DEEP LEARNING AND AI

PROYECTO IV

ALBERT FERNANDEZ SOLER

PROYECTO 1

ARQUITECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', input_shape=(32,32,3)))
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', input_shape=(32,32,3)))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(32, activation='relu'))
model.add(ks.layers.Dense(10, activation='softmax'))
```

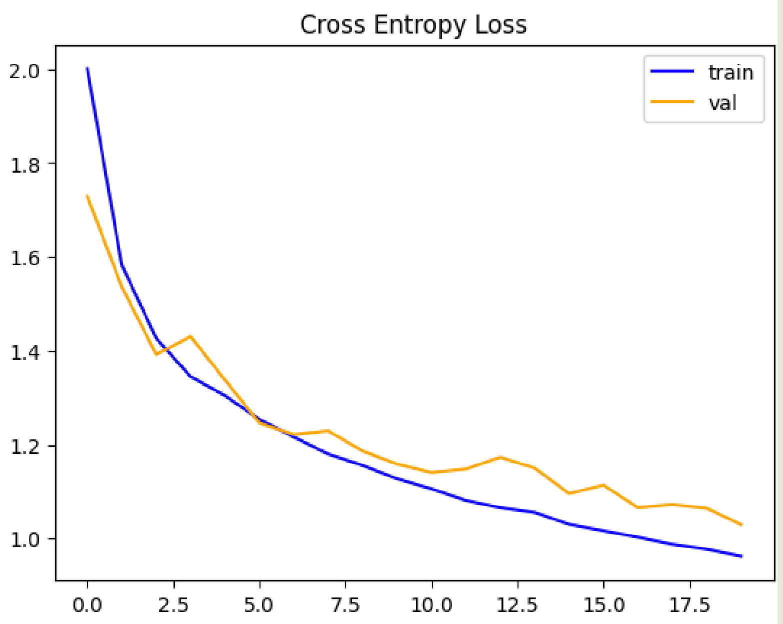
Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_9 (Conv2D)	(None, 16, 16, 32)	9,248
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 32)	0
flatten_8 (Flatten)	(None, 2048)	0
dense_16 (Dense)	(None, 32)	65,568
dense_17 (Dense)	(None, 10)	330

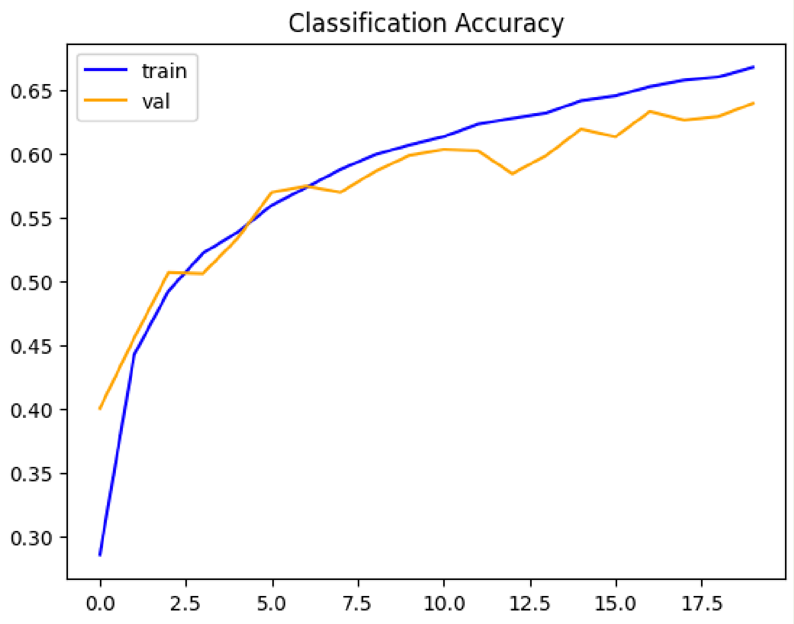
```
history = model.fit(x_train_scaled, y_train,
                    epochs=20,
                    batch_size= 512,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
✓ 1.3s

> 64.320
```





Para el proyecto 1, hemos añadido callbacks y creado una estructura un poco más compleja con una capa convulacional más. Ahora que tenemos nuestro benchmark, podemos empezar a mejorar el modelo.

PROYECTO 2

ARQUiTECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', input_shape=(32,32,3)))
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', input_shape=(32,32,3)))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(32, activation='relu'))
model.add(ks.layers.Dense(10, activation='softmax'))
```

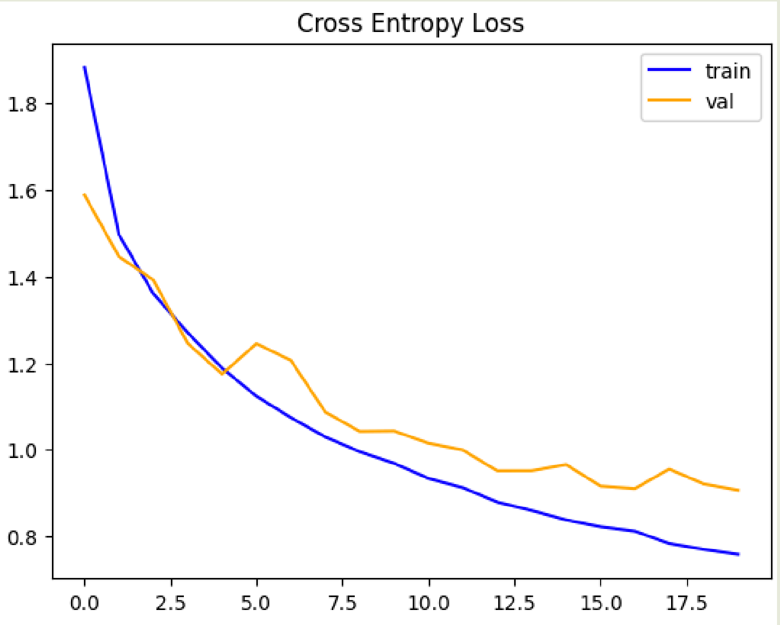
Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 32, 32, 32)	896
conv2d_11 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_12 (Conv2D)	(None, 16, 16, 32)	9,248
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 32)	0
conv2d_13 (Conv2D)	(None, 8, 8, 32)	9,248
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 32)	0
flatten_9 (Flatten)	(None, 512)	0
dense_18 (Dense)	(None, 32)	16,416
dense_19 (Dense)	(None, 10)	330

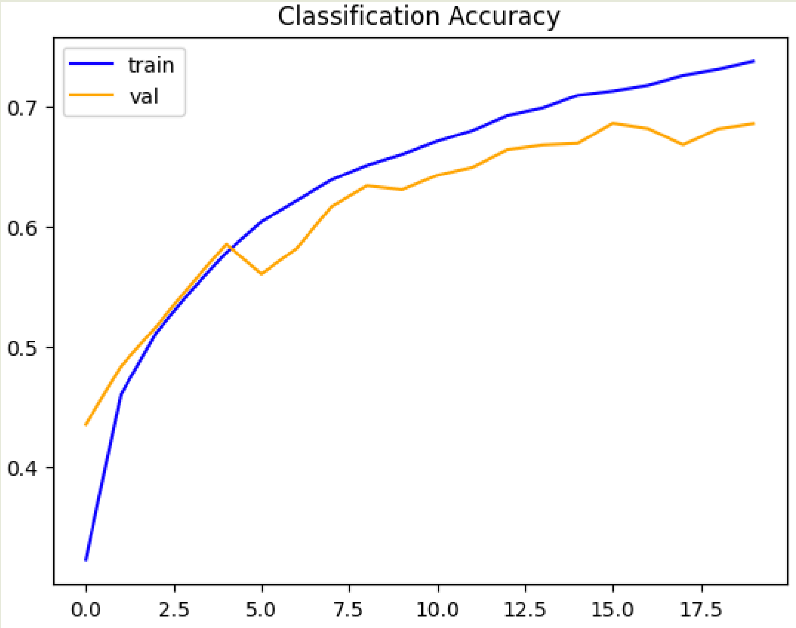
```
history = model.fit(x_train_scaled, y_train,
                    epochs=20,
                    batch_size= 512,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
✓ 3.2s

> 68.340
```





El modelo ha mejorado pero no llega al 80% mínimo esperado. Empezamos a ver un posible overfiting por lo que aumentaremos los callbacks. Augmentaremos el numero de capas, filtros y neuronas para hacer una estrucutra más compleja que permita al modelo mejorar su accuracy.

PROYECTO 3

ARQUITECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(16, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(ks.layers.Conv2D(16, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform'))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same',  kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same'))
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(64, activation='relu'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(32, activation='relu'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(10, activation='softmax'))
```

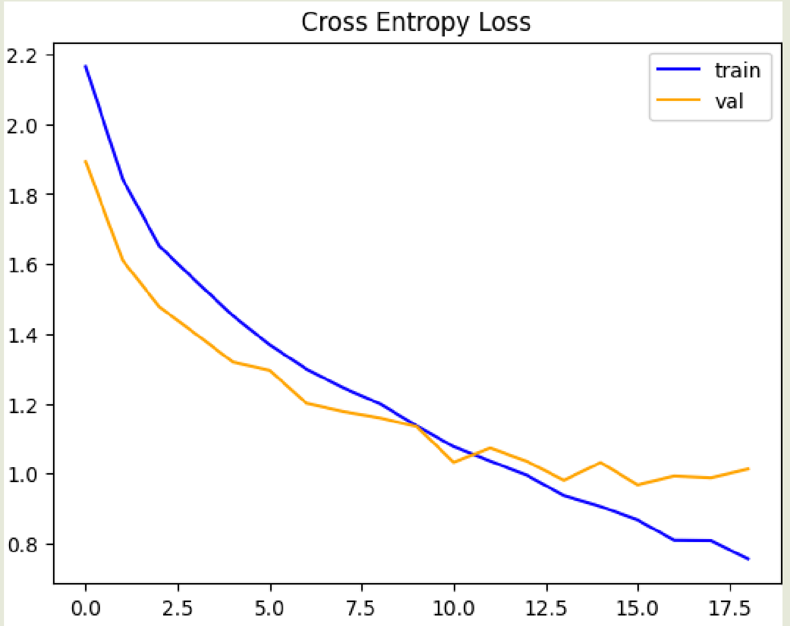
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 16)	448
conv2d_1 (Conv2D)	(None, 32, 32, 16)	2,320
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
conv2d_2 (Conv2D)	(None, 16, 16, 32)	4,640
conv2d_3 (Conv2D)	(None, 16, 16, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 32)	0
conv2d_4 (Conv2D)	(None, 8, 8, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 64)	32,832
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 10)	330

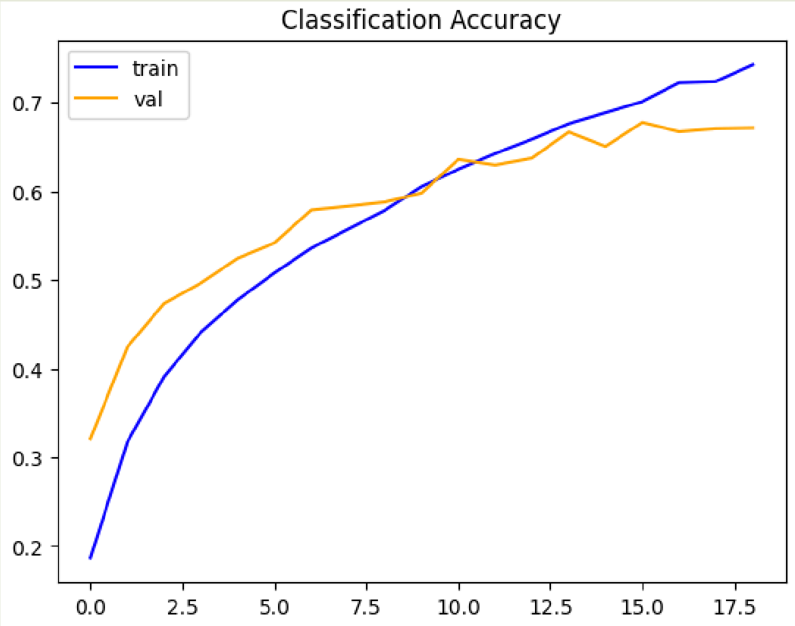
```
history = model.fit(x_train_scaled, y_train,
                    epochs=20,
                    batch_size= 512,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
✓ 3.1s

> 66.910
```





Pese a ser un modelo más complejo, el accuracy ha bajado y ha augmentado el overfitting. Reduciremos el dropout y augmentaremos la cantidad de capas, filtros y neuronas para aumentar el numero de parametros a entrenar.

PROYECTO 4

ARQUiTECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(10, activation='softmax'))
```

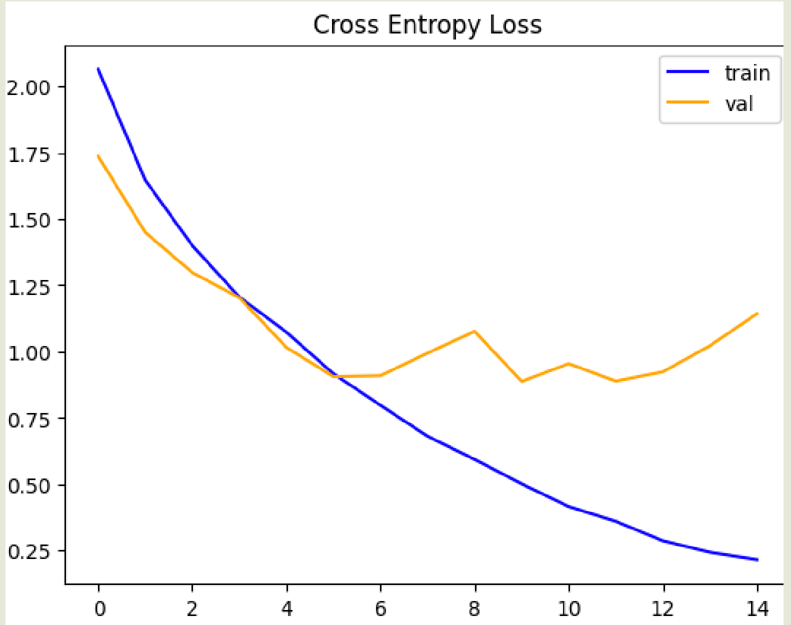
Layer (type)	Output Shape	Param #
conv2d_46 (Conv2D)	(None, 32, 32, 32)	896
conv2d_47 (Conv2D)	(None, 32, 32, 32)	9,248
max_pooling2d_24 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_48 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_49 (Conv2D)	(None, 16, 16, 64)	36,928
max_pooling2d_25 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_50 (Conv2D)	(None, 8, 8, 128)	73,856
conv2d_51 (Conv2D)	(None, 8, 8, 128)	147,584
max_pooling2d_26 (MaxPooling2D)	(None, 4, 4, 128)	0
conv2d_52 (Conv2D)	(None, 4, 4, 256)	295,168
conv2d_53 (Conv2D)	(None, 4, 4, 256)	590,080
max_pooling2d_27 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten_6 (Flatten)	(None, 1024)	0
dense_19 (Dense)	(None, 128)	131,200
dropout_13 (Dropout)	(None, 128)	0
dense_20 (Dense)	(None, 128)	16,512
dropout_14 (Dropout)	(None, 128)	0
dense_21 (Dense)	(None, 128)	16,512
dropout_15 (Dropout)	(None, 128)	0
dense_22 (Dense)	(None, 10)	1,290

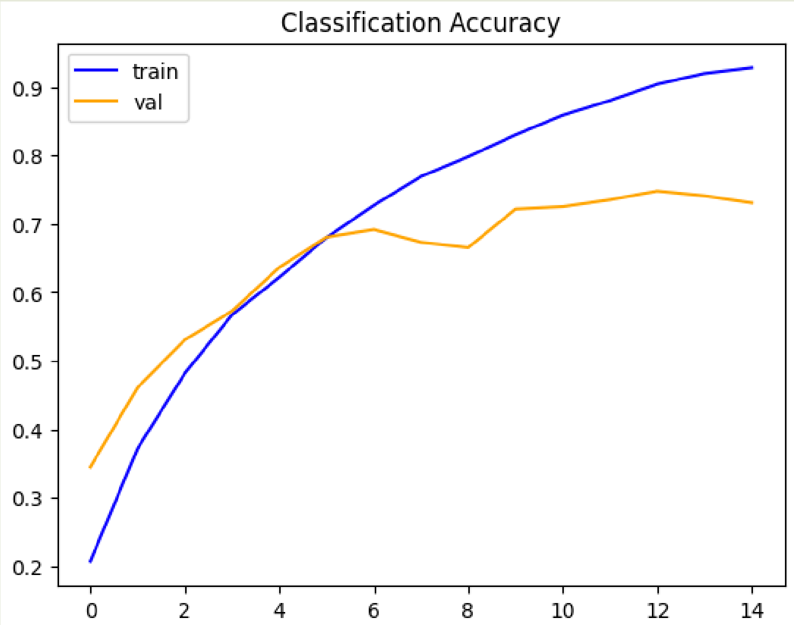
```
history = model.fit(x_train_scaled, y_train,
                    epochs=20,
                    batch_size= 512,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
✓ 6.0s

> 73.410
```





El accuracy ha augmentado considerablemente pero tambien el overfitting. Augmentaremos el dropout en una capa de dense y eliminaremos una capa de dense y ajustaremos el numero de neuronas. Elimaremos las últimas capas para reducir complejidad al modelo.

Añadiremos batchnormalization para añadir estabilidad al modelo.

PROYECTO 5

ARQUiTECTURA

```python
model = ks.Sequential()

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(192, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))


model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
conv2d_78 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_79 (Conv2D)	(None, 32, 32, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_40 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_80 (Conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_81 (Conv2D)	(None, 16, 16, 64)	36,928
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_41 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_82 (Conv2D)	(None, 8, 8, 192)	110,784
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 192)	768
conv2d_83 (Conv2D)	(None, 8, 8, 256)	442,624
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_42 (MaxPooling2D)	(None, 4, 4, 256)	0
flatten_11 (Flatten)	(None, 4096)	0
dense_39 (Dense)	(None, 128)	524,416
dropout_28 (Dropout)	(None, 128)	0
dense_40 (Dense)	(None, 64)	8,256
dropout_29 (Dropout)	(None, 64)	0
dense_41 (Dense)	(None, 10)	650

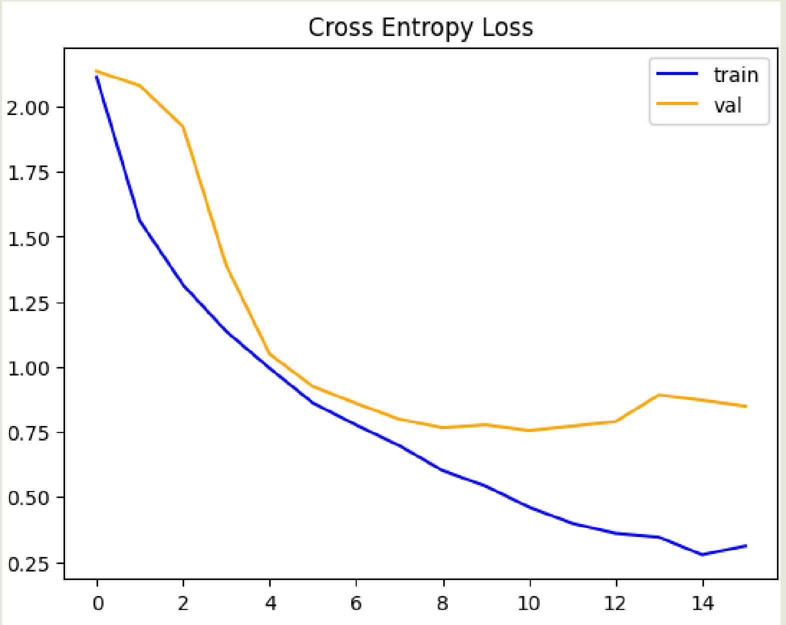
```python
history = model.fit(x_train_scaled, y_train,
                    epochs=30,
                    batch_size= 512,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

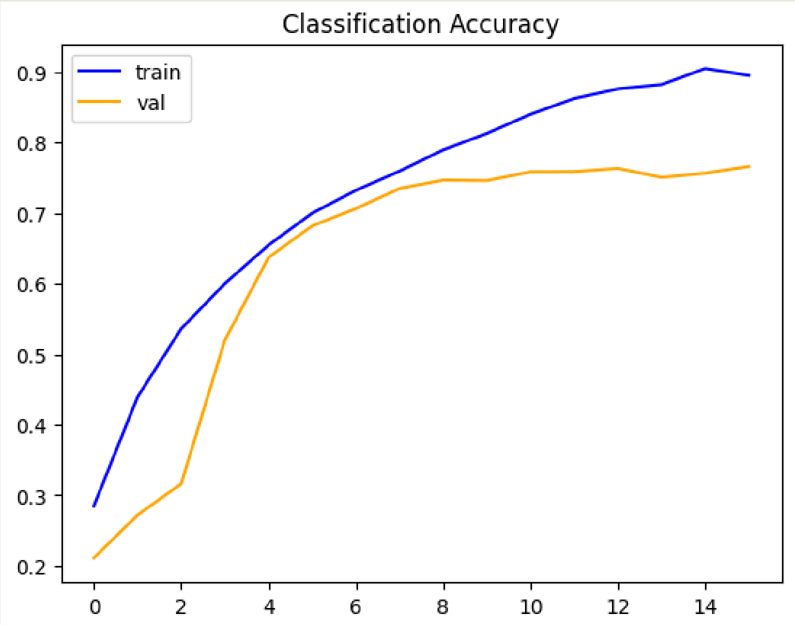
ACCURACY

```python
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
```

✓ 6.3s

```
> 76.160
```





El accuracy sigue augmentando pero tambien el overfitting. A partir de las epocas 14-15 ya no aprende más. Creo que tengo una arquitectura más sólida pero aún hay que hacer ajustes para llegar al 80%.

Añadiré más capas para un augmento más progresico, añadiré una capa de neuronas Dense y haré mejoras en el Learning Rate para reducir el overfitting y conseguir más estabilidad en el modelo.

PROYECTO 6

ARQUiTECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))

model.add(ks.layers.Conv2D(192, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(224, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))


model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(216, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.2))
model.add(ks.layers.Dense(10, activation='softmax'))
```

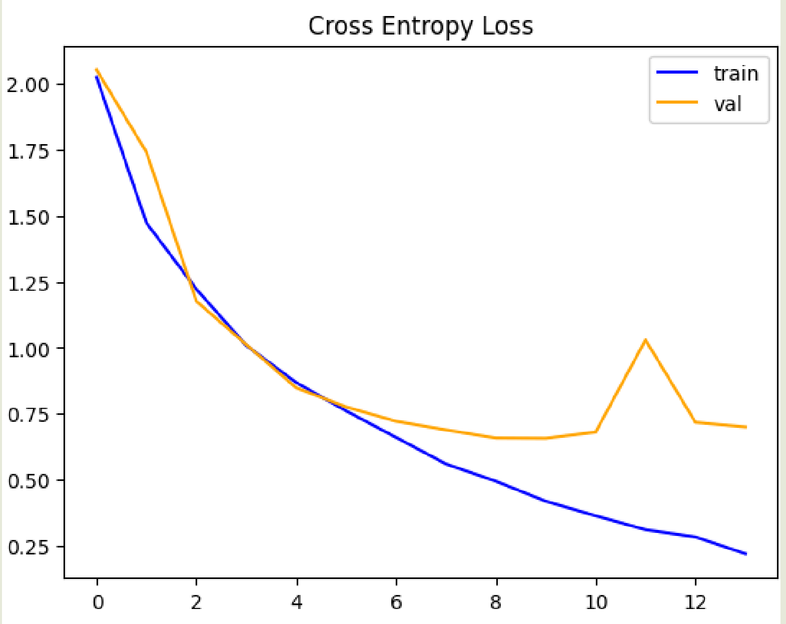
Layer (type)	Output Shape	Param #
conv2d_123 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_45 (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_124 (Conv2D)	(None, 32, 32, 32)	9,248
batch_normalization_46 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_61 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_125 (Conv2D)	(None, 16, 16, 64)	18,496
batch_normalization_47 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_126 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_48 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_62 (MaxPooling2D)	(None, 8, 8, 128)	0
conv2d_127 (Conv2D)	(None, 8, 8, 192)	221,376
batch_normalization_49 (BatchNormalization)	(None, 8, 8, 192)	768
conv2d_128 (Conv2D)	(None, 8, 8, 224)	387,296
batch_normalization_50 (BatchNormalization)	(None, 8, 8, 224)	896
conv2d_129 (Conv2D)	(None, 8, 8, 256)	516,352
batch_normalization_51 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_63 (MaxPooling2D)	(None, 4, 4, 256)	0
flatten_18 (Flatten)	(None, 4096)	0
dense_60 (Dense)	(None, 216)	884,952
dropout_42 (Dropout)	(None, 216)	0
dense_61 (Dense)	(None, 128)	27,776
dropout_43 (Dropout)	(None, 128)	0
dense_62 (Dense)	(None, 64)	8,256
dropout_44 (Dropout)	(None, 64)	0
dense_63 (Dense)	(None, 10)	650

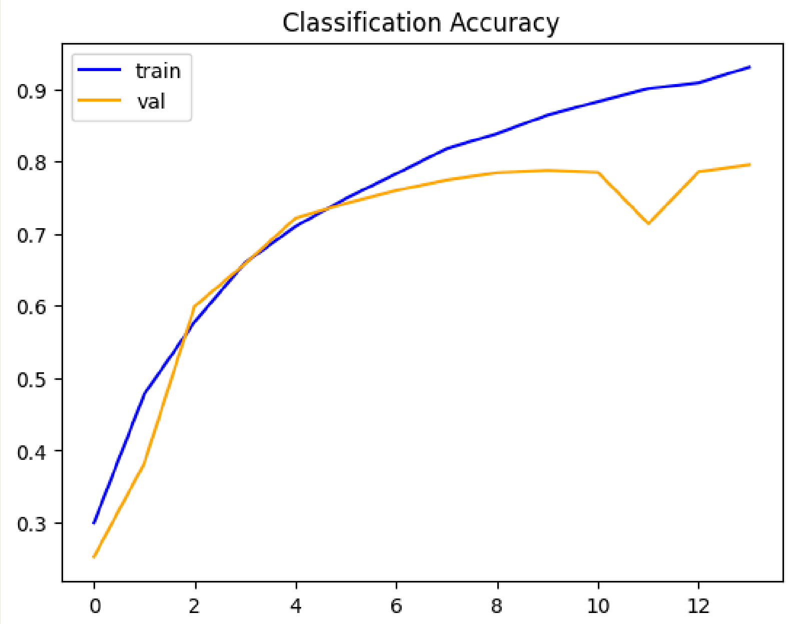
```
history = model.fit(x_train_scaled, y_train,
                    epochs=30,
                    batch_size= 256,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
✓ 11.6s

> 78.670
```





Estamos cerca del 80% lo que nos indica que debemos hacer ajustes menores en la arquitectura. Añadiremos el Learning Rate dinamico para una mejor generalización y reducción del overfitting. Con este objetivo, augmentaremos las neuronas y filtros de las primeras capas y augmentamos el dropout en la primera capa densa. Además también añadiremos dropout después de las capas convulacionales.

PROYECTO 7

ARQUITECTURA

```
model = ks.Sequential()

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))


model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))


model.add(ks.layers.Conv2D(224, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))



model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(10, activation='softmax'))
```

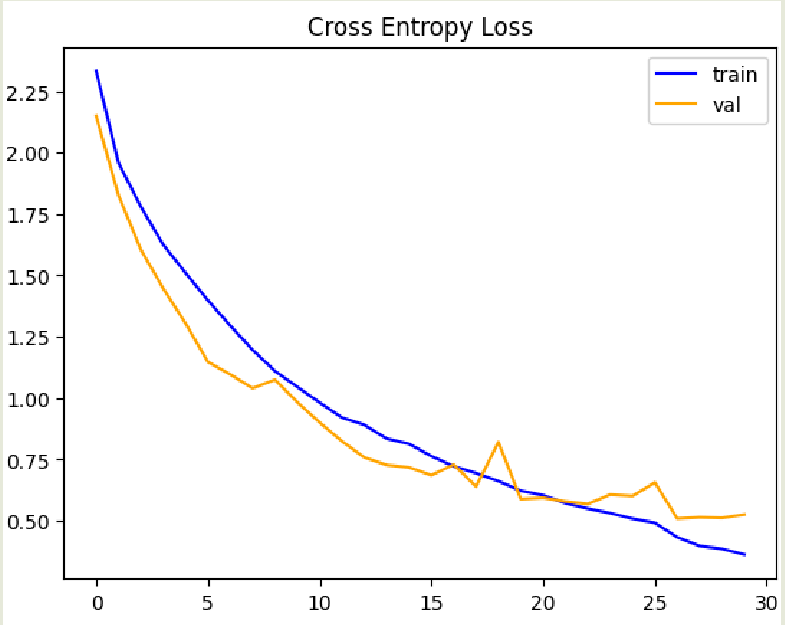
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1,792
batch_normalization (BatchNormalization)	(None, 32, 32, 64)	256
conv2d_1 (Conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
dropout (Dropout)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 128)	512
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 224)	258,272
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 224)	896
conv2d_5 (Conv2D)	(None, 8, 8, 256)	516,352
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	0
dropout_2 (Dropout)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 256)	1,048,832
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_4 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 10)	650

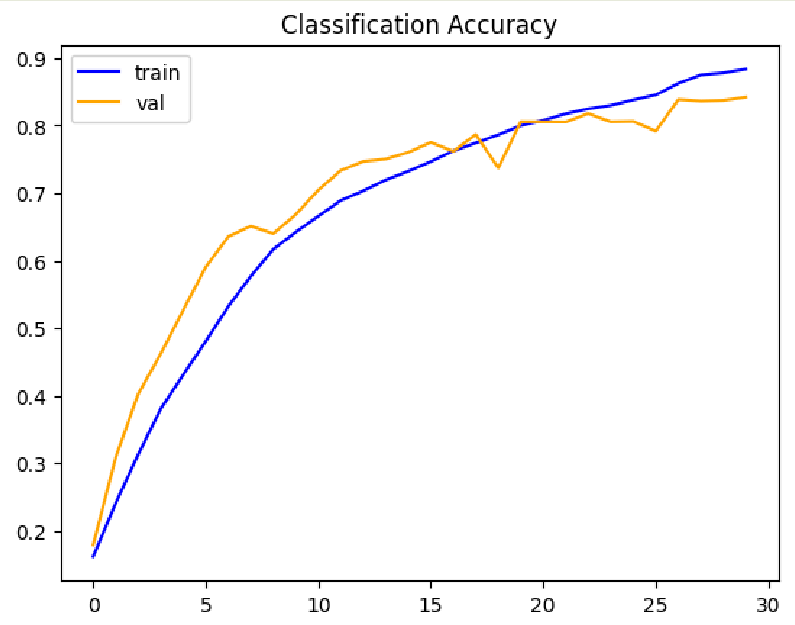
```
history = model.fit(x_train_scaled, y_train,
                    epochs=30,
                    batch_size= 256,
                    validation_data=(x_val_scaled, y_val),
                    callbacks=[callback_val_loss, callback_val_accuracy, reduce_lr])
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
> 83.360
```





Hemos superado la barrera del 80% y ademas el modelo no presenta un overfitting claro. Para evitar oscilaciones en la val_loss modificaremos la learning rate. También incluiremos tecnicas de data augmentation para tratar de mejorar el modelo.

PROYECTO 8

ARQUiTECTURA

```python
model = ks.Sequential()

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))


model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))


model.add(ks.layers.Conv2D(224, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.4))



model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.4))
model.add(ks.layers.Dense(10, activation='softmax'))
```

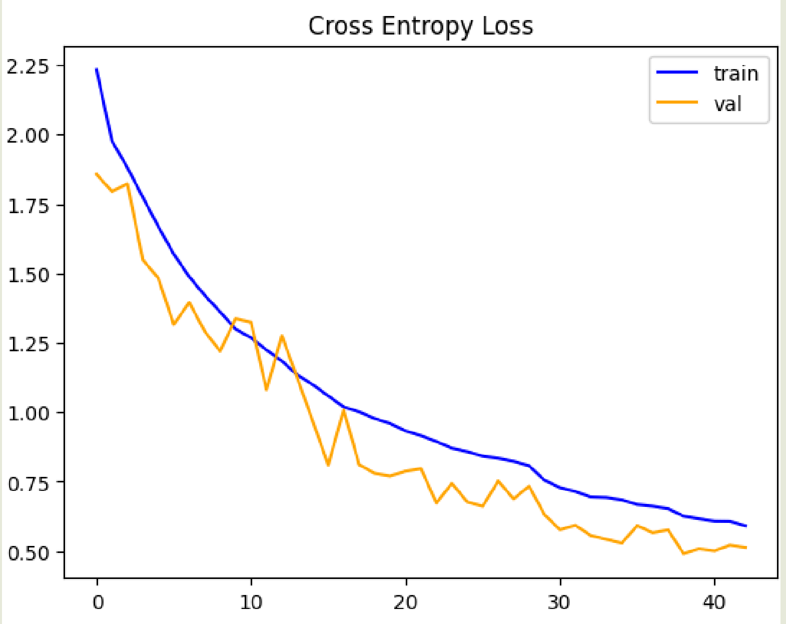
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1,792
batch_normalization (BatchNormalization)	(None, 32, 32, 64)	256
conv2d_1 (Conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
dropout (Dropout)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 128)	512
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 224)	258,272
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 224)	896
conv2d_5 (Conv2D)	(None, 8, 8, 256)	516,352
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	0
dropout_2 (Dropout)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 256)	1,048,832
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_4 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 10)	650

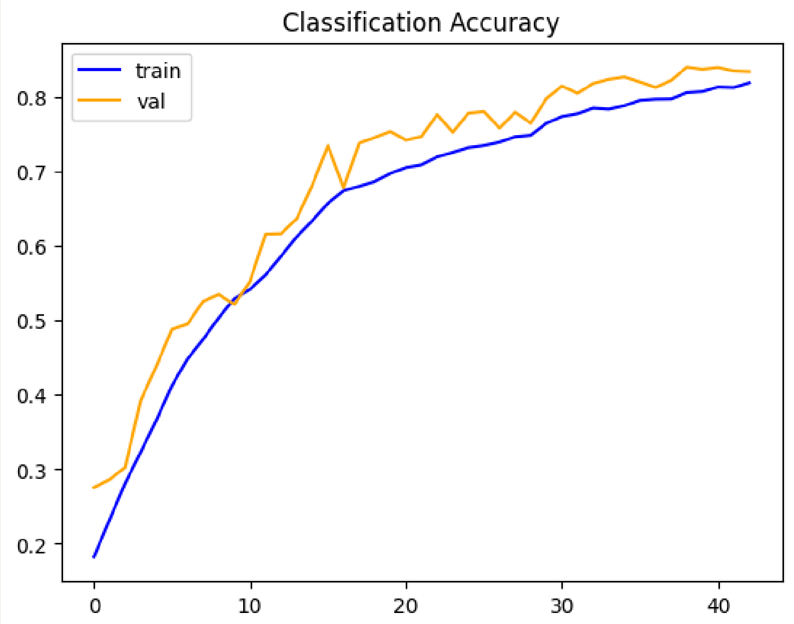
```python
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator,
    callbacks=[callback_val_loss, callback_val_accuracy, reduce_lr]
)
```

ACCURACY

```python
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
> 83.570
```





Vemos que el modelo ha mejorado un poco gracias al data augmentation, pero no significativamente. Probaremos con reducir la learning rate, el dropout y la complejidad del modelo para ver si llegamos a un accuracy mas alto consiguiendo un modelo más estable y que pierda menos información.

PROYECTO 9

ARQUITECTURA

```
model = Sequential()

model.add(Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))


model.add(Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.3))
model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(0.3))
model.add(Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 64)	1,792
batch_normalization_6 (BatchNormalization)	(None, 32, 32, 64)	256
conv2d_7 (Conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_7 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d_3 (MaxPooling2D)	(None, 16, 16, 64)	0
dropout_6 (Dropout)	(None, 16, 16, 64)	0
conv2d_8 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_8 (BatchNormalization)	(None, 16, 16, 128)	512
conv2d_9 (Conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_9 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_4 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_7 (Dropout)	(None, 8, 8, 128)	0
conv2d_10 (Conv2D)	(None, 8, 8, 128)	147,584
batch_normalization_10 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 128)	0

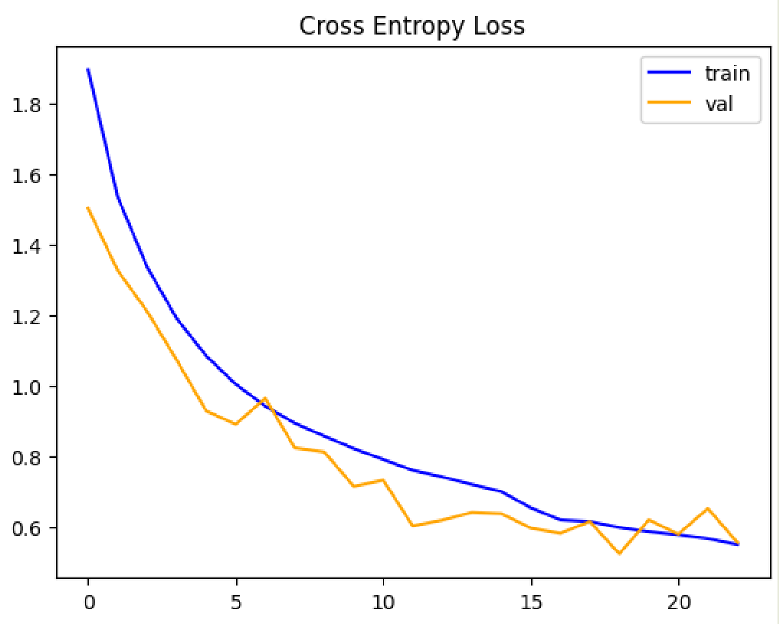
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_8 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 256)	524,544
dropout_9 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 128)	32,896
dropout_10 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 10)	1,290

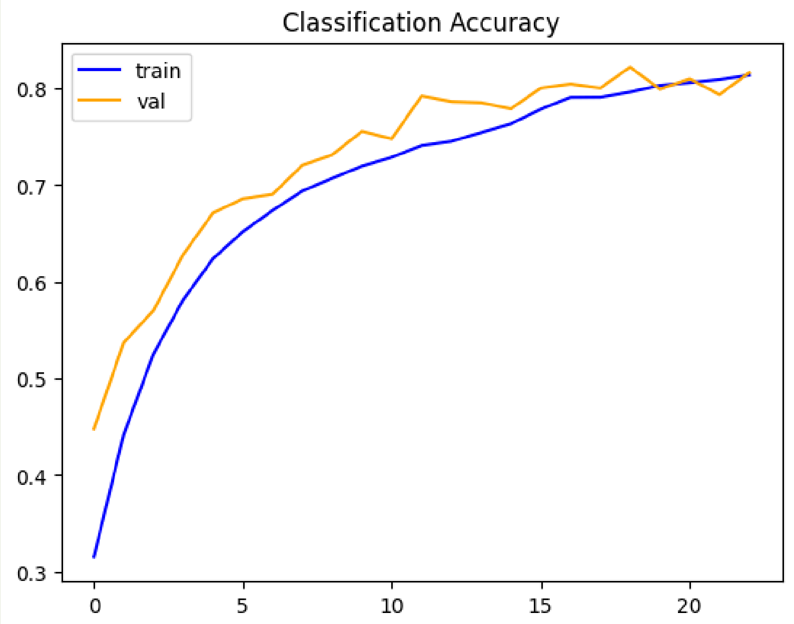
```
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator,
    callbacks=[callback_val_loss, callback_val_accuracy, reduce_lr]
)
```

ACCURACY

```
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
81.650
```





Al hacer un modelo más senzillo y con un LR más bajo, el accuracy ha bajado 2 puntos. Para el último modelo, volveremos al modelo anterior, manteniendo la LR y el dropout del proyecto 8 y siendo un poco más agresivo con los parametros del data augmentation e introducir otros tipos de transformaciones.

PROYECTO 10

ARQUITECTURA

```python
model = ks.Sequential()

model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform', input_shape=(32,32,3)))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(64, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.3))


model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(128, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.3))


model.add(ks.layers.Conv2D(224, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.Conv2D(256, (3, 3), strides=1, activation='relu', padding='same', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(ks.layers.MaxPooling2D((2, 2)))
model.add(ks.layers.Dropout(0.3))


model.add(ks.layers.Flatten())
model.add(ks.layers.Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(ks.layers.Dropout(0.3))
model.add(ks.layers.Dense(10, activation='softmax'))
```

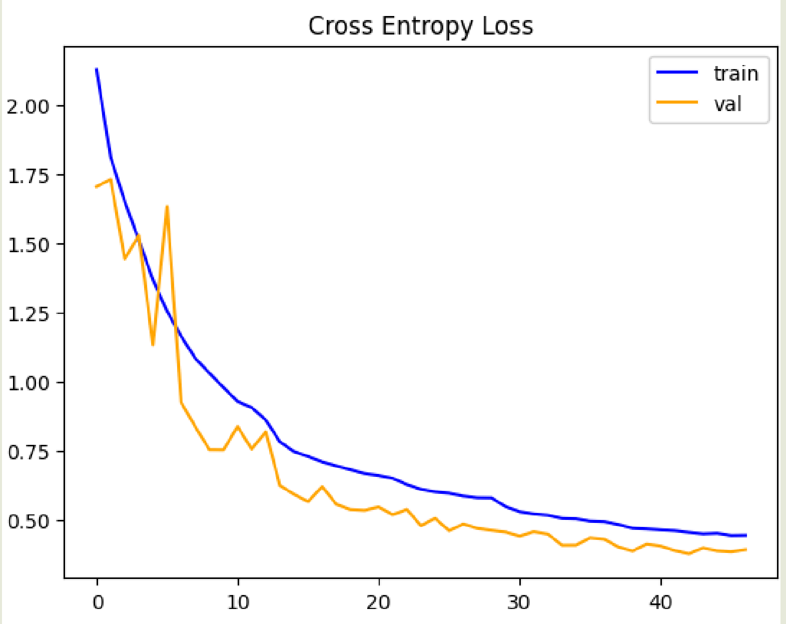
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	1,792
batch_normalization (BatchNormalization)	(None, 32, 32, 64)	256
conv2d_1 (Conv2D)	(None, 32, 32, 64)	36,928
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
dropout (Dropout)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 128)	512
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 224)	258,272
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 224)	896
conv2d_5 (Conv2D)	(None, 8, 8, 256)	516,352
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 256)	1,024
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	0
dropout_2 (Dropout)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 256)	1,048,832
dropout_3 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_4 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 10)	650

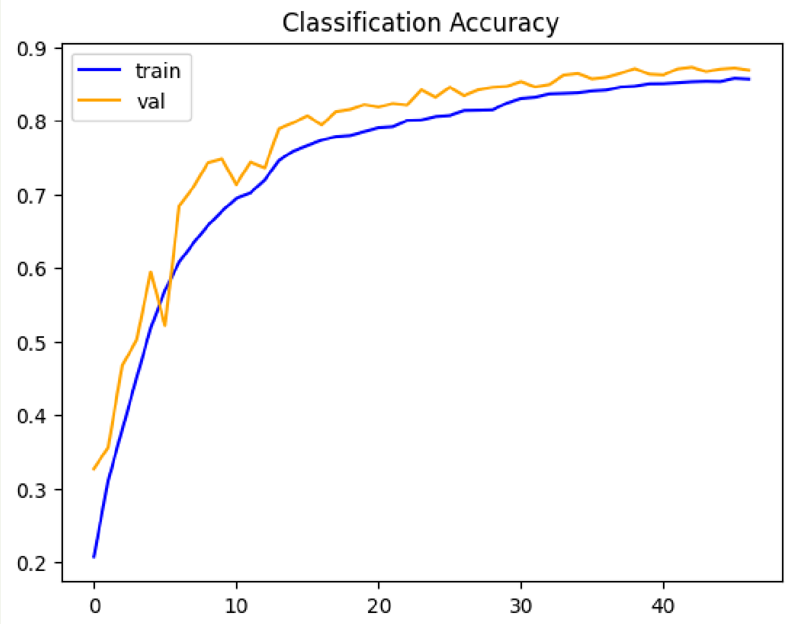
```python
history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator,
    callbacks=[callback_val_loss, callback_val_accuracy, reduce_lr]
)
```

ACCURACY

```python
_, acc = model.evaluate(x_test_scaled, y_test, verbose=0)
print('> %.3f' % (acc * 100.0))
```

```
> 86.940
```





La últimat versión del modelo es la más mejor a nivel de resultados. El accuracy de validación es superior al de train, mostrando una solidez en el aprendizaje. Se observa muy poco overfitting en las gráficas ya que tanto train como validación disminuyen progresivamente.