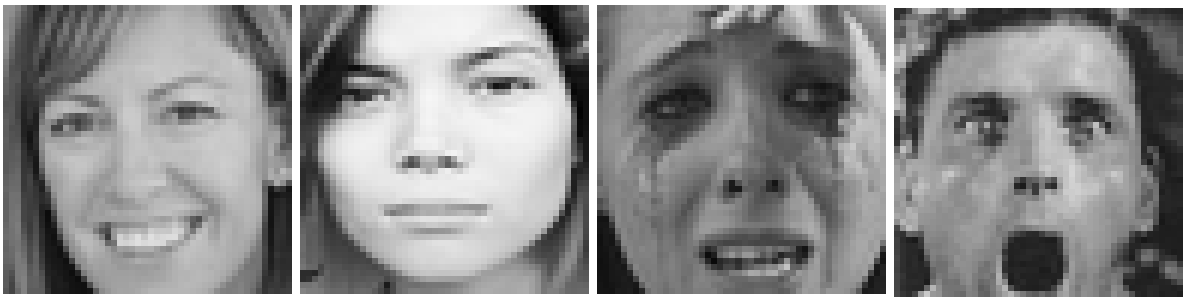
Este trabajo se plantea experimentar con distintos modelos para el reconocimiento de emociones faciales en imágenes a partir del conjunto de datos encontrado en Kaggle: Emotion Detection.

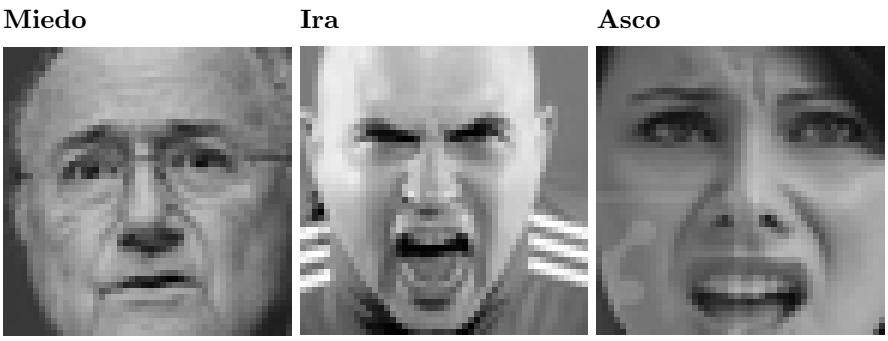
Para ello, lo primero que se hará será detallar los datos disponibles. A continuación, se destacarán algunos resultados vistos en la propia herramienta de Kaggle para tomarlos como referencia y, finalmente, se comentará de qué manera se ha atacado el problema, qué experimentos se han realizado y que resultados se han obtenido, comparándonos con aquellos de referencia.

2 Datos disponibles

Para llevar a cabo esta tarea disponemos de 28709 imágenes para entrenamiento y 7178 para validación. Estas imágenes son fotos de caras en blanco y negro de 48x48 píxeles, se dan subdivididas en train y test, y categorizadas en función de la emoción que muestra la expresión facial. Distinguimos siete clases:

Felicidad **Neutral** **Tristeza** **Sorpresa**





3 Benchmark

Se han encontrado diferentes cuadernos en Kaggle con enfoques interesantes. Entre ellos, pasamos a destacar algunos junto a sus resultados para poderlos comparar con el que se construirá:

- Emotion Detector: en este cuaderno se utilizan redes neuronales convolucionales para conseguir un 65.22% de precisión. Algo que llama la atención es que están importados los modelos preentrenados VGG16 y InceptionResNetV2. Sin embargo, no se hace uso de ellos, lo que nos da a entender que quizás dan peores resultados.
- Emotion Detection: en este cuaderno se utiliza una DenseNer169 como extractor de características de las imágenes, que combinado con unas cuantas capas densas con dropout, obtiene un resultado de 64.25% en validación.
- CNN for emotion detection: En este otro cuaderno, se consigue una precisión de 61.67% nuevamente a partir de CNN.
- Facial-Emotion-Recognition: Otra arquitectura utilizando CNNs que consigue un 61.36% de precisión.

4 Resumen experimentos

En caso de no decir lo contrario, se ha utilizado Adam con ReduceLROnPlateau(0.001-0.0000001) y 50 epochs, además data augmentation con:

- rotation_range=30
- shear_range=0.2
- zoom_range=0.2
- horizontal_flip= True
- rescale=1./255

Estos son los experimentos realizados para esta tarea:

Descripción red	**Accuracy**
Red especificada en Figura 1	0.6779
Imagenes cargadas directamente en rgb con Resnet152v2 + GlAvgPool + Drop(0.3) + Dense(512) + BN + Drop(0.2) + Dense(256) + BN + Drop(0.3) + Dense(7)	0.385
Convolución inicial para conseguir 3 dimensiones + Resnet152v2 + GlAvgPool + Drop(0.3) + Dense(512) + BN + Drop(0.2) + Dense(256) + BN + Drop(0.3) + Dense(7)	0.334
Imágenes cargadas directamente en rgb utilizando SGD con ResNet50 + GlAvgPool + Drop(0.3) + Dense(512) + BN + Drop(0.2) + Dense(7)	0.25
Red especificada en Figura 1 sustituyendo los MaxPool por convoluciones con stride=2	0.678
Red especificada en Figura 1 con cut-off (8x8)	0.676
Red especificada en Figura 1 con mixup y sustituyendo los MaxPool por convoluciones con stride=2	**0.685**

5 Descripción experimentación realizada

Lo primero que se ha hecho una vez se cargados los datos, ha sido adaptar la red construida en CIFAR a este problema. Así, la primera red con la que se ha experimentado ha sido la que se muestra en la Figura 1. Esta se ha probado además con técnicas básicas de *Data augmentation*. Inicializando esta red con *Adam* a 0.001, con *ReduceLROnPlateu* y con 50 epochs conseguimos un precisión de 0.6779.

Para intentar mejorar ese resultado, se ha probado a sustituir el MaxPool por una convolución con stride=2 para comprobar si la representación de la imagen es así más consistente. Obtenemos un val_accuracy de 0.678, que aunque consigue mejorar, la mejora es prácticamente imperceptible.

A continuación, se procede a aplicar la técnica de cut-off en el dataset que tenemos. Se ha extraído un cuadrado 8x8 aleatorio de cada imagen de entrenamiento. Las imágenes quedan como las que podemos ver a continuación.

Con esta técnica, se obtiene una precisión de 0.676. Vemos que todos están obteniendo resultados muy similares. También vamos a probar con mix-up. Mezclamos imágenes de tal forma que podamos ver 70% de una imagen y 30% de la otra, tal y como podemos ver en las siguientes imágenes.

Este técnica obtiene un resultado de 0.685, el mejor resultado visto hasta este momento.

Finalmente, también se ha querido probar con modelos preentrenados. El problema es que las imágenes están en blanco y negro, por lo que se nos plantean dos opciones: o bien las cargamos en rgb por el generador o bien añadimos una convolución inicial que ponga las dimensiones a 3. Para probar cuál de las dos técnicas funciona mejor, se ha probado con el modelo preentrenado de ResNet152V2, combinándolo con algunas capas densas más. Cargando directamente las imágenes a rgb, hemos obtenido una precisión de 0.385, mientras que añadiendo una convolución inicial para generar esos tres canales ha conseguido una precisión de 0.33, por lo que para las siguientes pruebas, vamos a cargarlas directamente en rgb por el generador. Sin embargo, después de probar también con el modelo preentrenado ResNet50 y viendo que este no puede pasar de una precisión de 0.25, se decide dejar de experimentar con modelos preentrenados, pues probablemente el convertir el color está empeorando nuestros resultados.

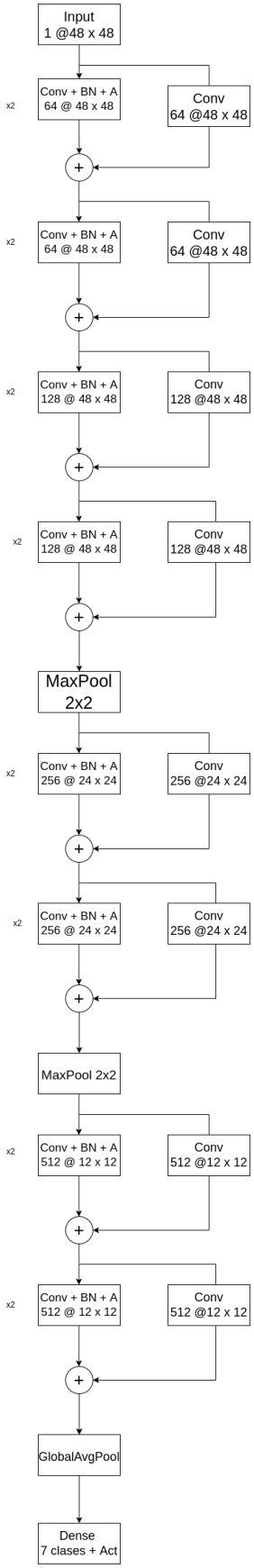


Figure 1: Arquitecture used for our first model

6 Conlusiones

Se han realizado muchos experimentos. Se ha probado con diferentes técnicas aprendidas y también con modelos preentrenados. Se ha probado cut-off de cuadrados de 8x8, obteniendo resultados muy parecidos a los que teníamos. También hemos probado mixup, que ha conseguido el mejor resultado junto con una sustitución del maxPooling por una convolución con stride=2. También se han probado modelos preentrenados, pero al ser imágenes en gris y tratarlas como imágenes en color, no se consigue realmente aprender sus características. A pesar del fracaso utilizando modelos preentrenados, hemos conseguido mejorar nuestra primera prueba y superar el punto de referencia establecido en los cuadernos de Kaggle.