Practica FSI

Javier Díaz Domínguez, Marcos Couros García

Dataset

El dataset está compuesto por 5 conjuntos de fotos tomadas con cámaras de móvil en nuestras casas. Las categorías elegidas son cubiertos, destornilladores, libros, tazas y zapatos.

Una vez elaborado el datases se procede a la creación y entrenamiento del modelo.

Data augmentation

Para aportar un poco de variarción sobre el conjunto de entrenamiento y de validación se han empleado métodos de *data augmentation* en la clase ImageDataGenerator.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range= 0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range= 0.1,
    horizontal_flip=True
)
```

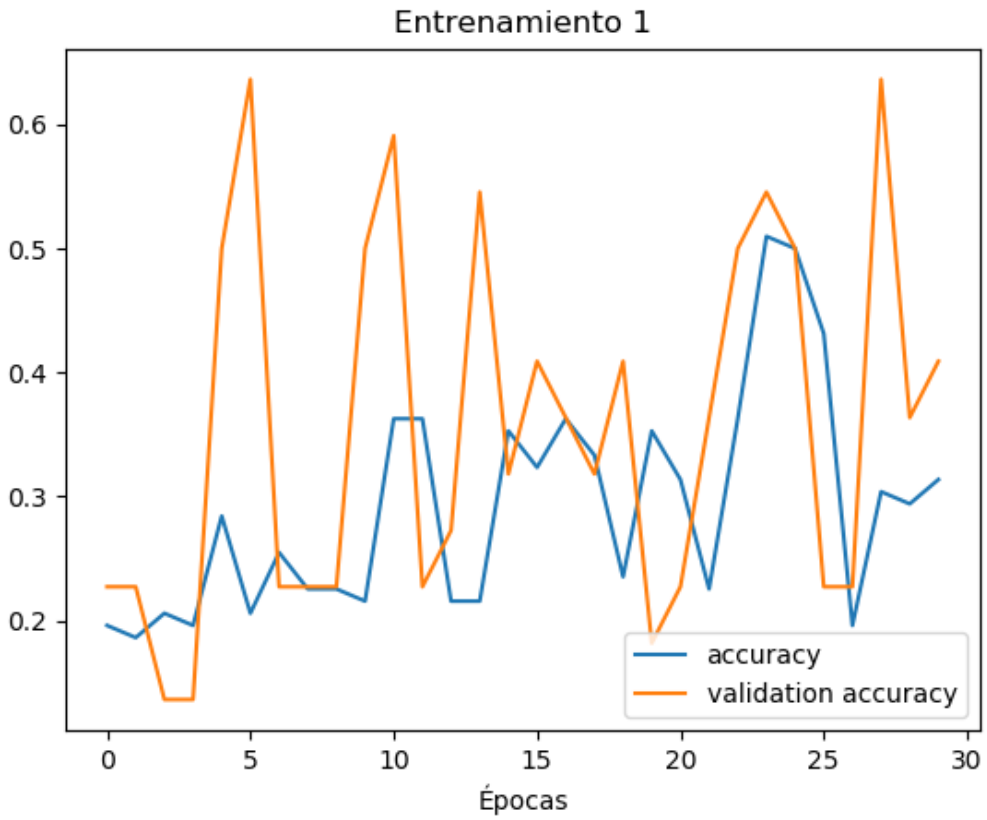
Los parámetros empleados son el rango de rotación, el rango de cambio de anchura y de altura, de zoom y de volteado lateral. Con estos parámetros podemos aumentar el número de muestras de entrenamiento a partir del dataset original.

Modelo

Para la elaboración del modelo se partió del modelo suministrado en las prácticas, con dos capas convolutivas, una capa RELU y la última softmax.

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=
(288,162,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])
```

El resultado del entrenamiento con este modelo en 30 épocas fue el siguiente

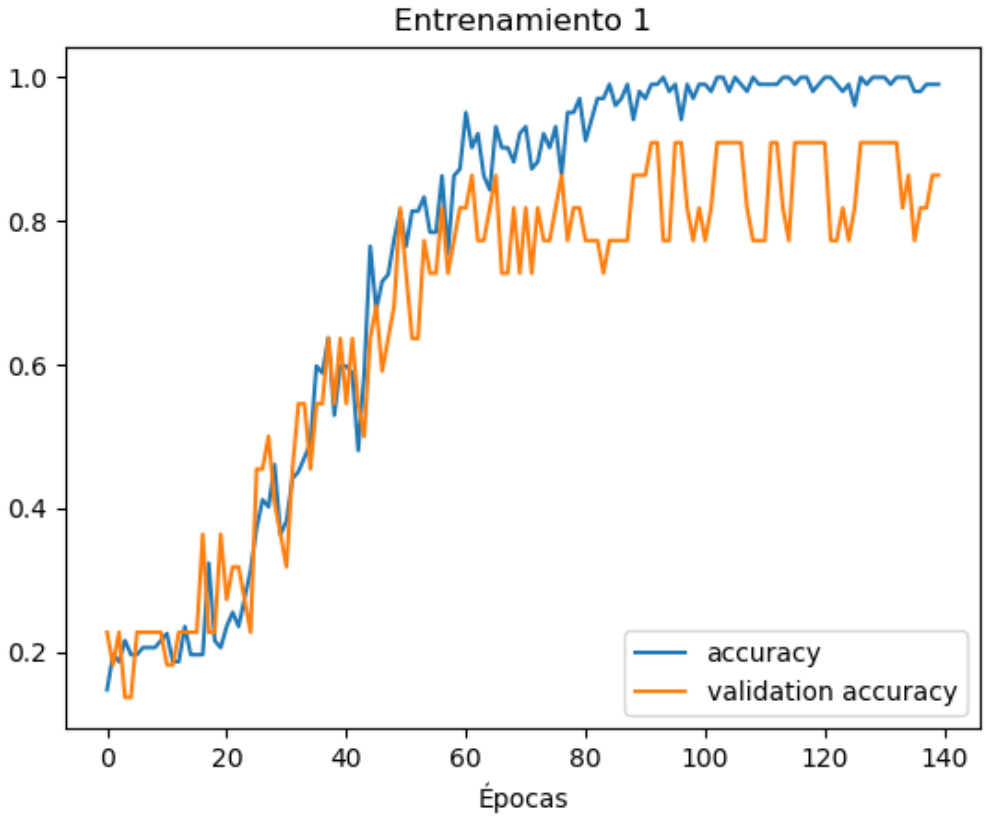


Como podemos observar en la gráfica, el valor de la precisión en la predicción del conjunto de validación y en el conjunto de entrenamiento fluctúa de manera descontrolada. Esto puede ser indicativo de que el optimizador empleado no es el mejor para el conjunto de datos con el que trabaja el modelo. Por ello, después de hacer algunos cambios en el modelo y el optimizador, hemos encontrado los mejores resultados en el siguiente modelo

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=
(288,162,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(60, activation='relu'))
model.add(Dense(30, activation='relu'))
model.add(Dense(5, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
```

A demás del cambio de optimizador, hemos quitado una capa convolutiva y añadido una capa de activación RELU.

Los resultados obtenidos con este modelo no son los mejores para el conjunto de validación, pero teniendo en cuenta el tamaño del dataset y la falta de diferenciación entre algunas categorías, consideramos el resultado suficientemente satisfactorio, con alrededor de un 80% de precisión para el conjunto de validación

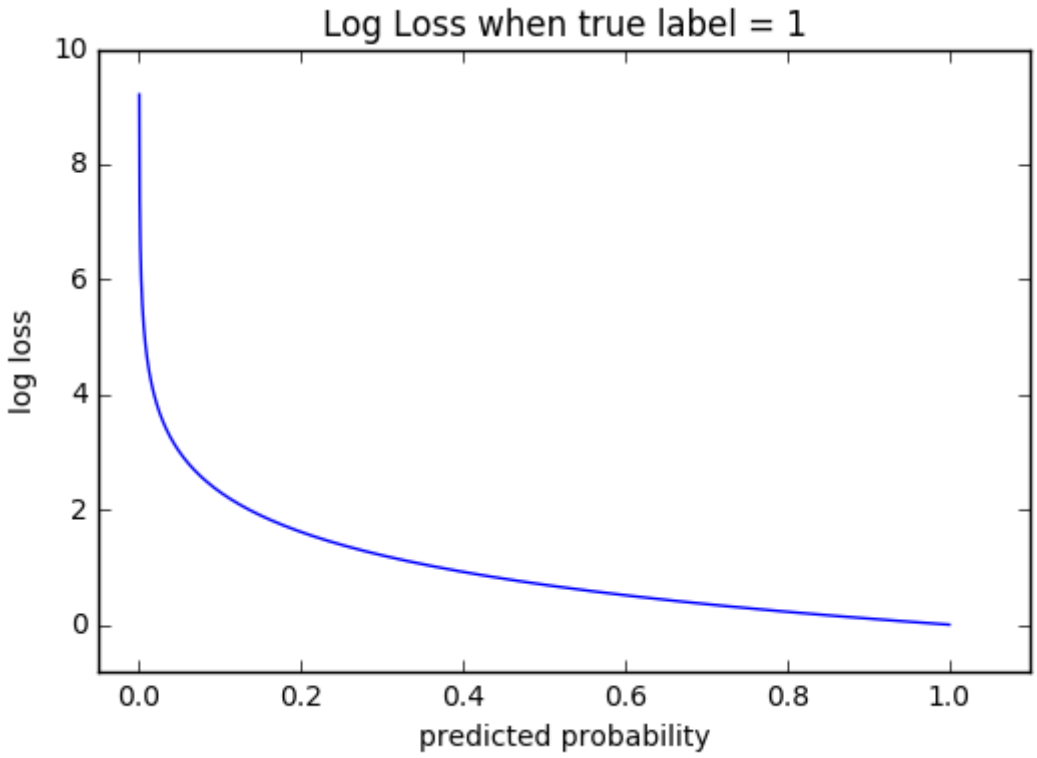


Categorical Cross Entropy

La función de pérdida Categorical Cross Entropy devuelve mayores valores de error para resultados muy lejanos del valor etiquetado. De esta manera, se penalizan con mayor severidad las predicciones erróneas que distan mucho del resultado real, ayudando al modelo a descartar los parámetros que aumenten este valor.

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{N}\sum_{i}^{N}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)]$$

Con esta formula, el valor de la pérdida para un valor real de 1 en función del valor predecido quedaría de la siguiente manera


Log Loss when true label = 1
log loss
predicted probability
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0