Detector de mascarillas

Julio Castaño Amorós

Febrero 2021

1 Introducción

Este proyecto consiste en entrenar una red neuronal convolucional para detectar si una persona lleva mascarilla o no. Para facilitar un poco el problema, utilizamos una red de OpenCV que detecta caras en la imagen.

2 Objetivos

Poner en práctica los conocimientos visto en clase sobre redes neuronales y ampliar conocimientos sobre este tipo de técnica.

3 Datos

El dataset utilizado se llama FaceMaskDetection. Este dataset está divido en tres conjuntos: train, validación y test. Un buen factor a tener en cuenta es que el dataset está balanceado con las siguientes cifras para cada conjunto:

- Train: con mascarilla 4156, sin mascarilla 4156.
- Validación: con mascarilla 1160, sin mascarilla 1160.
- Test: con mascarilla 567, sin mascarilla 593.

Con el objetivo de aumentar la cantidad de datos, se utiliza una técnica conocida como Data Augmentation, la cual consiste en aplicar transformaciones a los datos, tales como rotación, zoom...

Un factor importante a tener en cuenta es que se modifica el tamaño de las imágenes a 150x150 debido a que VGG19 fue entrenada con imágenes del reto ImageNet.

4 Desarrollo

Este proyecto está formado por tres programas distintos:

- Script de entrenamiento.
- Script de test en una única imagen.
- Script de test en un vídeo.

4.1 Script de entrenamiento

Este script consiste en aplicar Transfer Learning a una red neuronal con arquitectura VGG19. Esta arquitectura ha sido elegida debido a aspectos de hardware y debido también a que ya se he utilizado para otros proyectos personales.

Transfer Learning es una técnica que consiste en cargar una red (sin la capa de clasificación) con los pesos entrenados para un problema mucho mayor comparado con el que queremos realizar. De esta manera, los pesos de las capas convolucionales han aprendido a extraer características interesantes que, aunque no sean del problema que buscamos resolver, son de gran utilidad como un comienzo para nuestro entrenamiento.

El entrenamiento de la red se ha realizado con los siguientes parámetros:

- Clasificación binaria: 1 neurona en capa de salida, función de pérdida: bynary cross-entropy, función de activación en capa de salida: sigmoide aplicando un umbral.
- Optimizador: RMSprop con learning rate inicial de 2e-5.
- 30 épocas.
- 100 pasos por época.
- 25 pasos por validación.
- Para el resto se utilizan los valores por defecto.

Después del entrenamiento, se obtienen dos gráficas, una compara la precisión en train vs validación, y la otra compara el error en train vs validación.

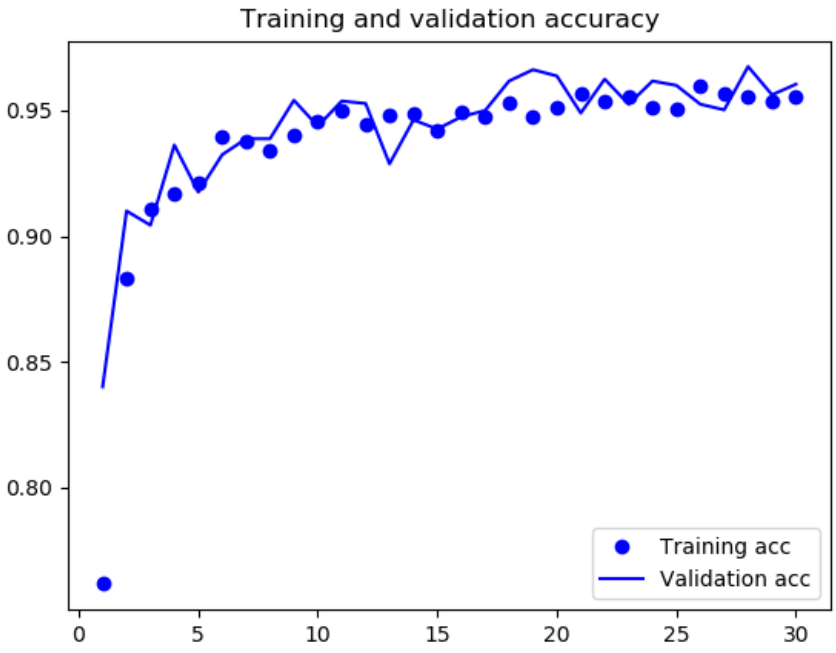

Figure 1: Accuracy.

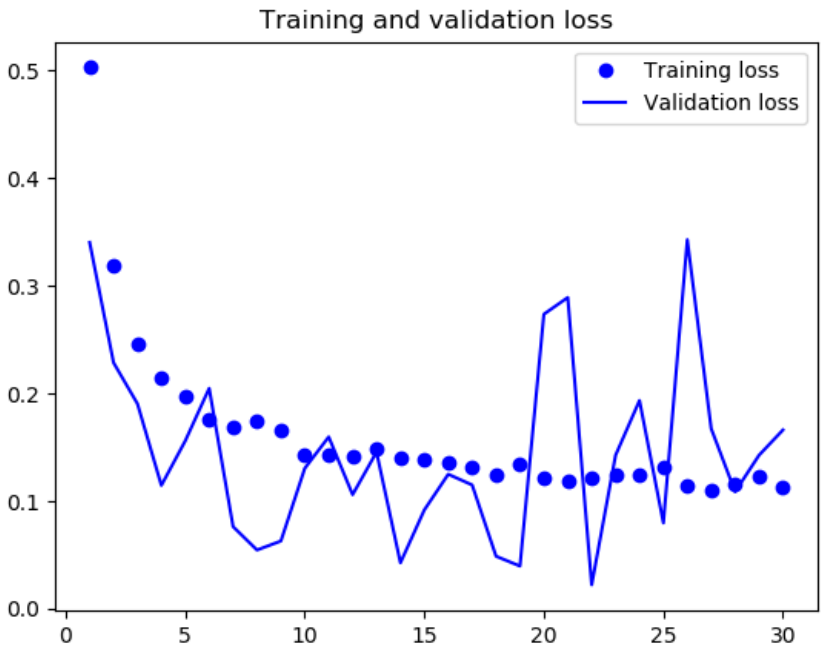

Figure 2: Loss.

5 Scripts de test y resultados

Una vez la red ha sido entrenada, el siguiente paso es lanzar la predicción sobre el conjunto de test para obtener una cifra de precisión que indique cuán bien detecta en ejemplos que no ha visto antes. La precisión obtenida para el conjunto de test es de 0.986.

El siguiente paso es utilizar la red para predecir ejemplos que no forman parte de este dataset. De esta manera se puede observar cuán bien la red ha aprendido a generalizar.

Para este propósito se desarrollan dos scripts, uno que detecta en una sóla imagen, y otro que detecta en un vídeo. Como se ha comentado en el apartado de introducción, se utiliza una red ya entrenada perteneciente al paquete dnn de la libería OpenCV para la detección de caras en imágenes. Los resultados de estos scripts se podrán observar en el vídeo adjuntado en la entrega. El vídeo se ve ralentizado debido a que el hardware utilizado no es potente ni adecuado para este tipo de aplicaciones.

6 Conclusión y futuras mejoras

Como conclusión, me gustaría comentar que he podido poner en práctica conocimientos tanto aprendidos en la asignatura como otros que ya conocía por experiencia propia sobre redes neuronales y Deep Learning.

Pienso que los resultados son buenos, pero siempre mejorables. Desde mi punto de vista, considero que las siguientes mejoras pueden ser interesantes:

- Mejorar el dataset. Investigando un poco en las muestras del dataset, se puede observar que hay muestras que pueden resultar un tanto extrañas, bien porque aparecen caras que no son de personas reales o porque hay caras que están medio cortadas.
- Podría ser interesante entrenar una red que detectara todo en un conjunto, es decir, que detecte si la persona lleva mascarilla o no en toda la imagen. Así se podría reducir el coste computacional ya que esta aplicación sólo tiene sentido en tiempo real.
- Utilizar una arquitectura de red más reciente. Como se ha comentado, se elige VGG porque ya he trabajado con esta arquitectura en proyectos anteriores, tanto VGG16 como VGG19, aunque es cierto que son arquitecturas un poco antiguas y que se podrían utilizar otras más recientes con las que obtendríamos mejores resultados.