Informe de Trabajo Práctico Final
Neuroscience of Learning Machines

Ayrton Tomás Valdes - atomasvaldes@gmail.com

Facultad de Ciencias Exactas - Universidad Nacional del Centro de la Provincia de Buenos Aires

Introducción

En este trabajo práctico se abordaron distintas técnicas y estructuras con el objetivo de obtener los mejores resultados posibles en la detección de emociones a través de redes neuronales. Se utilizó el set de datos Facial Expression Recognition 2013[1], la cual consiste en una serie de imágenes de caras de 48x48 píxeles en escala de grises categorizadas en siete distintas emociones: enojo, disgusto, felicidad, tristeza, sorpresa, neutro.

El informe se divide en tres partes y una breve conclusión.

Dataset: Donde se explica el conjunto de datos utilizado para el entrenamiento y testeo de las redes neuronales, a la vez que distintas problemáticas que esta presenta.

Redes neuronales: Donde se explican algunas de las distintas redes neuronales implementadas para la resolución de este problema hasta llegar a la de mejor performance obtenida.

Mejoras: Donde se explican distintas técnicas que se probaron para mejorar la performance de la red neuronal con mejor desempeño obtenido y los resultados de estas.

Dataset:

FER-2013 consiste en una serie de imágenes de caras de 48x48 píxeles en escala de grises categorizadas en siete distintas emociones: enojo, disgusto, felicidad, tristeza, sorpresa y neutro.

Algunos ejemplos de estas imágenes son las siguientes.

Este set de datos posee la siguiente distribución de imágenes a lo largo de sus categorías.

Imágenes utilizadas en la etapa de training.

- 4965 Imágenes en Neutral.
- 4097 Imágenes en Miedo.
- 3995 Imágenes en Enojado.
- 436 Imágenes en Disgusto.
- 3171 Imágenes en Sorpresa.
- 7215 Imágenes en Feliz.
- 1247 Imágenes en Tristeza

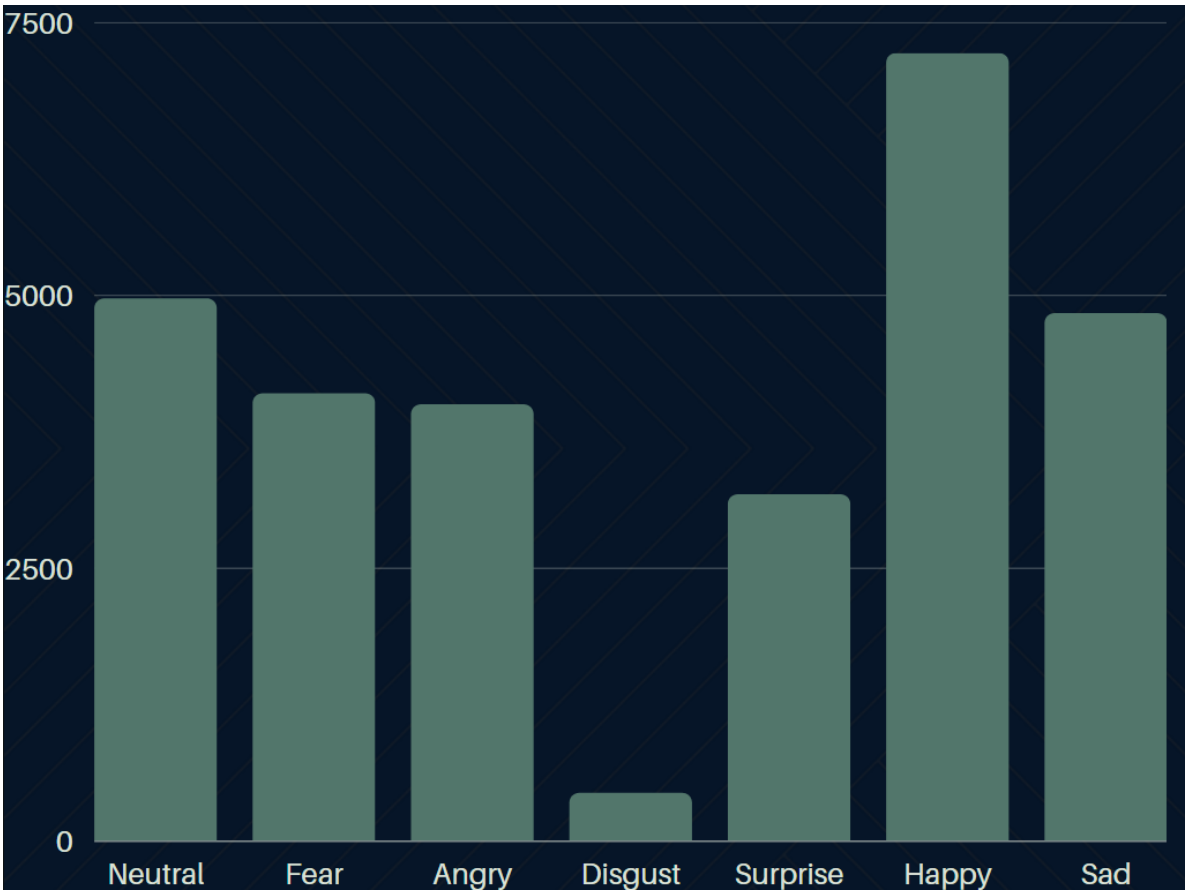

Gráfico 1. Distribución del dataset para training.

Imágenes utilizadas en la etapa de testing.

- 1233 Imágenes en Neutral.
- 1024 Imágenes en Miedo.
- 958 Imágenes en Enojado.
- 111 Imágenes en Disgusto.
- 831 Imágenes en Sorpresa.
- 1774 Imágenes en Feliz.
- 1247 Imágenes en Tristeza

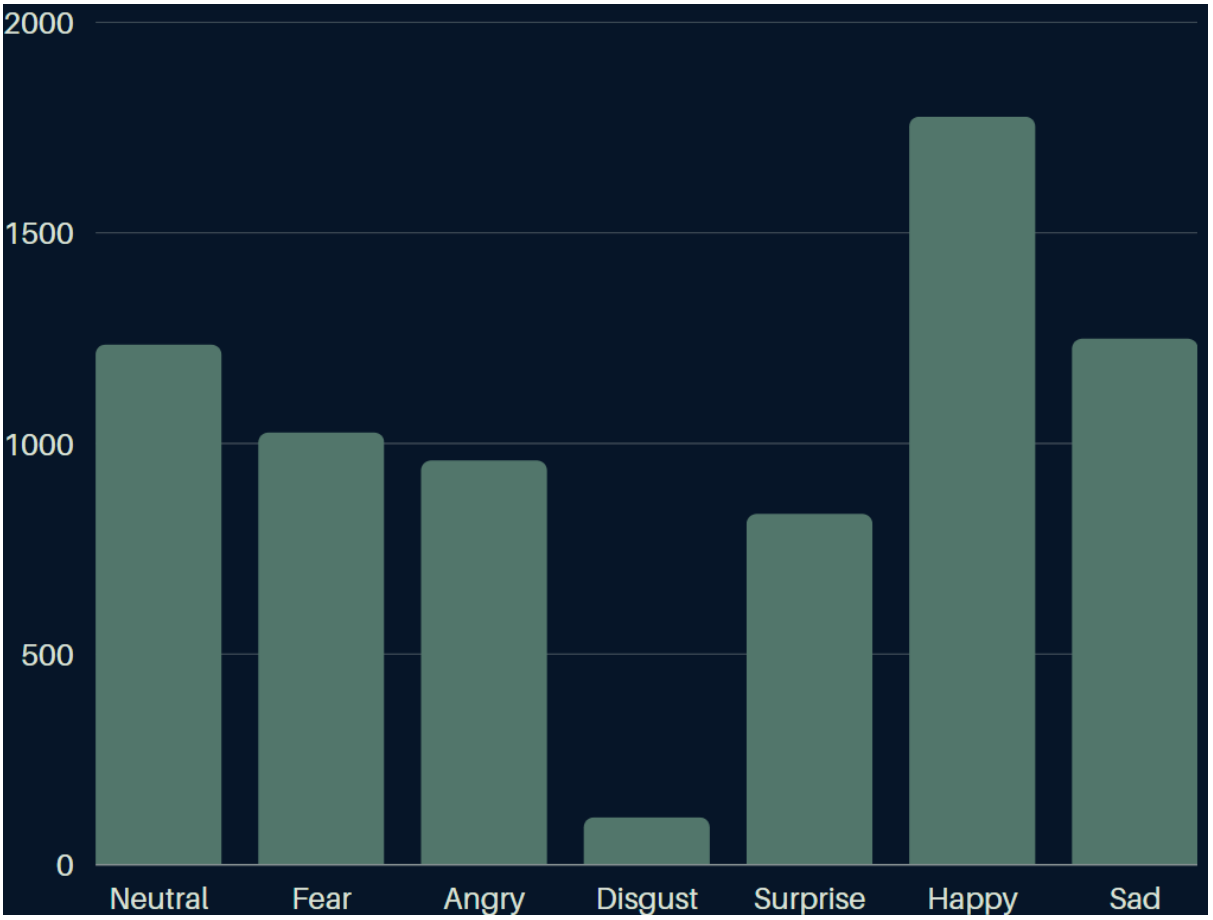

Gráfico 2. Distribución del dataset para testing.

El set de datos no es demasiado extenso, lo cual puede reducir la eficacia del entrenamiento de la red neuronal. Para resolver esto, se utilizó data augmentation para incrementar la cantidad de imágenes utilizadas en la etapa de training.
Otro problema presente en este conjunto de datos, es la inclusión por defecto de algunas imágenes corruptas en las distintas categorías. Algunos ejemplos de estas son los siguientes.

Si bien no son demasiadas las imágenes corruptas presentes, estas pueden causar problemas en la red neuronal, por lo cual se eliminaron del set de datos.

Por último, como se puede ver en los gráficos presentados, el set de datos es muy desbalanceado, lo cual puede causar problemas a la hora de obtener información relevante de las categorías con menor cantidad de datos disponible.

Redes Neuronales:

Se consideraron probar distintos tipos de redes neuronales para obtener el mejor rendimiento posible.
Como primer intento se probó una red densa multicapa, ya que el poco tamaño que poseen las imágenes del set de datos facilita la utilización de este tipo de redes completamente conectadas.

Para la mayoría de las redes, se aplicaron los siguientes aumentos de datos utilizando el preprocesamiento de ImageGenerator de Keras.

- Reescalado a 1./255, para facilitar el manejo de los datos de las imágenes.
- Horizontal flip
- Rotation range en 5
- width shift range en 0.25
- height shift range en 0.25
- Validation split en 0.2 para utilizar 20% de los datos en testing, debido a la poca cantidad de data para testing que hay presente en el conjunto de datos.

1. Red únicamente con capas densas. Se probó el conjunto de datos utilizando las siguientes capas

Layer (type)	Shape	Param #
flatten (Flatten)	(None)	0
dense_1 (Dense)	(64)	147520
○ Activación *RELU*		
dense_2 (Dense)	(128)	8320
○ Activación *RELU*		
dense_3 (Dense)	(7)	903
○ Activación *SOFTMAX*		

Total params: 156,743
Trainable params: 156,743
Non-trainable params: 0
Funcion de perdida: ‘categorical cross entropy’
Optimizador: Adam
Batch: 64
Epochs: 15

Obteniéndose los siguientes resultados:

Val_accuracy: 0.36848703026771545

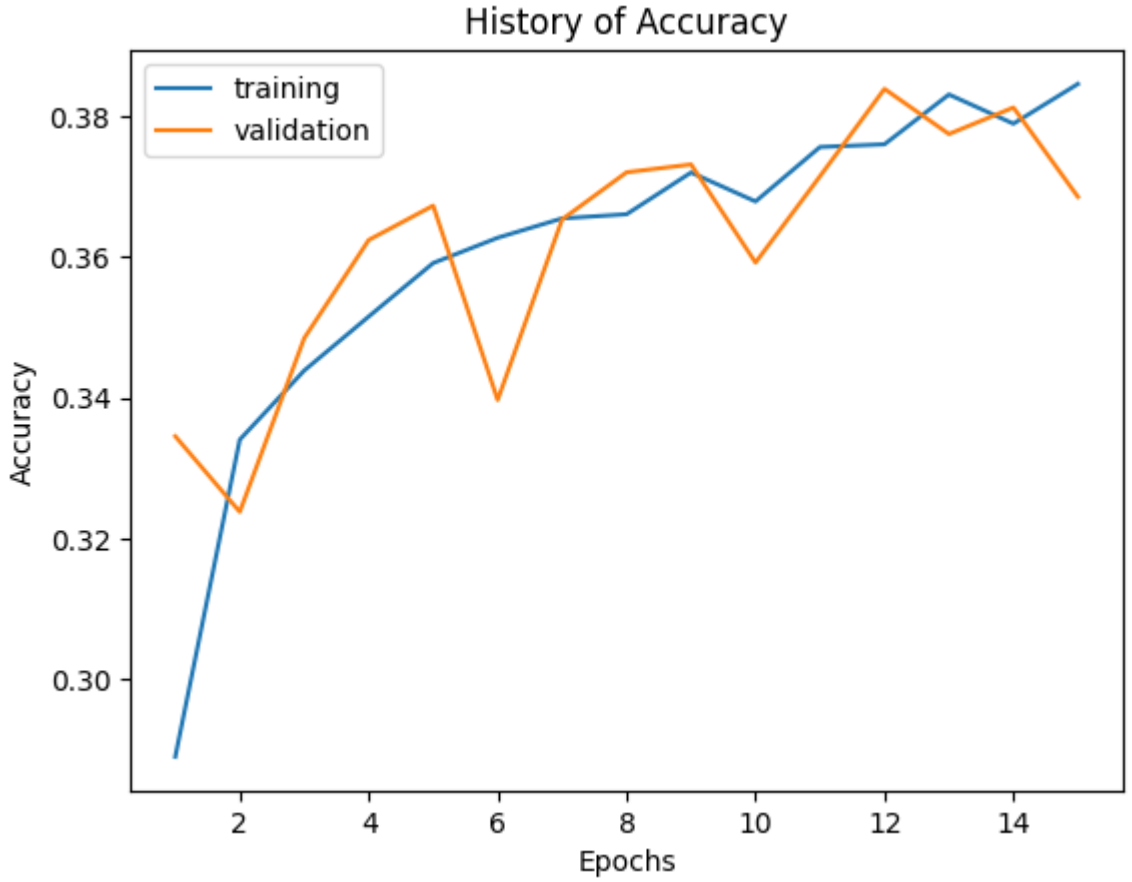

Gráfico 3. Resultados de red con capas densas.

Posteriormente, se combinó con redes convolucionales 2D, debido a la complejidad de los datos a analizar. Las imágenes en el conjunto de datos tienden a ser muy variadas y no estar del todo centradas, por lo que la posición de los datos en cada imagen afectará el desempeño de las capas densas, por esto, se utilizaron capas convolucionales para obtener características.

2. *Red utilizando tres capas convolucionales.*

Layer (type)	Shape	Param #
==========	==========	==========
• conv2d_1 (Conv2D 32, (5,5)) ○ Activación *RELU*	(44, 44, 32)	832
• pooling2d_1 (MaxPooling2D (2,2))	(22, 22, 32)	0

- conv2d_2 (Conv2D 32, (3,3)) (20, 20, 32) 9248
    - Activación *RELU*
- pooling2d_2 (MaxPooling2D (2,2)) (10, 10, 32) 0
- conv2d_3 (Conv2D 32, (5,5)) (6, 6, 32) 25632
    - Activación *RELU*
- pooling2d_3 (MaxPooling2D (2,2)) (3, 3, 32) 0
- flatten (Flatten) (288) 0
- dense_1 (Dense) (64) 18496
    - Activación *RELU*
- dense_2 (Dense) (128) 8320
    - Activación *RELU*
- dense_3 (Dense) (7) 903
    - Activación *SOFTMAX*

==========

Total params: 63,431
Trainable params: 63,431
Non-trainable params: 0
Epochs: 15
Batch: 64
Funcion de perdida: ‘categorical cross entropy’
Optimizador: Adam

Se implementaron tres capas de redes convolucionales de 32 features junto a las capas densas del intento anterior. Se agregaron, también, capas de max pooling de 2x2 después de cada capa convolucional, para reducir la muestra resultante.
Se decidió el no utilizar, en este intento, capas de dropout y batch normalization para poder comparar mejor el resultado del desempeño de este tipo de red con el anterior.

Los resultados obtenidos fueron los siguientes:

Val_accuracy: 0.531206488609314

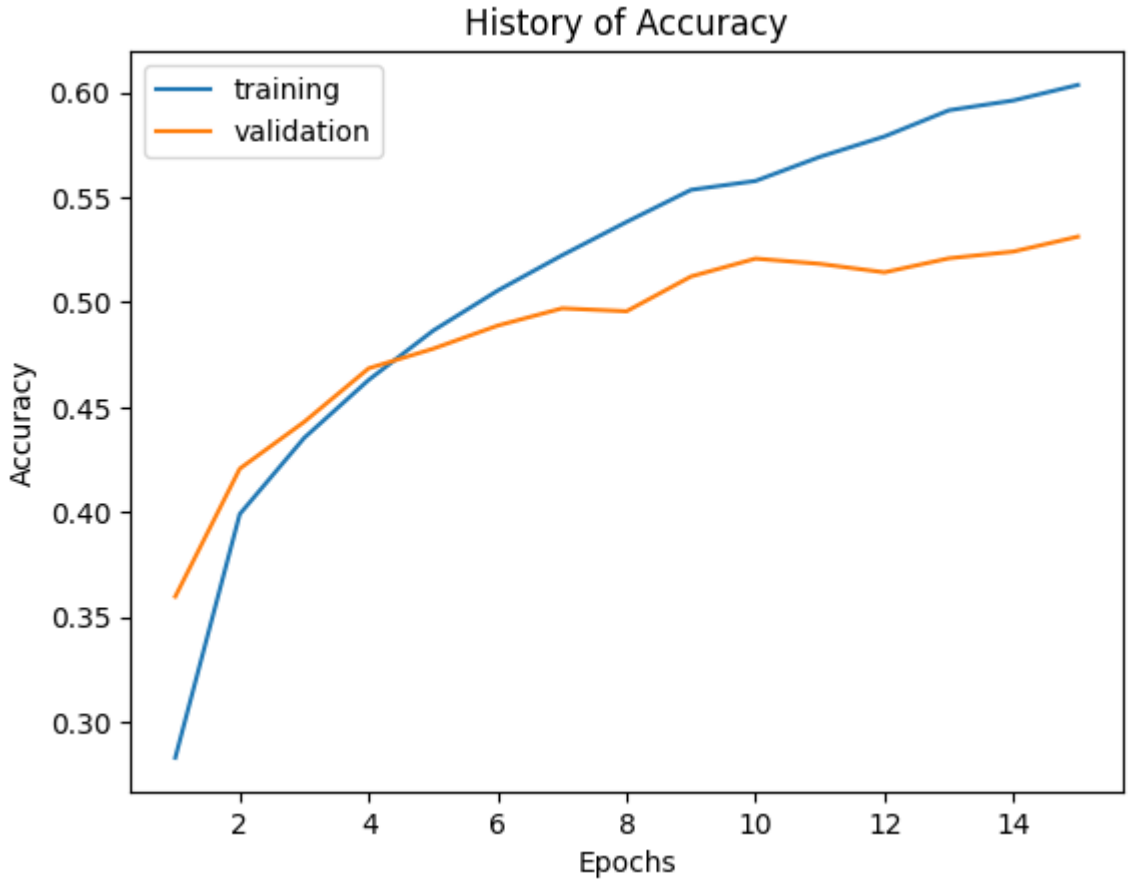

Gráfico 4. Resultados de red con capas convolucionales.

Como se esperaba, hubo una buena mejoría al agregar algunas capas de redes convolucionales.

Se probaron distintas topologías de red más, obteniendo algunas mejoras mínimas al aumentar el número de neuronas utilizadas en las capas densas y el número de features en las capas convolucionales, reduciendo el número de imágenes por batch o cambiando el tipo de optimizador usado. El mejor resultado se obtuvo al utilizar la siguiente red.

3. *Red con mejor desempeño obtenido.*

Layer (type)	Shape	Param #

- conv2d (Conv2D 64 (3,3)) (48, 48, 128) 1280
  - Activación *RELU*
  - Strides: 1
  - Padding: *same*

- batch_normalization_1 (BatchNormalization) (48, 48, 128) 512
- max_pooling2d (MaxPooling2D (2,2)) (24, 24, 128) 0
- dropout (Dropout (0.25)) (24, 24, 128) 0
- conv2d_1 (Conv2D 125 (5,5)) (24, 24, 256) 295168
    - Activación: *RELU*
    - Strides: 1
    - Padding: *same*
- batch_normalization_2 (BatchNormalization) (24, 24, 256) 1024
- max_pooling2d_1 (MaxPooling2D (2,2)) (12, 12, 256) 0
- dropout_1 (Dropout (0.25)) (12, 12, 256) 0
- conv2d_2 (Conv2D 512 (3,3)) (12, 12, 512) 3277312
    - Activación: *RELU*
    - Strides: 1
    - Padding: *same*
- batch_normalization_2 (BatchNormalization) (12, 12, 512) 2048
- max_pooling2d_2 (MaxPooling2D (2,2)) (6, 6, 512) 0
- dropout_2 (Dropout (0.25)) (6, 6, 512) 0
- flatten (Flatten) (18432) 0
- dense (Dense) (256) 4718848
    - Activación: *RELU*
- batch_normalization_3 (BatchNormalization) (256) 1024
- dropout_3 (Dropout (0.50)) (256) 0
- dense_1 (Dense) (512) 131584
    - Activación: *RELU*
- batch_normalization_4 (BatchNormalization) (512) 2048

- dropout_4 (Dropout (0.50)) (512) 0
- dense_2 (Dense) (7) 3591
    - Activación: SOFTMAX

==========

Total params: 8,434,439
Trainable params: 8,431,111
Non-trainable params: 3,328
Funcion de perdida: ‘categorical cross entropy’
Optimizador: Adam
Uso de Reduce Learning Rate on Plateau
Epochs: 50
Batches: 64

Se aumentó el número de features que obtienen las capas convolucionales y el número de neuronas de las capas densas. Se aplicaron padding en *same* y strides en 1 para intentar reducir la pérdida de información en los bordes de la imagen. Se utilizó Batch Normalization después de cada red densa y convolucional para reducir el overfitting, a la vez que una capa de dropout por la misma razón, aumentando el porcentaje de dropout al 50% únicamente en las capas densas para evitar perder información importante de las primeras capas de la red.
Por último se aplicó un mecanismo de reducción de learning rate para forzar a la red a reducir su learning rate cuando detecte que no se está mejorando la precisión de los resultados.
El ReduceLearningRateOnPlateau se aplicó con la siguiente configuración:

- Learning Rate inicial: 0.005
- factor: 0.1
- Patience: 2
- Learning rate minimo: 0.00001
- model: auto

Los resultados obtenidos fueron los siguientes:

val_accuracy: 0.6331847310066223

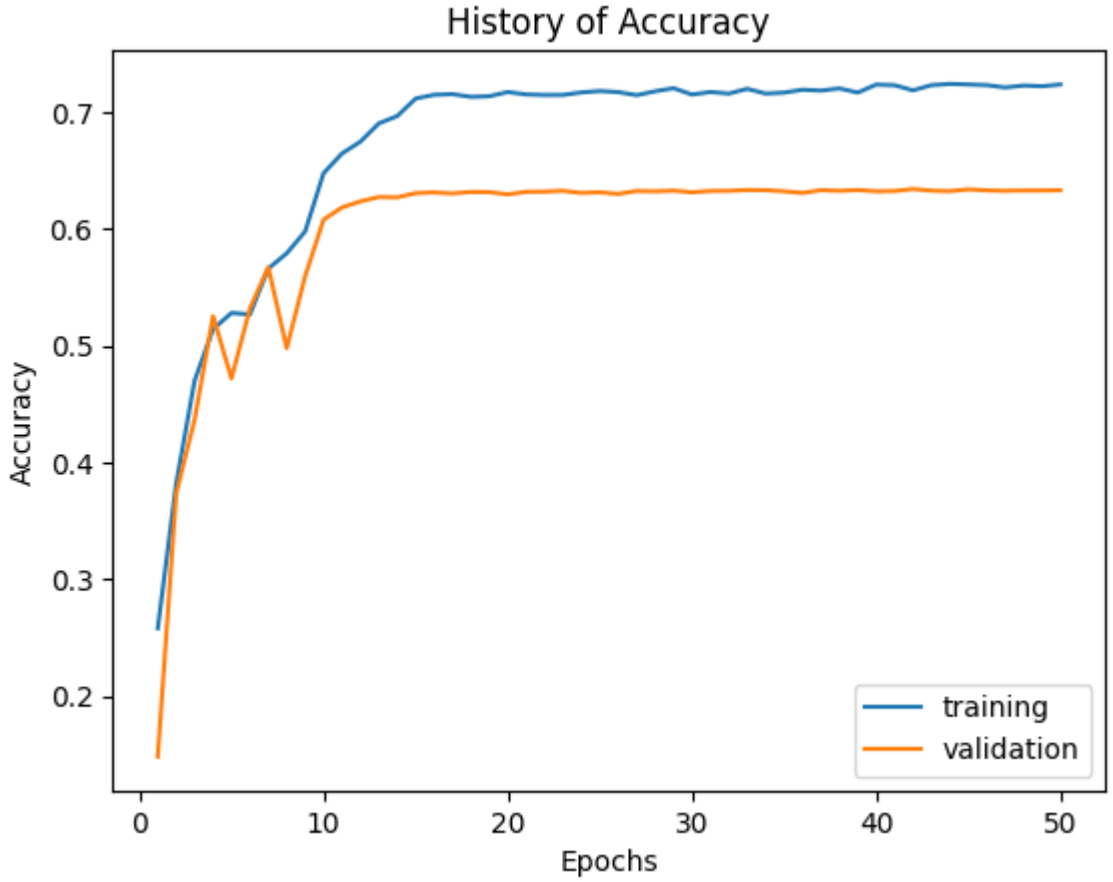

Gráfico 5. Resultados de red con mejor performance obtenida.

Se puede ver que la performance mejora considerablemente respecto a los demás intentos.

Mejoras:

Para intentar mejorar la performance de esta última red neuronal, se probaron algunos métodos más. La reducción del número de imágenes en batches, inicializar las redes densas utilizando kernel_initializer='he_uniform' para probar la performance de la red, y también utilizar otras arquitecturas de redes convolucionales como VGG16.

Un ejemplo de los resultados de una red utilizando esta arquitectura es el siguiente.

Val_accuracy: 0.4224017858505249

Gráfico 6. Resultados de red con arquitectura VGG 16.

La cual utilizó una capa VGG 16, junto con tres redes densas de 512, 128 y 64 neuronas respectivamente, y una red densa más de 7 neuronas para la salida. También se aplicaron capas de Batch Normlization y dropout de 25% para la capa de convolucional y de 50% para las densas.

Se probaron 10 epoch con 64 imágenes por batch y se descartó la utilización futura de esta arquitectura debido al alto consumo de tiempo que produce entrenar una red tan grande a la vez que una red tan grande no produce una mejoría tan notable en un conjunto de datos tan pequeño.

Para probar si es posible corregir el impacto que produce el desbalance del conjunto de datos de FER-2013 en la performance de las redes neuronales, se propuso balancear los datos del conjunto de manera manual, eliminando exceso de imágenes y sacando las categorías que presentan un desbalance muy significativo. No se hizo ningún cambio al conjunto de datos de testeo.

Luego de esto, la base de datos presentó estas distribuciones.

- 3041 Imagenes para Neutral
- 3039 Imagenes para Miedo
- 3033 Imagenes para Enojo
- 3049 Imagenes para Sorpresa
- 3008 Imagenes para Felicidad
- 3022 Imagenes para Tristeza

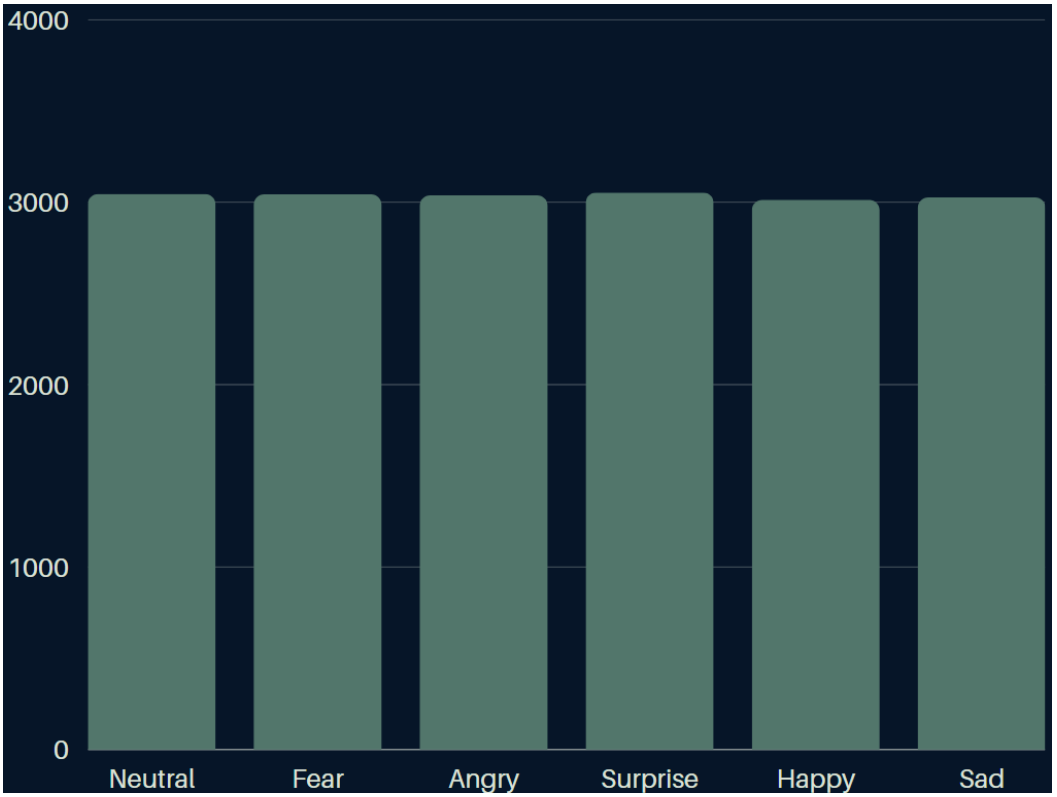

Gráfico 7. Distribución del dataset balanceado para testing.

No se obtuvo ninguna mejora al utilizar este nuevo set de datos balanceados en las redes presentadas anteriormente.

Se cambió la cantidad de neuronas de la última capa densa en cada modelo de 7 a 6 para tener en consideración que el nuevo dataset posee 6 categorías en lugar de 7.

Para la primera red con capas convolucionales, aplicada bajo las mismas condiciones, se obtuvo lo siguiente.

Val_accuracy: 0.5051707029342651

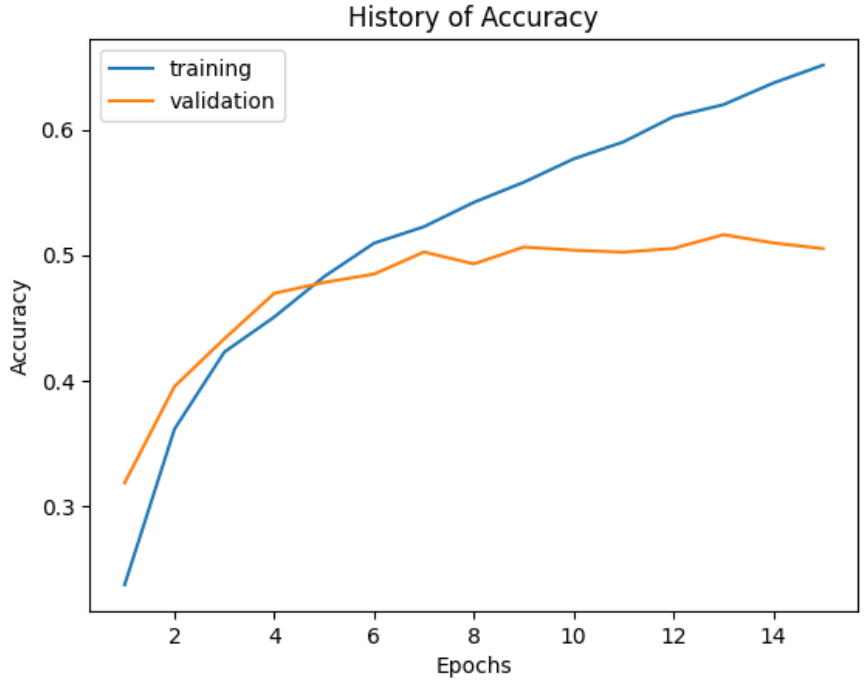

Gráfico 8. Resultados de red con capas convolucionales, usando dataset balanceado.

Para la red óptima, se aplicó bajo las mismas condiciones, exceptuando que se utilizó un batch de 32 en lugar de 64, en busca de una mejor performance.

Val_accuracy: 0.5881853103637695

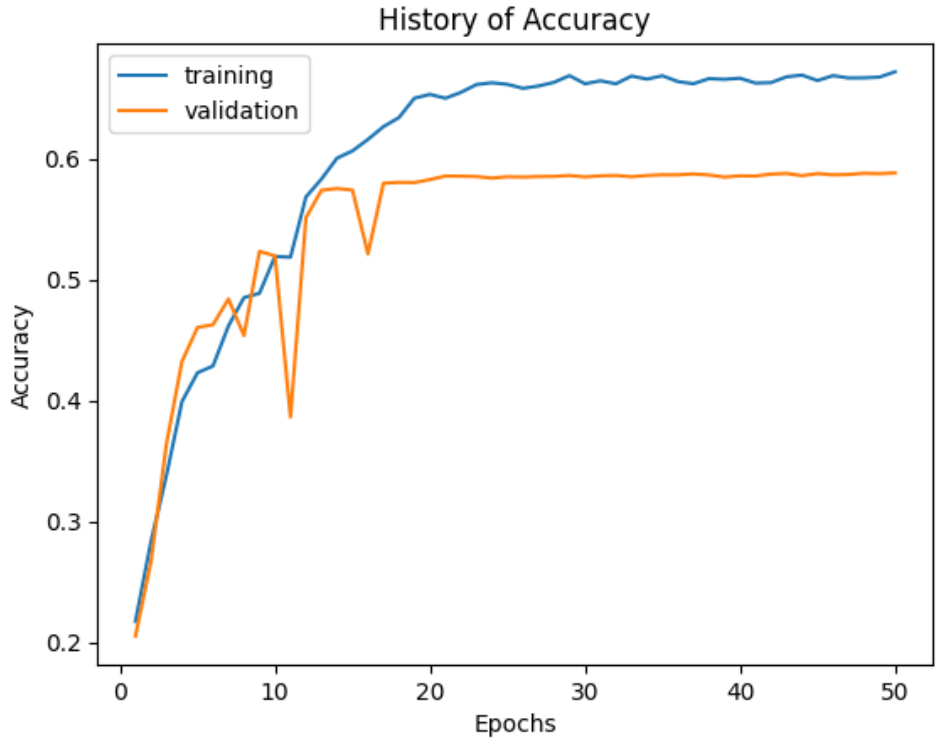

Gráfico 9. Resultados de red con mejor performance utilizando dataset balanceado.

Conclusión

El conjunto de datos FER-2013 presenta una gran cantidad de imágenes que pueden ser utilizadas para el entrenamiento del reconocimiento de emociones y expresiones faciales, pero el desbalance de sus datos, la limitación en la cantidad de imágenes y la complejidad de los datos a analizar, dificultan la obtención de una precisión elevada.
Investigadores han utilizado[2] otras arquitecturas de redes, tales como ResNet 50, la cual es una red neuronal de 50 capas de profundidad, o VGG 16, la cual posee 16 capas de profundidad, para obtener una performance de más del 70%, utilizando también, datos auxiliares para ayudar a nivelar las categorías que normalmente la red no puede reconocer correctamente de manera frecuente.

Enlaces:

1. https://www.kaggle.com/datasets/msambare/fer2013
2. http://cs230.stanford.edu/projects_winter_2020/reports/32610274.pdf