Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
T.U.I.A

Aprendizaje Automático II

Trabajo Práctico 1

Redes Densas y Convolucionales
2024

Mateo Gravi Fiorino
Lucas Gauto

Problema 1

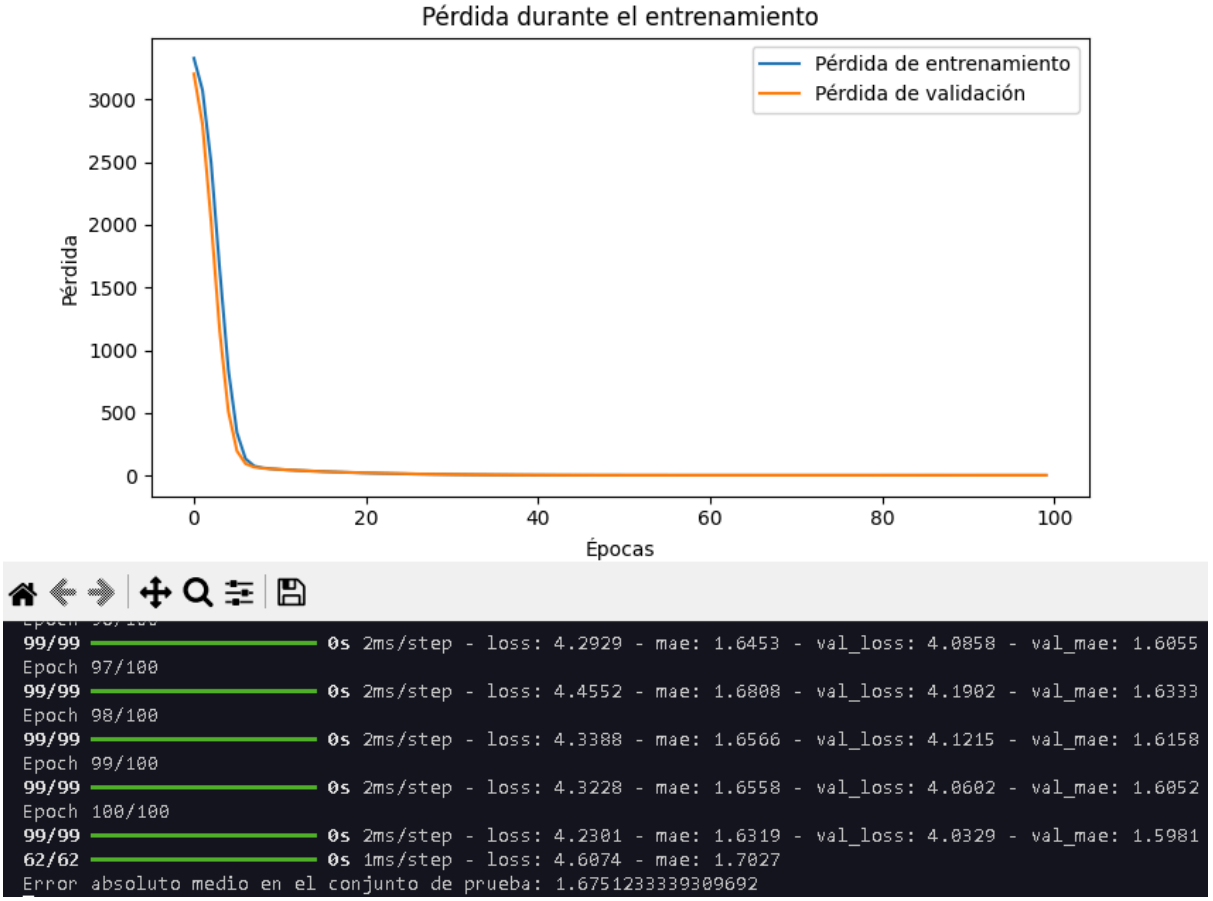
Descripción:
En este problema, se presenta un conjunto de datos que contiene información sobre el rendimiento académico de estudiantes universitarios, así como diversos factores que podrían influir en él. El objetivo es construir un modelo de regresión utilizando redes neuronales para predecir el índice de rendimiento académico de los estudiantes basado en las características proporcionadas.

Para la evaluacion y analisis.
Se debe ejecutar el archivo que se encuentra en:

|
|___ /Problema 1
—-> Scripts
—-> analisis.py

En CLI: ´streamlit run analisis.py´

Resultado de evaluación del modelo.



Problema 2

Vemos unas imágenes de ejemplo. La totalidad puede encontrarse en repositorio o correr el script correspondiente y probar con cámara web.

Problema 3

Se puede encontrar toda la solución en el ipynb correspondiente en el repositorio.
Además, incluimos imágenes en este informe.

Consideraciones: cada uno de los modelos implementados fue entrenado con un número de épocas de 20 y un batch size de 120, para poder facilitar la comparación. Además, se implementó un early stopping para evitar situaciones en las que los modelos dejen de mejorar.

Modelo con capas densas:

El modelo con capas densas, como su nombre lo indica, es un modelo formado únicamente con capas densas o fully-connected. Este modelo, de los cuatro implementados, fue el más básico o con menos complejidad, reflejando en sus resultados.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
flatten (Flatten)	(None, 150528)	0
dense (Dense)	(None, 250)	37,632,250
dense_1 (Dense)	(None, 250)	62,750
dense_2 (Dense)	(None, 250)	62,750
dense_3 (Dense)	(None, 6)	1,506

```
Epoch 1/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 9s 48ms/step - accuracy: 0.1746 - loss: 1.8311 - val_accuracy: 0.1807 - val_loss: 1.7941
Epoch 2/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - accuracy: 0.1615 - loss: 1.8096 - val_accuracy: 0.1943 - val_loss: 1.7894
Epoch 3/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 27ms/step - accuracy: 0.1722 - loss: 1.7953 - val_accuracy: 0.1580 - val_loss: 1.7986
Epoch 4/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - accuracy: 0.1735 - loss: 1.7961 - val_accuracy: 0.1750 - val_loss: 1.7940
Epoch 5/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.1686 - loss: 1.7959 - val_accuracy: 0.1843 - val_loss: 1.7896
Epoch 6/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 27ms/step - accuracy: 0.1723 - loss: 1.7965 - val_accuracy: 0.1750 - val_loss: 1.7966
Epoch 7/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.1723 - loss: 1.7973 - val_accuracy: 0.1843 - val_loss: 1.7904
```

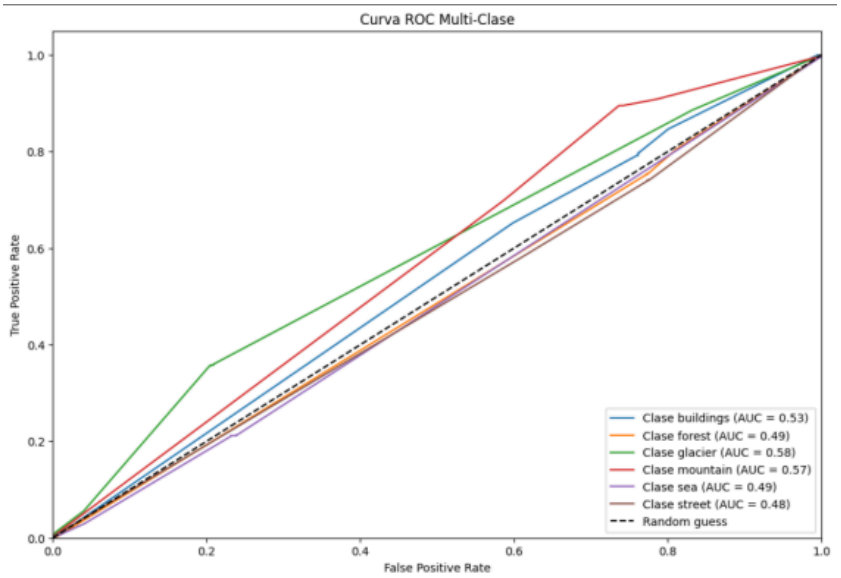
Resultados:

El modelo, como se mencionó anteriormente, no realizó buenos avances en la minimización de la pérdida.

Predicciones con imágenes aleatorias:

Curva ROC:

Analizando el valor del área bajo la curva (AUC) de la curva ROC podemos realizar conclusiones sobre la performance del modelo, entendiendo que un valor cercano al 1 indica una clasificación perfecta y un valor de 0,5 nos informa sobre una clasificación aleatoria. En este caso nos encontramos mucho más cerca de una clasificación aleatoria, donde el AUC de cada clase está cercano al 0,5.



Modelo con capas convolucionales

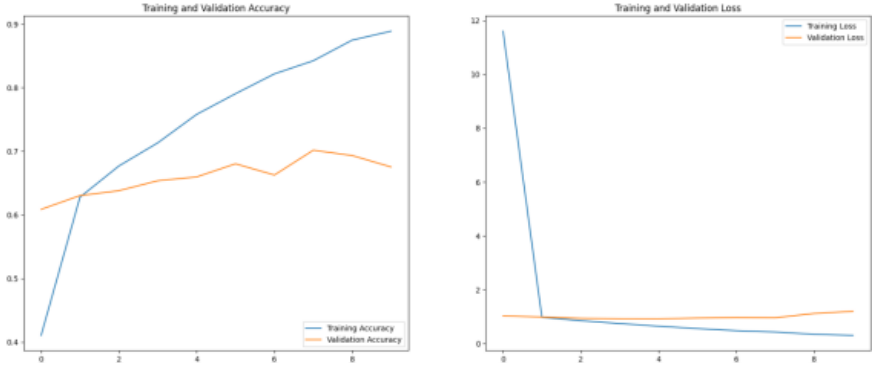
En el modelo de capas convolucionales le añadimos algo de complejidad, incorporando herramientas que permiten a la red un aprendizaje más acertado de las imágenes, como las convoluciones y el pooling. El último paso de la red es alimentar con los mapas de características obtenidos a un conjunto de redes densas para realizar la clasificación.

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 8)	224
activation (Activation)	(None, 224, 224, 8)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 8)	0
conv2d_1 (Conv2D)	(None, 112, 112, 16)	1,168
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 16)	0
activation_1 (Activation)	(None, 56, 56, 16)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 16)	0
conv2d_2 (Conv2D)	(None, 28, 28, 32)	4,640
activation_2 (Activation)	(None, 28, 28, 32)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dropout (Dropout)	(None, 6272)	0
dense_4 (Dense)	(None, 250)	1,568,250
dense_5 (Dense)	(None, 250)	62,750
dense_6 (Dense)	(None, 6)	1,506

```
Epoch 1/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 21s 110ms/step - accuracy: 0.2884 - loss: 38.2553 - val_accuracy: 0.6087 - val_loss: 1.0300
Epoch 2/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - accuracy: 0.6140 - loss: 1.0129 - val_accuracy: 0.6303 - val_loss: 0.9858
Epoch 3/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - accuracy: 0.6672 - loss: 0.8754 - val_accuracy: 0.6380 - val_loss: 0.9377
Epoch 4/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 53ms/step - accuracy: 0.7011 - loss: 0.7853 - val_accuracy: 0.6537 - val_loss: 0.9202
Epoch 5/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 53ms/step - accuracy: 0.7479 - loss: 0.6778 - val_accuracy: 0.6597 - val_loss: 0.9183
Epoch 6/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 53ms/step - accuracy: 0.7799 - loss: 0.5923 - val_accuracy: 0.6803 - val_loss: 0.9485
Epoch 7/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - accuracy: 0.8093 - loss: 0.5098 - val_accuracy: 0.6627 - val_loss: 0.9647
Epoch 8/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 54ms/step - accuracy: 0.8212 - loss: 0.4826 - val_accuracy: 0.7017 - val_loss: 0.9592
Epoch 9/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 53ms/step - accuracy: 0.8680 - loss: 0.3641 - val_accuracy: 0.6933 - val_loss: 1.1169
Epoch 10/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - accuracy: 0.8859 - loss: 0.3197 - val_accuracy: 0.6753 - val_loss: 1.1973
```

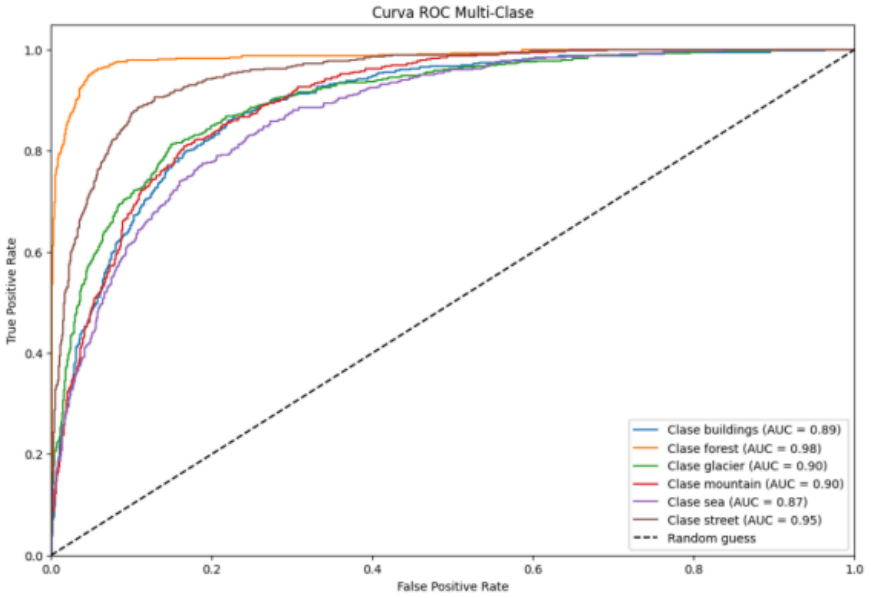
Resultados:

Con esta red obtuvimos mejores resultados, siendo la misma capaz de clasificar correctamente dos de tres imágenes aleatorias (de las cuales una de ellas es ambigua).



Curva ROC:

La curva ROC de este modelo muestra grandes avances, con una mejor clasificación de los edificios (‘buildings’) y donde la AUC de todas las clases está por encima del 0,8.



Modelo con bloques residuales identidad

La arquitectura de las redes residuales fue implementada inicialmente para evitar que las redes de profundidades significativas eviten su degradación característica de las grandes cantidades de capas y, de esa manera, la profundidad de la red deje de ser un problema para la misma.

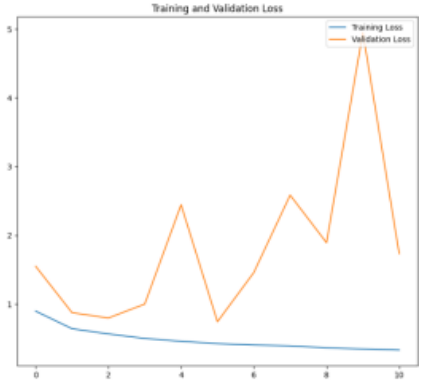
Es por este motivo que nuestra implementación de la red residual es el modelo más extenso que tenemos en cuanto a arquitectura.

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0	-
conv2d_5 (Conv2D)	(None, 224, 224, 32)	896	input_layer_2[0][0]
activation_5 (Activation)	(None, 224, 224, 32)	0	conv2d_5[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 74, 74, 32)	0	activation_5[0][0]
conv2d_7 (Conv2D)	(None, 74, 74, 64)	18,496	max_pooling2d_6[0][0]
batch_normalization_1 (BatchNormalization)	(None, 74, 74, 64)	256	conv2d_7[0][0]
activation_6 (Activation)	(None, 74, 74, 64)	0	batch_normalization_1…
conv2d_8 (Conv2D)	(None, 74, 74, 64)	36,928	activation_6[0][0]
conv2d_6 (Conv2D)	(None, 74, 74, 64)	2,112	max_pooling2d_6[0][0]
batch_normalization_2 (BatchNormalization)	(None, 74, 74, 64)	256	conv2d_8[0][0]
batch_normalization (BatchNormalization)	(None, 74, 74, 64)	256	conv2d_6[0][0]
add (Add)	(None, 74, 74, 64)	0	batch_normalization_2… batch_normalization[0…
activation_7 (Activation)	(None, 74, 74, 64)	0	add[0][0]
conv2d_9 (Conv2D)	(None, 74, 74, 64)	36,928	activation_7[0][0]
batch_normalization_3 (BatchNormalization)	(None, 74, 74, 64)	256	conv2d_9[0][0]
activation_8 (Activation)	(None, 74, 74, 64)	0	batch_normalization_3…
conv2d_10 (Conv2D)	(None, 74, 74, 64)	36,928	activation_8[0][0]
batch_normalization_4 (BatchNormalization)	(None, 74, 74, 64)	256	conv2d_10[0][0]
add_1 (Add)	(None, 74, 74, 64)	0	batch_normalization_4… activation_7[0][0]
activation_9 (Activation)	(None, 74, 74, 64)	0	add_1[0][0]
conv2d_12 (Conv2D)	(None, 37, 37, 128)	73,856	activation_9[0][0]
batch_normalization_6 (BatchNormalization)	(None, 37, 37, 128)	512	conv2d_12[0][0]
activation_10 (Activation)	(None, 37, 37, 128)	0	batch_normalization_6…
conv2d_13 (Conv2D)	(None, 37, 37, 128)	147,584	activation_10[0][0]
conv2d_11 (Conv2D)	(None, 37, 37, 128)	8,320	activation_9[0][0]
batch_normalization_7 (BatchNormalization)	(None, 37, 37, 128)	512	conv2d_13[0][0]
batch_normalization_5 (BatchNormalization)	(None, 37, 37, 128)	512	conv2d_11[0][0]
add_2 (Add)	(None, 37, 37, 128)	0	batch_normalization_7… batch_normalization_5…
activation_11 (Activation)	(None, 37, 37, 128)	0	add_2[0][0]
conv2d_14 (Conv2D)	(None, 37, 37, 128)	147,584	activation_11[0][0]
batch_normalization_8 (BatchNormalization)	(None, 37, 37, 128)	512	conv2d_14[0][0]
activation_12 (Activation)	(None, 37, 37, 128)	0	batch_normalization_8…
conv2d_15 (Conv2D)	(None, 37, 37, 128)	147,584	activation_12[0][0]
batch_normalization_9 (BatchNormalization)	(None, 37, 37, 128)	512	conv2d_15[0][0]
add_3 (Add)	(None, 37, 37, 128)	0	batch_normalization_9… activation_11[0][0]
activation_13 (Activation)	(None, 37, 37, 128)	0	add_3[0][0]
global_average_pooling2d (GlobalAveragePooling2D)	(None, 128)	0	activation_13[0][0]
dense_7 (Dense)	(None, 256)	33,024	global_average_poolin…
dense_8 (Dense)	(None, 256)	65,792	dense_7[0][0]
dense_9 (Dense)	(None, 256)	65,792	dense_8[0][0]
dense_10 (Dense)	(None, 6)	1,542	dense_9[0][0]

```
Epoch 1/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 99s 595ms/step - accuracy: 0.5613 - loss: 1.0840 - val_accuracy: 0.4590 - val_loss: 1.5480
Epoch 2/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 388ms/step - accuracy: 0.7483 - loss: 0.6642 - val_accuracy: 0.6540 - val_loss: 0.8734
Epoch 3/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 386ms/step - accuracy: 0.7742 - loss: 0.5883 - val_accuracy: 0.6843 - val_loss: 0.7956
Epoch 4/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8089 - loss: 0.5133 - val_accuracy: 0.6437 - val_loss: 0.9984
Epoch 5/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8289 - loss: 0.4622 - val_accuracy: 0.5103 - val_loss: 2.4431
Epoch 6/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8423 - loss: 0.4283 - val_accuracy: 0.7607 - val_loss: 0.7415
Epoch 7/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 386ms/step - accuracy: 0.8567 - loss: 0.3957 - val_accuracy: 0.5750 - val_loss: 1.4569
Epoch 8/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8581 - loss: 0.3878 - val_accuracy: 0.3817 - val_loss: 2.5833
Epoch 9/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8636 - loss: 0.3724 - val_accuracy: 0.5043 - val_loss: 1.8915
Epoch 10/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8782 - loss: 0.3386 - val_accuracy: 0.3167 - val_loss: 4.9485
Epoch 11/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 45s 387ms/step - accuracy: 0.8760 - loss: 0.3365 - val_accuracy: 0.5653 - val_loss: 1.7320
```

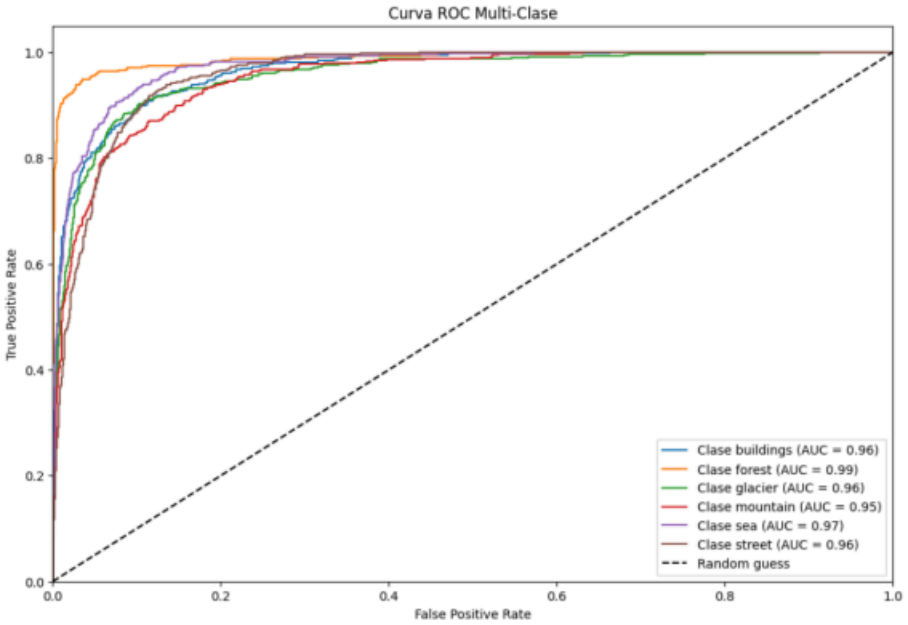
Resultados:

De manera similar al caso anterior, observamos una red con predicciones aceptables, en donde el error se da principalmente en imágenes ambiguas.



Curva ROC:

Las curvas ROC de este modelo son mejores que las del modelo convencional (las AUC de cada clase se encuentran por encima de 0,95, indicando un muy buen performance a la hora de clasificar).



Modelo con backbone

Por último tenemos un modelo realizado con transfer-learning. Utilizamos el modelo ResNet 50 para la realización de este backbone y el posterior entrenamiento sobre el dataset, obteniendo (como era de esperarse) los mejores resultados en la clasificación.

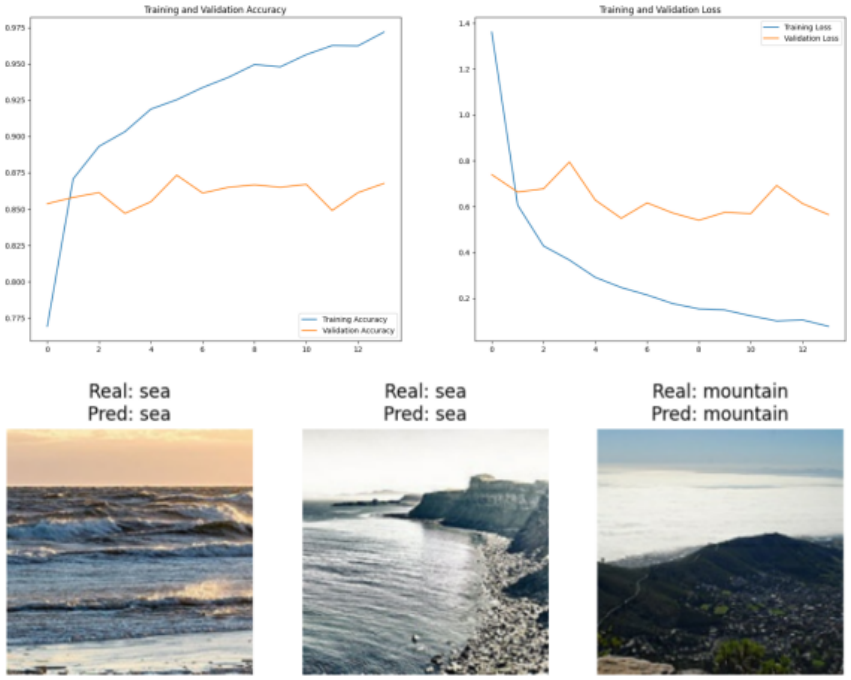
Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_max_pooling2d (GlobalMaxPooling2D)	(None, 2048)	0
dense_11 (Dense)	(None, 6)	12,294

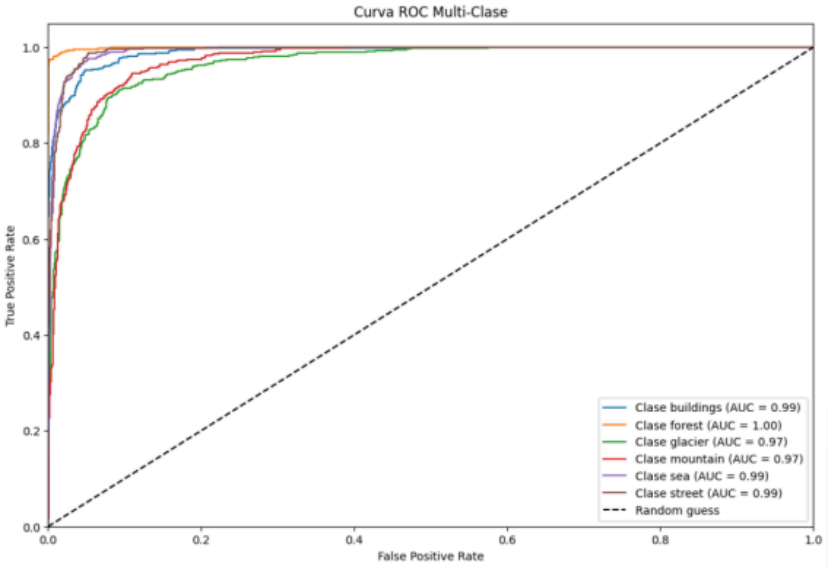
Total params: 23,624,596 (90.12 MB)
Trainable params: 12,294 (48.02 KB)
Non-trainable params: 23,587,712 (89.98 MB)
Optimizer params: 24,590 (96.06 KB)

```
Epoch 1/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 78s 506ms/step - accuracy: 0.6373 - loss: 2.5284 - val_accuracy: 0.8537 - val_loss: 0.7394
Epoch 2/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 367ms/step - accuracy: 0.8640 - loss: 0.6612 - val_accuracy: 0.8580 - val_loss: 0.6633
Epoch 3/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 366ms/step - accuracy: 0.8892 - loss: 0.4541 - val_accuracy: 0.8613 - val_loss: 0.6774
Epoch 4/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 42s 363ms/step - accuracy: 0.8980 - loss: 0.3838 - val_accuracy: 0.8470 - val_loss: 0.7943
Epoch 5/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 365ms/step - accuracy: 0.9148 - loss: 0.3296 - val_accuracy: 0.8550 - val_loss: 0.6280
Epoch 6/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 367ms/step - accuracy: 0.9246 - loss: 0.2463 - val_accuracy: 0.8733 - val_loss: 0.5488
Epoch 7/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 365ms/step - accuracy: 0.9331 - loss: 0.2144 - val_accuracy: 0.8610 - val_loss: 0.6159
Epoch 8/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 365ms/step - accuracy: 0.9395 - loss: 0.1757 - val_accuracy: 0.8650 - val_loss: 0.5715
Epoch 9/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 366ms/step - accuracy: 0.9459 - loss: 0.1641 - val_accuracy: 0.8667 - val_loss: 0.5401
Epoch 10/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 365ms/step - accuracy: 0.9476 - loss: 0.1475 - val_accuracy: 0.8650 - val_loss: 0.5748
Epoch 11/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 365ms/step - accuracy: 0.9561 - loss: 0.1231 - val_accuracy: 0.8670 - val_loss: 0.5692
Epoch 12/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 364ms/step - accuracy: 0.9644 - loss: 0.0967 - val_accuracy: 0.8490 - val_loss: 0.6917
Epoch 13/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 364ms/step - accuracy: 0.9640 - loss: 0.1030 - val_accuracy: 0.8613 - val_loss: 0.6130
Epoch 14/20
117/117 ━━━━━━━━━━━━━━━━━━━━ 43s 364ms/step - accuracy: 0.9741 - loss: 0.0734 - val_accuracy: 0.8677 - val_loss: 0.5648
```

Resultados:
Los resultados nos dieron una pérdida muy baja y un accuracy casi perfecto. En cuanto a las predicciones aleatorias, con este modelo logramos una predicción perfecta en la mayoría de los intentos, algo que no sucedía con los anteriores.



Curva ROC:
Como se mencionó anteriormente, este modelo fue el que realizó mejores clasificaciones, y esto se ve reflejado en las AUC de cada clase, estando todas cercanas a 1 (clasificación perfecta).



Conclusiones:
Abordando el problema 3 del práctico pudimos observar claramente la diferencia de las distintas arquitecturas de redes neuronales para la tarea de clasificación, entendiendo la importancia de elegir la arquitectura correcta para cada caso así como las distintas técnicas que existen para la creación de una arquitectura personalizada.