Tecnológico de Monterrey

Implementación de un modelo de Deep Learning

María Fernanda Torres Alcubilla A01285041

24 de noviembre del 2023

Deep Learning

Profesor. Christian Carlos Mendoza Buenrostro

Para esta implementación se hizo uso del dataset cifar10 de la librería tensorflow, este dataset consiste de 60mil imágenes a color de tamaño 32x32, estas están divididas equitativamente en 10 clases, las cuales son: aviones, carros, pájaros, gatos, venados, perros, ranas, caballos, barcos y camiones. Cada imagen es exclusiva a esa clase, lo que significa que no hay dos objetos en una imagen y para los modelos se usaron 50mil imágenes para entrenamiento y 10mil para el subset de prueba y se usó un modelo de CNN.

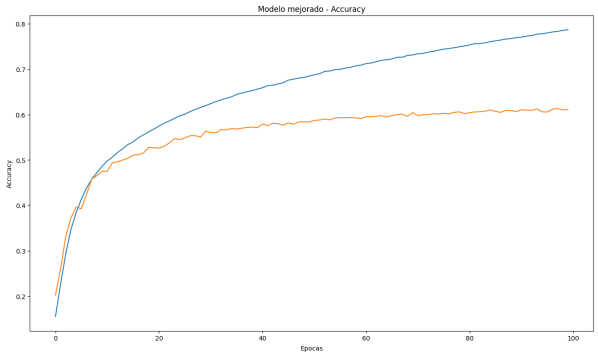
El modelo 1 consiste en una red CNN, con múltiples capas, especialmente densas al finalizar, esto con el objetivo de ayudar con la especialización y aumentar las métricas. Para esto, se utilizó la siguiente arquitectura presentada en el código, se tienen múltiples capas Conv2D, con el objetivo del aprendizaje de características, al combinarlas con MaxPoolin2D se logra la reducción de dimensionalidad de entrada para la conservación de las características más importantes. Al finalizar, se tienen 4 capas densas, más la de salida.

```
model = models.Sequential()
model.add(Conv2D(filters=32,kernel_size=(3,3),padding='same',input_shape=(
32,32,3), activation='relu'))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(units=512, activation='relu'))
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=10, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

Este modelo se compiló con la función de pérdida ‘sparse_categorical_crossentropy’ y el optimizador a ‘adagrad’ y un total de 100 épocas. En la Imagen 1 se tiene la precisión y pérdida a través de las épocas, donde la línea azul representa al set de entrenamiento y la naranja el de validación. Observamos que se presenta una clara situación de ‘overfitting’, ya que los valores de precisión del subset de validación no aumentan, mientras el de entrenamiento si.

En el subset de prueba se obtuvo una precisión de 0.61, el cual no es aceptable.



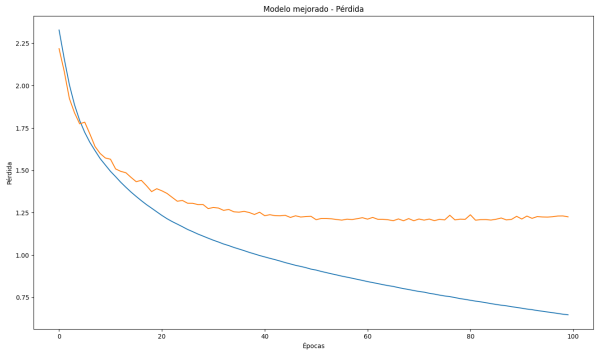


Imagen 1.1 Precisión del modelo inicial *Imagen 1.2* Pérdida del modelo inicial

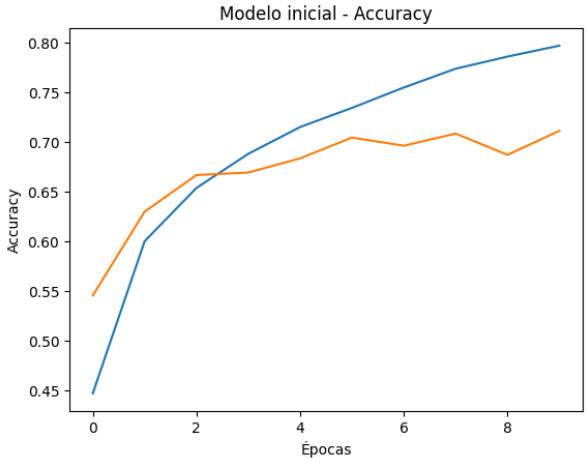
Imagen 1. Métricas del modelo inicial respecto a la época

Para el segundo modelo se usaron las imagenes normalizadas con la arquitectura descrita en el código, además, se observa que el modelo empieza con un input 32x32 (tamaño de la imagen original) y como es a color se tienen 3 canales, se agregan capas de agrupación máxima para la reducción de dimensionalidad de la salida de la capa anterior y al final se aplica una capa densa y en la de salida la función de activación softmax para la clasificación multiclase, donde se regresa la probabilidad de que la imagen pertenezca a cada clase.

El modelo, a diferencia con el modelo anterior, para solucionar el overfitting, se disminuyeron las épocas ya que la precisión para el set de validación no crecía en gran medida y además se eliminaron algunas capas de Conv2D y capas densas para disminuir la especialización. Después, se compiló el modelo con el optimizador de 'adam' y función de pérdida 'sparse_categorical_crossentropy' y se entrenó con 10 épocas, la métrica a evaluar en este caso fue la precisión y pérdida. A continuación, en la Imagen 2 se observan las gráficas de estos valores con respecto a la época, donde la línea azul representa en el subset de entrenamiento y la naranaja en el de validación.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32,
32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```



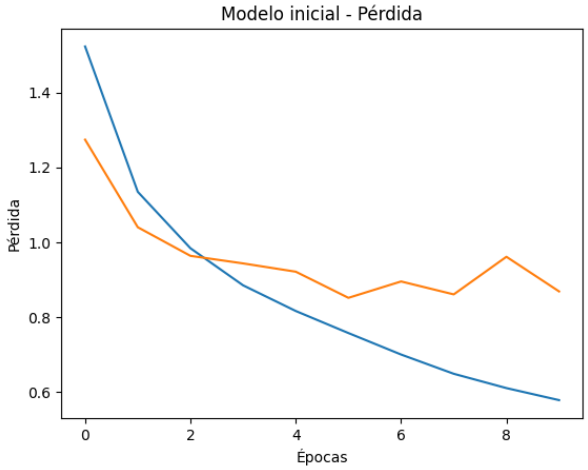


Imagen 1.1 Precisión del modelo final *Imagen 1.2* Pérdida del modelo final

Imagen 1. Métricas del modelo final respecto a la época

A partir de la Imagen 2.1 y 2.2 observamos que se produce un poco de overfitting pero mejora notablemente a comparación con el modelo 1, esto se observa ya que al pasar las épocas el modelo se ajusta a los datos de entrenamiento pero al momento de realizar validación ya no hay mejoría y en momentos decrece. En el subset de prueba se obtuvo una precisión de 0.71, el cual es un bastante mejor al anterior.

Finalizando, se realizaron diversas predicciones con imágenes recientes obtenidas de internet, no parecidas al dataset original, a continuación se muestran las imagenes, su predicción y confianza, respectivamente. Observamos que no clasifica a la perfección pero obtuvo ¾ correctas.

Avión
100%

Carro
100%

Avión
100%

Gato
99.99%

Referencias

Adhishthite. (s. f.). *Cifar10-Optimizers/CIFAR-Optimizers.ipynb at Master · Adhishthite/Cifar10-Optimizers*. GitHub. https://github.com/adhishthite/cifar10-optimizers/blob/master/CIFAR-Optimizers.ipynb

CIFAR-10 and CIFAR-100 datasets. (s. f.). https://www.cs.toronto.edu/~kriz/cifar.html