modelS_2loss_2optimizers

June 12, 2025

0.1 Importação de bibliotecas

[1]:
```
from tensorflow import keras
from keras import layers
from keras.preprocessing import image_dataset_from_directory
import matplotlib.pyplot as plt
from keras.utils import to_categorical
import tensorflow as tf
import numpy as np
from keras.preprocessing import image
from sklearn.metrics import classification_report
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix
import os, shutil
```

```
2025-06-12 20:06:21.497301: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1749755181.535213     858 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1749755181.543829     858 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1749755181.575946     858 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749755181.575973     858 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749755181.575976     858 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749755181.575978     858 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
```

```
target more than once.
2025-06-12 20:06:21.584765: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
```

0.2 Funções

[2]:
```python
def get_true_pred(model, dataset):
    y_true = []
    y_pred = []
    for images, labels in dataset.unbatch().batch(1):
        y_true.append(np.argmax(labels.numpy()))
        pred = model.predict(images, verbose=0)
        y_pred.append(np.argmax(pred))
    return np.array(y_true), np.array(y_pred)
```

0.3 Carregamento do dataset

Carrega o dataset distribuido pelos diferentes conjuntos de dados.

[3]:
```python
train_dir = 'Dataset/archive/seg_train'
validation_dir = 'Dataset/archive/seg_val'
test_dir = 'Dataset/archive/seg_test'

train_buildings_dir = 'Dataset/archive/seg_train/buildings/'
train_forest_dir = 'Dataset/archive/seg_train/forest'
train_glacier_dir = 'Dataset/archive/seg_train/glacier'
train_mountain_dir = 'Dataset/archive/seg_train/mountain'
train_sea_dir = 'Dataset/archive/seg_train/sea'
train_street_dir = 'Dataset/archive/seg_train/street'

val_buildings_dir = 'Dataset/archive/seg_val/buildings'
val_forest_dir = 'Dataset/archive/seg_val/forest'
val_glacier_dir = 'Dataset/archive/seg_val/glacier'
val_mountain_dir = 'Dataset/archive/seg_val/mountain'
val_sea_dir = 'Dataset/archive/seg_val/sea'
val_street_dir = 'Dataset/archive/seg_val/street'

test_buildings_dir = 'Dataset/archive/seg_test/buildings'
test_forest_dir = 'Dataset/archive/seg_test/forest'
test_glacier_dir = 'Dataset/archive/seg_test/glacier'
test_mountain_dir = 'Dataset/archive/seg_test/mountain'
test_sea_dir = 'Dataset/archive/seg_test/sea'
test_street_dir = 'Dataset/archive/seg_test/street'

print('total training buildings images:', len(os.listdir(train_buildings_dir)))
```

```
print('total training forest images:', len(os.listdir(train_forest_dir)))
print('total training glacier images:', len(os.listdir(train_glacier_dir)))
print('total training mountain images:', len(os.listdir(train_mountain_dir)))
print('total training sea images:', len(os.listdir(train_sea_dir)))
print('total training street images:', len(os.listdir(train_street_dir)))

print('total validation buildings images:', len(os.listdir(val_buildings_dir)))
print('total validation forest images:', len(os.listdir(val_forest_dir)))
print('total validation glacier images:', len(os.listdir(val_glacier_dir)))
print('total validation mountain images:', len(os.listdir(val_mountain_dir)))
print('total validation sea images:', len(os.listdir(val_sea_dir)))
print('total validation street images:', len(os.listdir(val_street_dir)))

print('total test buildings images:', len(os.listdir(test_buildings_dir)))
print('total test forest images:', len(os.listdir(test_forest_dir)))
print('total test glacier images:', len(os.listdir(test_glacier_dir)))
print('total test mountain images:', len(os.listdir(test_mountain_dir)))
print('total test sea images:', len(os.listdir(test_sea_dir)))
print('total test street images:', len(os.listdir(test_street_dir)))
```

```
total training buildings images: 1691
total training forest images: 1771
total training glacier images: 1904
total training mountain images: 2012
total training sea images: 1774
total training street images: 1882
total validation buildings images: 500
total validation forest images: 500
total validation glacier images: 500
total validation mountain images: 500
total validation sea images: 500
total validation street images: 500
total test buildings images: 437
total test forest images: 474
total test glacier images: 553
total test mountain images: 525
total test sea images: 510
total test street images: 501
```

0.4 Distribuição de imagens por classe e por conjunto de dados

As imagens estão distribuidas por 3 conjuntos de dados: train, validation e test. Cada um desses conjuntos está distribuido por 6 classes: buildings, forest, glacier, mountain, sea e street.

0.4.1 Número total de imagens por classe:

Classe	Treino	Validação	Teste	Total
Buildings	1691	500	437	2628
Forest	1771	500	474	2745
Glacier	1904	500	553	2957
Mountain	2012	500	525	3037
Sea	1774	500	510	2784
Street	1882	500	501	2883
Total	**11034**	**3000**	**3000**	**17034**

0.4.2 Número total de imagens por conjunto de dados:

Conjunto de dados	Total
Treino	11034
Validação	3000
Teste	3000
Total geral	**17034**

1 Processamento dos dados

Carrega, redimensiona e organiza imagens em batches com rótulos one-hot, preparando os dados de treino, validação e teste.

[4]:
```
IMG_SIZE = 150
BATCH_SIZE = 32

# Processing the data
train_dataset = image_dataset_from_directory(
    train_dir,
    label_mode='categorical',
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)

validation_dataset = image_dataset_from_directory(
    validation_dir,
    label_mode='categorical',
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)

test_dataset = image_dataset_from_directory(
    test_dir,
    label_mode='categorical',
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE)

print(test_dataset)
```

```
class_names = train_dataset.class_names
print("Classes:", class_names)
```

```
Found 11034 files belonging to 6 classes.
```

```
I0000 00:00:1749755266.582247     858 gpu_device.cc:2019] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 6542 MB memory:  -> device: 0,
name: NVIDIA GeForce GTX 1070, pci bus id: 0000:01:00.0, compute capability: 6.1
```

```
Found 3000 files belonging to 6 classes.
Found 3000 files belonging to 6 classes.
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 150, 150, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(None, 6), dtype=tf.float32,
name=None))>
Classes: ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
```

2 Modelo (loss: categorical_crossentropy, optimizer: RMSprop)

2.1 Criação da CNN

Criação da CNN que irá receber imagens de 150x150 píxeis, aplica normalização e passa por quatro camadas convolucionais com max pooling para extrair características, seguidas de uma camada densa com 512 unidades e uma camada de saída softmax para classificação.

[18]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

print(model.summary())
```

```
Model: "functional_2"
```

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 150, 150, 3)	0
rescaling_1 (Rescaling)	(None, 150, 150, 3)	0

```
conv2d_4 (Conv2D)               (None, 148, 148, 32)          896

max_pooling2d_4 (MaxPooling2D)  (None, 74, 74, 32)            0

conv2d_5 (Conv2D)               (None, 72, 72, 64)          18,496

max_pooling2d_5 (MaxPooling2D)  (None, 36, 36, 64)            0

conv2d_6 (Conv2D)               (None, 34, 34, 128)         73,856

max_pooling2d_6 (MaxPooling2D)  (None, 17, 17, 128)           0

conv2d_7 (Conv2D)               (None, 15, 15, 128)        147,584

max_pooling2d_7 (MaxPooling2D)  (None, 7, 7, 128)             0

flatten_1 (Flatten)             (None, 6272)                  0

dense_2 (Dense)                 (None, 512)             3,211,776

dense_3 (Dense)                 (None, 6)                   3,078


Total params: 3,455,686 (13.18 MB)

Trainable params: 3,455,686 (13.18 MB)

Non-trainable params: 0 (0.00 B)

None
```

2.2 Compilação da CNN

Compilação da CNN utilizando a loss **categorical crossentropy** e o optimizer **RMSprop**.

[6]:
```
model.compile(
loss='categorical_crossentropy',
optimizer=tf.keras.optimizers.RMSprop(learning_rate=1e-4),
metrics=['acc'])
```

2.3 Definição do callback

Definição de um callback que guarda automaticamente o modelo com a menor perda (loss) de validação durante o treino.

[7]:
```
checkpoint_filepath = 'modelS_CatCross_RMS.keras'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss',
    save_best_only=True)
```

2.4 Treino da CNN

Treino da CNN durante 50 épocas utilizando o dataset de validação e o callback para guardar o melhor modelo.

[8]:
```
history_CatCross_RMS = model.fit(
train_dataset,
epochs=50,
validation_data=validation_dataset,
callbacks=[model_checkpoint_callback])
```

```
Epoch 1/50

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1749755408.539309    1346 service.cc:152] XLA service 0x78e8400049f0
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1749755408.539356    1346 service.cc:160]   StreamExecutor device
(0): NVIDIA GeForce GTX 1070, Compute Capability 6.1
2025-06-12 20:10:08.628419: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1749755408.886822    1346 cuda_dnn.cc:529] Loaded cuDNN version
90300
2025-06-12 20:10:14.061045: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.12 = (f32[32,32,148,148]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,3,150,150]{3,2,1,0} %bitcast.2658, f32[32,3,3,3]{3,2,1,0}
%bitcast.2459, f32[32]{0} %bitcast.3030), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:14.120199: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.13 = (f32[32,64,72,72]{3,2,1,0}, u8[0]{0}) custom-
```

```
call(f32[32,32,74,74]{3,2,1,0} %bitcast.3085, f32[64,32,3,3]{3,2,1,0}
%bitcast.2480, f32[64]{0} %bitcast.3125), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1_2/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:14.327237: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.14 = (f32[32,128,34,34]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,64,36,36]{3,2,1,0} %bitcast.3176, f32[128,64,3,3]{3,2,1,0}
%bitcast.2499, f32[128]{0} %bitcast.3216), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_2_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:14.514173: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.15 = (f32[32,128,15,15]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,128,17,17]{3,2,1,0} %bitcast.3267, f32[128,128,3,3]{3,2,1,0}
%bitcast.2518, f32[128]{0} %bitcast.3307), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_3_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}

  5/345            9s 29ms/step - acc: 0.1741
- loss: 1.7926

I0000 00:00:1749755417.195745    1346 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

343/345            0s 30ms/step -
acc: 0.4793 - loss: 1.3048

2025-06-12 20:10:28.002427: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
```

Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-activation.12 = (f32[26,32,148,148]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,3,150,150]{3,2,1,0} %bitcast.2658, f32[32,3,3,3]{3,2,1,0} %bitcast.2459, f32[32]{0} %bitcast.3030), window={size=3x3}, dim_labels=bf01_oi01->bf01, custom_call_target="__cudnn$convBiasActivationForward", metadata={op_type="Conv2D" op_name="functional_1/conv2d_1/convolution" source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_config":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:28.052716: I external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549] Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-activation.13 = (f32[26,64,72,72]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,32,74,74]{3,2,1,0} %bitcast.3085, f32[64,32,3,3]{3,2,1,0} %bitcast.2480, f32[64]{0} %bitcast.3125), window={size=3x3}, dim_labels=bf01_oi01->bf01, custom_call_target="__cudnn$convBiasActivationForward", metadata={op_type="Conv2D" op_name="functional_1/conv2d_1_2/convolution" source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_config":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:28.231026: I external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549] Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-activation.14 = (f32[26,128,34,34]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,64,36,36]{3,2,1,0} %bitcast.3176, f32[128,64,3,3]{3,2,1,0} %bitcast.2499, f32[128]{0} %bitcast.3216), window={size=3x3}, dim_labels=bf01_oi01->bf01, custom_call_target="__cudnn$convBiasActivationForward", metadata={op_type="Conv2D" op_name="functional_1/conv2d_2_1/convolution" source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_config":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyrelu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:28.370095: I external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549] Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-activation.15 = (f32[26,128,15,15]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,128,17,17]{3,2,1,0} %bitcast.3267, f32[128,128,3,3]{3,2,1,0} %bitcast.2518, f32[128]{0} %bitcast.3307), window={size=3x3}, dim_labels=bf01_oi01->bf01, custom_call_target="__cudnn$convBiasActivationForward",

metadata={op_type="Conv2D" op_name="functional_1/conv2d_3_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kNone","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}

345/345 0s 39ms/step -
acc: 0.4797 - loss: 1.3040

2025-06-12 20:10:31.220247: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.12 = (f32[32,32,148,148]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,3,150,150]{3,2,1,0} %bitcast.569, f32[32,3,3,3]{3,2,1,0}
%bitcast.576, f32[32]{0} %bitcast.578), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:31.301916: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.13 = (f32[32,64,72,72]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,32,74,74]{3,2,1,0} %bitcast.586, f32[64,32,3,3]{3,2,1,0}
%bitcast.593, f32[64]{0} %bitcast.595), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1_2/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:31.511840: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.14 = (f32[32,128,34,34]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,64,36,36]{3,2,1,0} %bitcast.601, f32[128,64,3,3]{3,2,1,0}
%bitcast.608, f32[128]{0} %bitcast.610), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_2_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={

"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:31.693318: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.15 = (f32[32,128,15,15]{3,2,1,0}, u8[0]{0}) custom-
call(f32[32,128,17,17]{3,2,1,0} %bitcast.616, f32[128,128,3,3]{3,2,1,0}
%bitcast.623, f32[128]{0} %bitcast.625), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_3_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:36.686971: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.12 = (f32[24,32,148,148]{3,2,1,0}, u8[0]{0}) custom-
call(f32[24,3,150,150]{3,2,1,0} %bitcast.569, f32[32,3,3,3]{3,2,1,0}
%bitcast.576, f32[32]{0} %bitcast.578), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:36.732031: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.13 = (f32[24,64,72,72]{3,2,1,0}, u8[0]{0}) custom-
call(f32[24,32,74,74]{3,2,1,0} %bitcast.586, f32[64,32,3,3]{3,2,1,0}
%bitcast.593, f32[64]{0} %bitcast.595), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1_2/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:36.885420: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-

```
activation.14 = (f32[24,128,34,34]{3,2,1,0}, u8[0]{0}) custom-
call(f32[24,64,36,36]{3,2,1,0} %bitcast.601, f32[128,64,3,3]{3,2,1,0}
%bitcast.608, f32[128]{0} %bitcast.610), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_2_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:10:37.034066: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.15 = (f32[24,128,15,15]{3,2,1,0}, u8[0]{0}) custom-
call(f32[24,128,17,17]{3,2,1,0} %bitcast.616, f32[128,128,3,3]{3,2,1,0}
%bitcast.623, f32[128]{0} %bitcast.625), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_3_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}

345/345            32s 60ms/step -
acc: 0.4799 - loss: 1.3035 - val_acc: 0.6147 - val_loss: 1.0001
Epoch 2/50
345/345            11s 33ms/step -
acc: 0.6378 - loss: 0.9455 - val_acc: 0.6273 - val_loss: 0.9433
Epoch 3/50
345/345            11s 32ms/step -
acc: 0.6909 - loss: 0.8272 - val_acc: 0.7133 - val_loss: 0.7712
Epoch 4/50
345/345            14s 40ms/step -
acc: 0.7273 - loss: 0.7409 - val_acc: 0.7127 - val_loss: 0.7510
Epoch 5/50
345/345            11s 31ms/step -
acc: 0.7502 - loss: 0.6786 - val_acc: 0.5873 - val_loss: 1.0899
Epoch 6/50
345/345            11s 32ms/step -
acc: 0.7675 - loss: 0.6406 - val_acc: 0.7593 - val_loss: 0.6479
Epoch 7/50
345/345            14s 41ms/step -
acc: 0.7861 - loss: 0.5857 - val_acc: 0.7723 - val_loss: 0.6296
Epoch 8/50
345/345            11s 32ms/step -
```

```
acc: 0.8000 - loss: 0.5504 - val_acc: 0.7933 - val_loss: 0.5771
Epoch 9/50
345/345             11s 32ms/step -
acc: 0.8089 - loss: 0.5090 - val_acc: 0.7880 - val_loss: 0.6025
Epoch 10/50
345/345             14s 41ms/step -
acc: 0.8282 - loss: 0.4808 - val_acc: 0.8070 - val_loss: 0.5311
Epoch 11/50
345/345             11s 31ms/step -
acc: 0.8384 - loss: 0.4425 - val_acc: 0.8063 - val_loss: 0.5388
Epoch 12/50
345/345             11s 32ms/step -
acc: 0.8510 - loss: 0.4095 - val_acc: 0.8173 - val_loss: 0.5204
Epoch 13/50
345/345             14s 41ms/step -
acc: 0.8649 - loss: 0.3892 - val_acc: 0.8210 - val_loss: 0.5094
Epoch 14/50
345/345             11s 33ms/step -
acc: 0.8702 - loss: 0.3635 - val_acc: 0.8250 - val_loss: 0.5011
Epoch 15/50
345/345             11s 31ms/step -
acc: 0.8802 - loss: 0.3297 - val_acc: 0.8277 - val_loss: 0.5056
Epoch 16/50
345/345             14s 40ms/step -
acc: 0.8881 - loss: 0.3096 - val_acc: 0.8220 - val_loss: 0.5235
Epoch 17/50
345/345             11s 31ms/step -
acc: 0.8963 - loss: 0.2870 - val_acc: 0.8223 - val_loss: 0.5169
Epoch 18/50
345/345             11s 31ms/step -
acc: 0.9104 - loss: 0.2554 - val_acc: 0.8317 - val_loss: 0.5126
Epoch 19/50
345/345             14s 40ms/step -
acc: 0.9138 - loss: 0.2410 - val_acc: 0.8317 - val_loss: 0.5212
Epoch 20/50
345/345             11s 32ms/step -
acc: 0.9273 - loss: 0.2118 - val_acc: 0.8260 - val_loss: 0.5310
Epoch 21/50
345/345             11s 32ms/step -
acc: 0.9305 - loss: 0.1972 - val_acc: 0.8223 - val_loss: 0.5691
Epoch 22/50
345/345             14s 40ms/step -
acc: 0.9422 - loss: 0.1724 - val_acc: 0.7290 - val_loss: 1.1331
Epoch 23/50
345/345             11s 31ms/step -
acc: 0.9468 - loss: 0.1567 - val_acc: 0.7937 - val_loss: 0.8158
Epoch 24/50
345/345             11s 31ms/step -
```

```
acc: 0.9520 - loss: 0.1341 - val_acc: 0.8207 - val_loss: 0.6506
Epoch 25/50
345/345             14s 39ms/step -
acc: 0.9644 - loss: 0.1092 - val_acc: 0.8067 - val_loss: 0.7928
Epoch 26/50
345/345             11s 31ms/step -
acc: 0.9647 - loss: 0.1049 - val_acc: 0.8173 - val_loss: 0.7138
Epoch 27/50
345/345             11s 31ms/step -
acc: 0.9708 - loss: 0.0910 - val_acc: 0.8153 - val_loss: 0.7679
Epoch 28/50
345/345             14s 39ms/step -
acc: 0.9751 - loss: 0.0828 - val_acc: 0.8250 - val_loss: 0.7370
Epoch 29/50
345/345             11s 31ms/step -
acc: 0.9806 - loss: 0.0648 - val_acc: 0.7557 - val_loss: 1.2384
Epoch 30/50
345/345             11s 31ms/step -
acc: 0.9834 - loss: 0.0623 - val_acc: 0.8187 - val_loss: 0.8667
Epoch 31/50
345/345             14s 39ms/step -
acc: 0.9853 - loss: 0.0517 - val_acc: 0.8230 - val_loss: 0.8482
Epoch 32/50
345/345             12s 33ms/step -
acc: 0.9859 - loss: 0.0532 - val_acc: 0.8217 - val_loss: 0.8822
Epoch 33/50
345/345             11s 33ms/step -
acc: 0.9900 - loss: 0.0361 - val_acc: 0.8210 - val_loss: 0.9564
Epoch 34/50
345/345             14s 40ms/step -
acc: 0.9877 - loss: 0.0398 - val_acc: 0.8127 - val_loss: 0.9933
Epoch 35/50
345/345             11s 31ms/step -
acc: 0.9909 - loss: 0.0311 - val_acc: 0.8177 - val_loss: 0.9931
Epoch 36/50
345/345             11s 31ms/step -
acc: 0.9918 - loss: 0.0280 - val_acc: 0.8157 - val_loss: 1.0464
Epoch 37/50
345/345             14s 39ms/step -
acc: 0.9910 - loss: 0.0309 - val_acc: 0.8250 - val_loss: 0.9815
Epoch 38/50
345/345             11s 31ms/step -
acc: 0.9923 - loss: 0.0244 - val_acc: 0.8230 - val_loss: 1.0329
Epoch 39/50
345/345             11s 31ms/step -
acc: 0.9898 - loss: 0.0302 - val_acc: 0.8093 - val_loss: 1.1644
Epoch 40/50
345/345             14s 39ms/step -
```

```
acc: 0.9935 - loss: 0.0317 - val_acc: 0.8297 - val_loss: 1.0730
Epoch 41/50
345/345             11s 31ms/step -
acc: 0.9935 - loss: 0.0218 - val_acc: 0.8017 - val_loss: 1.3321
Epoch 42/50
345/345             11s 31ms/step -
acc: 0.9927 - loss: 0.0280 - val_acc: 0.8277 - val_loss: 1.1037
Epoch 43/50
345/345             14s 39ms/step -
acc: 0.9944 - loss: 0.0182 - val_acc: 0.8173 - val_loss: 1.1486
Epoch 44/50
345/345             11s 31ms/step -
acc: 0.9948 - loss: 0.0158 - val_acc: 0.8313 - val_loss: 1.1271
Epoch 45/50
345/345             11s 31ms/step -
acc: 0.9966 - loss: 0.0117 - val_acc: 0.8237 - val_loss: 1.2193
Epoch 46/50
345/345             14s 39ms/step -
acc: 0.9939 - loss: 0.0221 - val_acc: 0.8010 - val_loss: 1.3957
Epoch 47/50
345/345             11s 31ms/step -
acc: 0.9960 - loss: 0.0113 - val_acc: 0.8170 - val_loss: 1.2861
Epoch 48/50
345/345             11s 31ms/step -
acc: 0.9962 - loss: 0.0149 - val_acc: 0.8213 - val_loss: 1.2919
Epoch 49/50
345/345             14s 39ms/step -
acc: 0.9959 - loss: 0.0138 - val_acc: 0.8187 - val_loss: 1.3313
Epoch 50/50
345/345             11s 31ms/step -
acc: 0.9960 - loss: 0.0111 - val_acc: 0.8187 - val_loss: 1.3664
```

[9]:
```
best_epoch = np.argmin(history_CatCross_RMS.history['val_loss']) + 1
print(f"Melhor época (menor val_loss): {best_epoch}")
```

```
Melhor época (menor val_loss): 14
```

2.5 Carregamento do modelo e validação

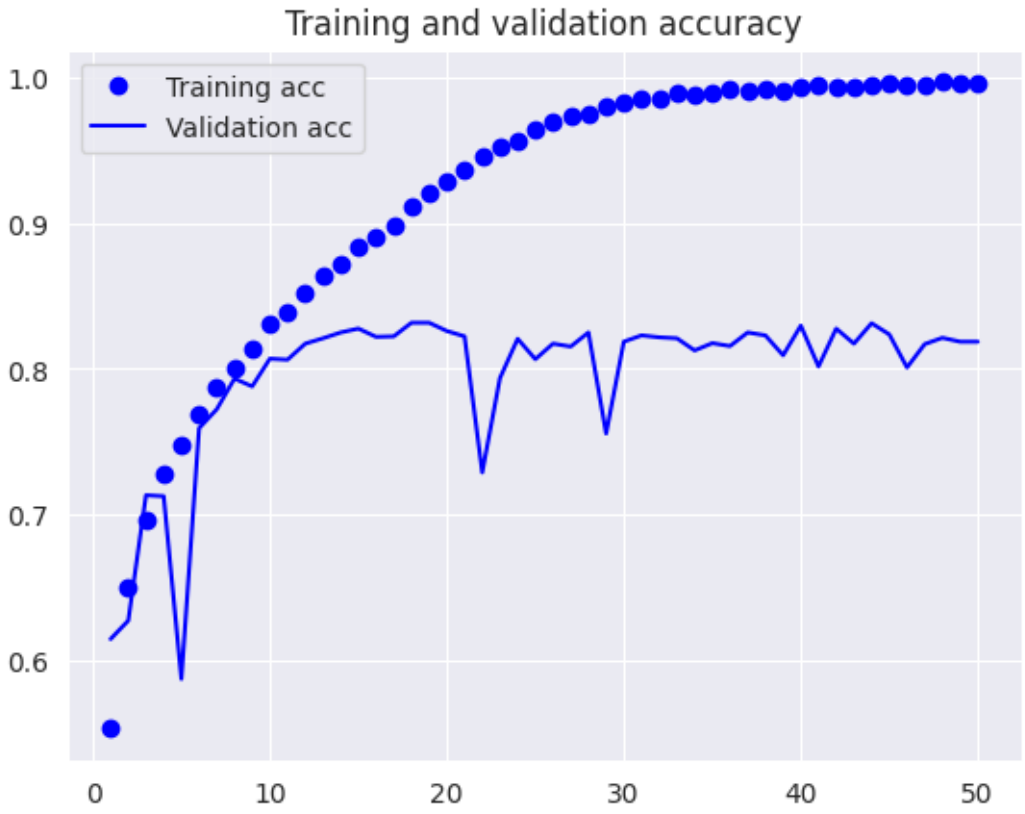
Carregamento e avaliação do modelo através do valor da accuracy.

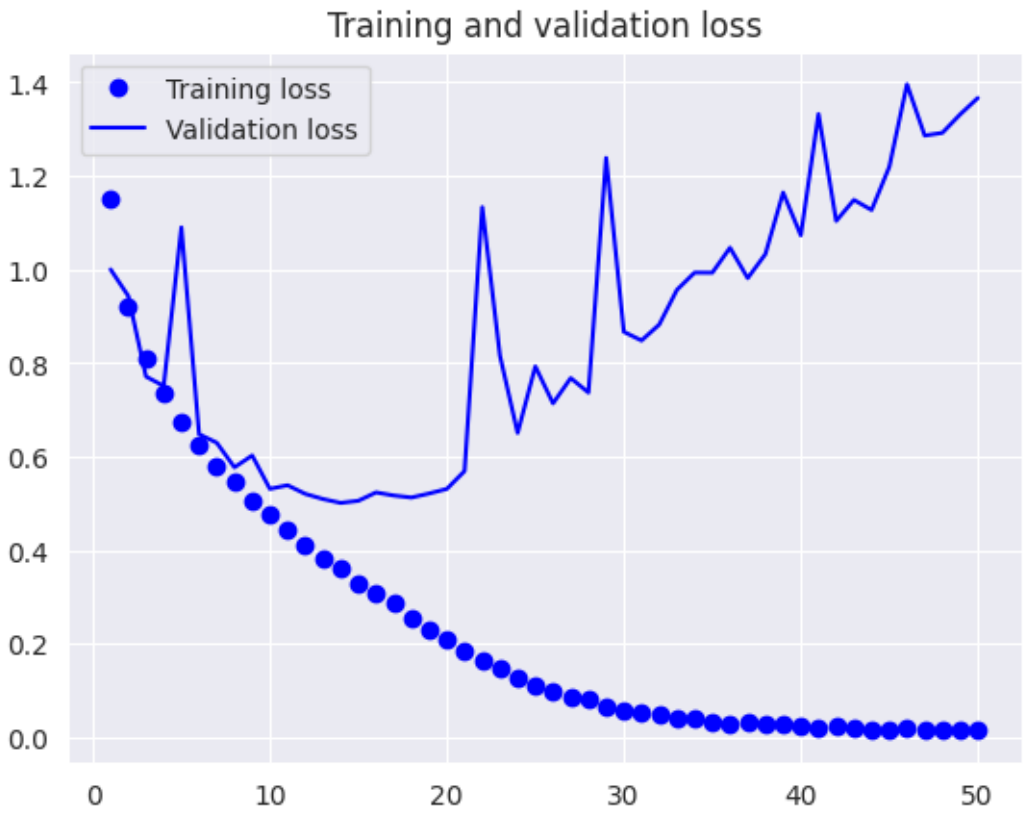
[10]:
```
modelS_CatCross_RMS = keras.models.load_model('modelS_CatCross_RMS.keras')
val_loss, val_acc = modelS_CatCross_RMS.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

```
94/94             3s 19ms/step - acc:
0.8345 - loss: 0.4762
val_acc: 0.824999988079071
```

Representação gráfica dos valores da accuracy e da loss ao longo das épocas.

[11]:
```
acc = history_CatCross_RMS.history['acc']
val_acc = history_CatCross_RMS.history['val_acc']
loss = history_CatCross_RMS.history['loss']
val_loss = history_CatCross_RMS.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





Avaliação da performance do modelo no conjunto de teste, utilizando o relatório de classificação. O relatório apresenta, para cada classe, as métricas precision, recall e F1-score, permitindo analisar detalhadamente os acertos e erros por classe.

[12]:
```
y_true, y_pred = get_true_pred(modelS_CatCross_RMS, test_dataset)
report = classification_report(y_true, y_pred, target_names=class_names,␣
 ↪output_dict=True)
class_only_report = {k: v for k, v in report.items() if k in class_names}
df = pd.DataFrame(class_only_report).T
print(df[['precision', 'recall', 'f1-score']].round(3))
```

```
2025-06-12 20:22:12.173160: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.12 = (f32[1,32,148,148]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,3,150,150]{3,2,1,0} %bitcast.262, f32[32,3,3,3]{3,2,1,0}
%bitcast.269, f32[32]{0} %bitcast.271), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
```

"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:22:12.250748: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.13 = (f32[1,64,72,72]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,32,74,74]{3,2,1,0} %bitcast.278, f32[64,32,3,3]{3,2,1,0}
%bitcast.285, f32[64]{0} %bitcast.287), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_1_2/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:22:12.354361: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.14 = (f32[1,128,34,34]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,64,36,36]{3,2,1,0} %bitcast.293, f32[128,64,3,3]{3,2,1,0}
%bitcast.300, f32[128]{0} %bitcast.302), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_2_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 20:22:12.451257: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.15 = (f32[1,128,15,15]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,128,17,17]{3,2,1,0} %bitcast.308, f32[128,128,3,3]{3,2,1,0}
%bitcast.315, f32[128]{0} %bitcast.317), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1/conv2d_3_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}

	precision	recall	f1-score
buildings	0.778	0.828	0.803

```
forest          0.959    0.930     0.944
glacier         0.846    0.714     0.775
mountain        0.798    0.792     0.795
sea             0.809    0.849     0.829
street          0.803    0.884     0.841
```

```
2025-06-12 20:25:36.900883: I tensorflow/core/framework/local_rendezvous.cc:407]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

3 Modelo (loss: KLDivergence, optimizer: SGD)

3.1 Criação da CNN

Criação da CNN que irá receber imagens de 150x150 píxeis, aplica normalização e passa por quatro camadas convolucionais com max pooling para extrair características, seguidas de uma camada densa com 512 unidades e uma camada de saída softmax para classificação.

[19]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

print(model.summary())
```

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 150, 150, 3)	0
rescaling_1 (Rescaling)	(None, 150, 150, 3)	0
conv2d_4 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_5 (Conv2D)	(None, 72, 72, 64)	18,496

```
 max_pooling2d_5 (MaxPooling2D)    (None, 36, 36, 64)                0

 conv2d_6 (Conv2D)                 (None, 34, 34, 128)           73,856

 max_pooling2d_6 (MaxPooling2D)    (None, 17, 17, 128)               0

 conv2d_7 (Conv2D)                 (None, 15, 15, 128)          147,584

 max_pooling2d_7 (MaxPooling2D)    (None, 7, 7, 128)                 0

 flatten_1 (Flatten)               (None, 6272)                      0

 dense_2 (Dense)                   (None, 512)                3,211,776

 dense_3 (Dense)                   (None, 6)                      3,078


Total params: 3,455,686 (13.18 MB)

Trainable params: 3,455,686 (13.18 MB)

Non-trainable params: 0 (0.00 B)

None
```

3.2 Compilação da CNN

Compilação da CNN utilizando a loss **KLDivergence** e o optimizer **SGD**.

[22]:
```
model.compile(
loss=tf.keras.losses.KLDivergence(),
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01),
metrics=['acc'])
```

3.3 Definição do callback

Definição de um callback que guarda automaticamente o modelo com a menor perda (loss) de validação durante o treino.

[23]:
```
checkpoint_filepath = 'modelS_KLD_SGD.keras'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss',
    save_best_only=True)
```

3.4 Treino da CNN

Treino da CNN durante 50 épocas utilizando o dataset de validação e o callback para guardar o melhor modelo.

[24]:
```
history_KLD_SGD = model.fit(
train_dataset,
epochs=50,
validation_data=validation_dataset,
callbacks=[model_checkpoint_callback])
```

```
Epoch 1/50
345/345             23s 58ms/step -
acc: 0.2734 - loss: 1.6998 - val_acc: 0.5510 - val_loss: 1.1666
Epoch 2/50
345/345             13s 36ms/step -
acc: 0.5540 - loss: 1.1383 - val_acc: 0.5503 - val_loss: 1.1265
Epoch 3/50
345/345             18s 51ms/step -
acc: 0.5951 - loss: 1.0223 - val_acc: 0.5517 - val_loss: 1.1783
Epoch 4/50
345/345             13s 36ms/step -
acc: 0.6254 - loss: 0.9684 - val_acc: 0.6033 - val_loss: 0.9922
Epoch 5/50
345/345             18s 51ms/step -
acc: 0.6522 - loss: 0.9054 - val_acc: 0.6173 - val_loss: 0.9913
Epoch 6/50
345/345             13s 36ms/step -
acc: 0.6746 - loss: 0.8651 - val_acc: 0.6667 - val_loss: 0.8426
Epoch 7/50
345/345             15s 43ms/step -
acc: 0.6883 - loss: 0.8153 - val_acc: 0.6953 - val_loss: 0.7875
Epoch 8/50
345/345             15s 45ms/step -
acc: 0.7067 - loss: 0.7811 - val_acc: 0.7007 - val_loss: 0.7920
Epoch 9/50
345/345             15s 42ms/step -
acc: 0.7281 - loss: 0.7337 - val_acc: 0.7297 - val_loss: 0.7190
Epoch 10/50
345/345             16s 45ms/step -
acc: 0.7335 - loss: 0.7063 - val_acc: 0.6907 - val_loss: 0.7914
Epoch 11/50
345/345             12s 36ms/step -
acc: 0.7546 - loss: 0.6710 - val_acc: 0.7057 - val_loss: 0.7903
Epoch 12/50
345/345             15s 42ms/step -
acc: 0.7693 - loss: 0.6299 - val_acc: 0.7143 - val_loss: 0.7497
Epoch 13/50
345/345             15s 45ms/step -
```

```
acc: 0.7765 - loss: 0.6087 - val_acc: 0.7107 - val_loss: 0.8038
Epoch 14/50
345/345             15s 43ms/step -
acc: 0.7878 - loss: 0.5744 - val_acc: 0.7243 - val_loss: 0.7106
Epoch 15/50
345/345             15s 45ms/step -
acc: 0.8027 - loss: 0.5369 - val_acc: 0.7737 - val_loss: 0.6126
Epoch 16/50
345/345             15s 43ms/step -
acc: 0.8192 - loss: 0.5055 - val_acc: 0.7527 - val_loss: 0.6946
Epoch 17/50
345/345             15s 44ms/step -
acc: 0.8295 - loss: 0.4680 - val_acc: 0.7340 - val_loss: 0.7086
Epoch 18/50
345/345             15s 43ms/step -
acc: 0.8392 - loss: 0.4406 - val_acc: 0.7810 - val_loss: 0.5993
Epoch 19/50
345/345             12s 36ms/step -
acc: 0.8525 - loss: 0.4081 - val_acc: 0.7920 - val_loss: 0.6009
Epoch 20/50
345/345             18s 51ms/step -
acc: 0.8643 - loss: 0.3793 - val_acc: 0.7820 - val_loss: 0.6240
Epoch 21/50
345/345             13s 36ms/step -
acc: 0.8805 - loss: 0.3434 - val_acc: 0.7807 - val_loss: 0.6528
Epoch 22/50
345/345             18s 52ms/step -
acc: 0.8928 - loss: 0.3116 - val_acc: 0.7893 - val_loss: 0.6325
Epoch 23/50
345/345             13s 36ms/step -
acc: 0.9108 - loss: 0.2693 - val_acc: 0.7313 - val_loss: 0.8749
Epoch 24/50
345/345             16s 46ms/step -
acc: 0.9153 - loss: 0.2499 - val_acc: 0.7850 - val_loss: 0.6652
Epoch 25/50
345/345             14s 41ms/step -
acc: 0.9293 - loss: 0.2226 - val_acc: 0.7927 - val_loss: 0.6933
Epoch 26/50
345/345             12s 36ms/step -
acc: 0.9388 - loss: 0.1840 - val_acc: 0.7593 - val_loss: 0.8301
Epoch 27/50
345/345             18s 52ms/step -
acc: 0.9419 - loss: 0.1657 - val_acc: 0.7917 - val_loss: 0.7645
Epoch 28/50
345/345             12s 36ms/step -
acc: 0.9375 - loss: 0.1975 - val_acc: 0.7853 - val_loss: 0.8055
Epoch 29/50
345/345             18s 51ms/step -
```

```
acc: 0.9615 - loss: 0.1263 - val_acc: 0.7900 - val_loss: 0.7972
Epoch 30/50
345/345             12s 36ms/step -
acc: 0.9710 - loss: 0.0907 - val_acc: 0.7877 - val_loss: 0.8953
Epoch 31/50
345/345             18s 52ms/step -
acc: 0.9648 - loss: 0.1182 - val_acc: 0.7757 - val_loss: 0.9071
Epoch 32/50
345/345             12s 36ms/step -
acc: 0.9802 - loss: 0.0796 - val_acc: 0.7840 - val_loss: 0.9276
Epoch 33/50
345/345             14s 41ms/step -
acc: 0.9790 - loss: 0.0851 - val_acc: 0.7763 - val_loss: 0.9262
Epoch 34/50
345/345             16s 46ms/step -
acc: 0.9784 - loss: 0.0739 - val_acc: 0.7877 - val_loss: 0.9464
Epoch 35/50
345/345             12s 36ms/step -
acc: 0.9870 - loss: 0.0508 - val_acc: 0.7957 - val_loss: 0.9720
Epoch 36/50
345/345             18s 53ms/step -
acc: 0.9922 - loss: 0.0327 - val_acc: 0.7940 - val_loss: 1.0449
Epoch 37/50
345/345             13s 37ms/step -
acc: 0.9924 - loss: 0.0298 - val_acc: 0.7920 - val_loss: 1.0758
Epoch 38/50
345/345             15s 42ms/step -
acc: 0.9943 - loss: 0.0253 - val_acc: 0.7917 - val_loss: 1.0968
Epoch 39/50
345/345             15s 45ms/step -
acc: 0.9604 - loss: 0.2109 - val_acc: 0.7860 - val_loss: 1.0312
Epoch 40/50
345/345             14s 42ms/step -
acc: 0.9909 - loss: 0.0437 - val_acc: 0.7893 - val_loss: 1.0458
Epoch 41/50
345/345             15s 44ms/step -
acc: 0.9971 - loss: 0.0200 - val_acc: 0.7873 - val_loss: 1.1056
Epoch 42/50
345/345             15s 43ms/step -
acc: 0.9943 - loss: 0.0246 - val_acc: 0.7837 - val_loss: 1.1357
Epoch 43/50
345/345             15s 44ms/step -
acc: 0.9780 - loss: 0.1248 - val_acc: 0.7567 - val_loss: 1.2022
Epoch 44/50
345/345             15s 43ms/step -
acc: 0.9897 - loss: 0.0439 - val_acc: 0.7820 - val_loss: 1.0372
Epoch 45/50
345/345             13s 39ms/step -
```

```
acc: 0.9882 - loss: 0.0417 - val_acc: 0.7960 - val_loss: 1.0720
Epoch 46/50
345/345             18s 51ms/step -
acc: 0.9976 - loss: 0.0154 - val_acc: 0.7963 - val_loss: 1.1339
Epoch 47/50
345/345             15s 43ms/step -
acc: 0.9970 - loss: 0.0145 - val_acc: 0.7950 - val_loss: 1.1477
Epoch 48/50
345/345             15s 45ms/step -
acc: 0.9983 - loss: 0.0121 - val_acc: 0.7913 - val_loss: 1.1862
Epoch 49/50
345/345             15s 44ms/step -
acc: 0.9988 - loss: 0.0109 - val_acc: 0.7980 - val_loss: 1.1894
Epoch 50/50
345/345             15s 45ms/step -
acc: 0.9981 - loss: 0.0100 - val_acc: 0.7970 - val_loss: 1.2049
```

[25]:
```
best_epoch = np.argmin(history_KLD_SGD.history['val_loss']) + 1
print(f"Melhor época (menor val_loss): {best_epoch}")
```

```
Melhor época (menor val_loss): 18
```

3.5 Carregamento do modelo e validação

Carregamento e avaliação do modelo através do valor da accuracy.

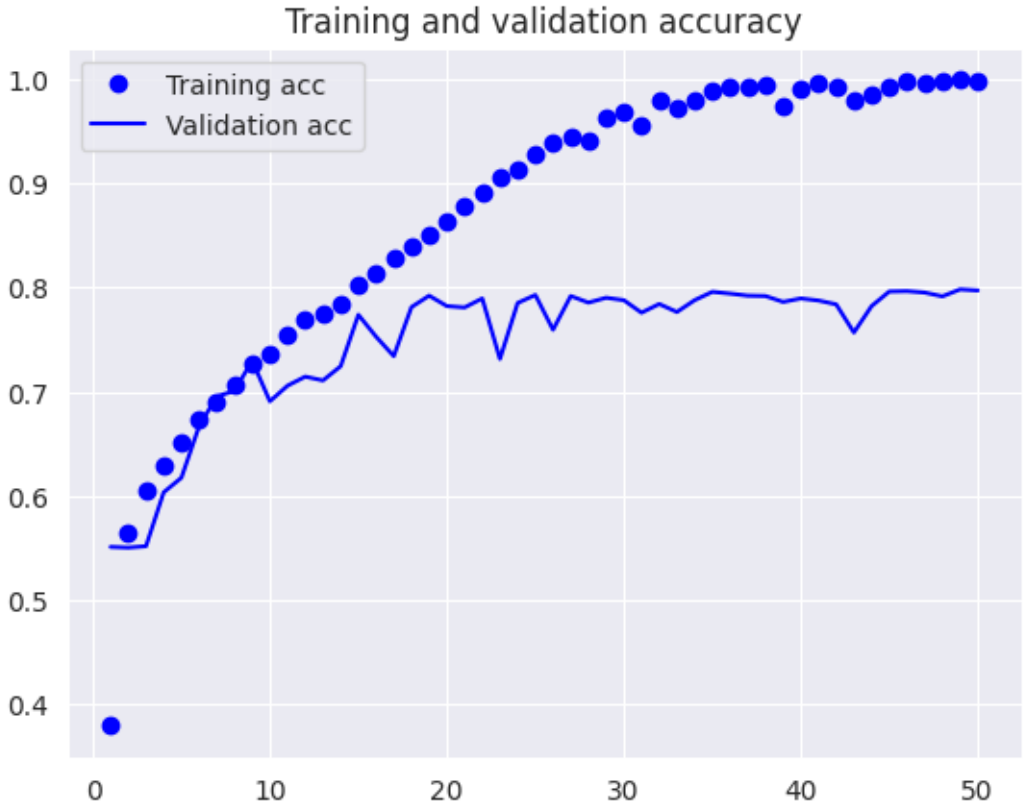
[26]:
```
modelS_KLD_SGD = keras.models.load_model('modelS_KLD_SGD.keras')
val_loss, val_acc = modelS_KLD_SGD.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

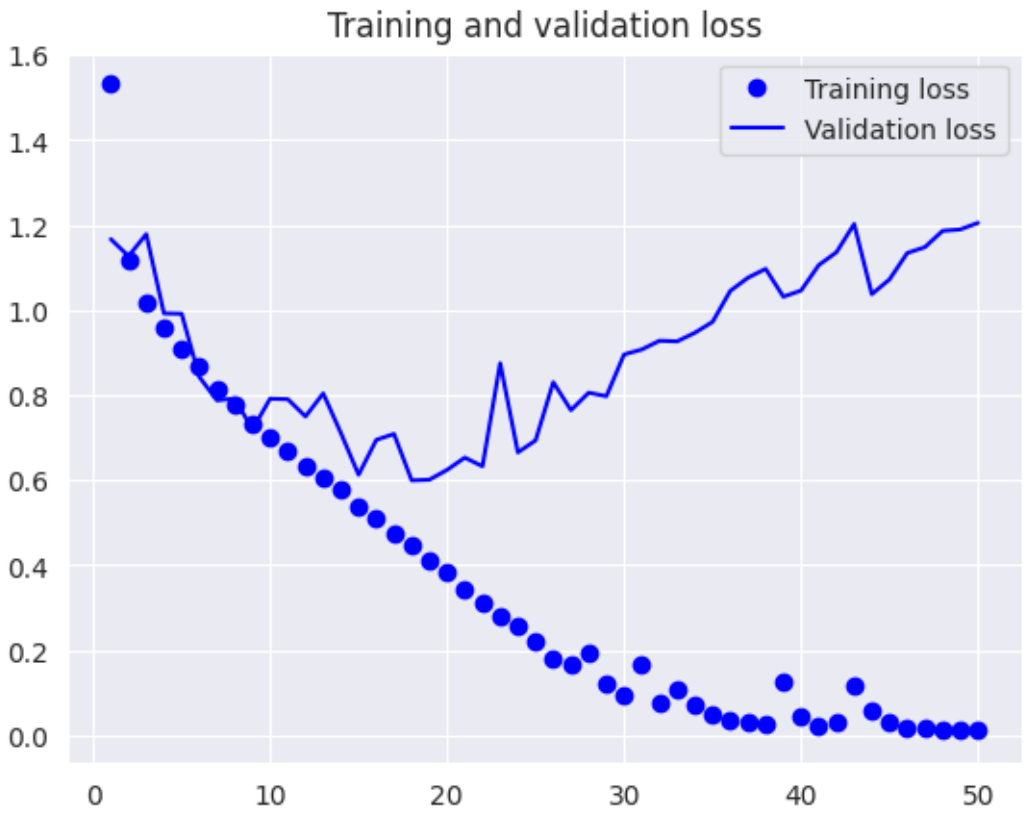
```
94/94             4s 28ms/step - acc:
0.7871 - loss: 0.5720
val_acc: 0.781000018119812
```

Representação gráfica dos valores da accuracy e da loss ao longo das épocas.

[27]:
```
acc = history_KLD_SGD.history['acc']
val_acc = history_KLD_SGD.history['val_acc']
loss = history_KLD_SGD.history['loss']
val_loss = history_KLD_SGD.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
```

```
plt.legend()
plt.show()
```





Avaliação da performance do modelo no conjunto de teste, utilizando o relatório de classificação. O relatório apresenta, para cada classe, as métricas precision, recall e F1-score, permitindo analisar detalhadamente os acertos e erros por classe.

[28]:
```
y_true, y_pred = get_true_pred(modelS_KLD_SGD, test_dataset)
report = classification_report(y_true, y_pred, target_names=class_names,␣
 ↪output_dict=True)
class_only_report = {k: v for k, v in report.items() if k in class_names}
df = pd.DataFrame(class_only_report).T
print(df[['precision', 'recall', 'f1-score']].round(3))
```

```
           precision  recall  f1-score
buildings      0.732   0.769     0.750
forest         0.933   0.876     0.903
glacier        0.689   0.807     0.743
mountain       0.774   0.737     0.755
sea            0.771   0.747     0.759
street         0.846   0.768     0.805
```

```
2025-06-12 20:52:07.769680: I tensorflow/core/framework/local_rendezvous.cc:407]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

4 Model (loss: categorical_crossentropy, optimizer: SGD)

4.1 Criação da CNN

Criação da CNN que irá receber imagens de 150x150 píxeis, aplica normalização e passa por quatro camadas convolucionais com max pooling para extrair características, seguidas de uma camada densa com 512 unidades e uma camada de saída softmax para classificação.

[33]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

print(model.summary())
```

```
Model: "functional_5"
```

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 150, 150, 3)	0
rescaling_2 (Rescaling)	(None, 150, 150, 3)	0
conv2d_8 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_8 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_9 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_10 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_10 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_11 (Conv2D)	(None, 15, 15, 128)	147,584

```
 max_pooling2d_11 (MaxPooling2D)   (None, 7, 7, 128)                0

 flatten_2 (Flatten)               (None, 6272)                     0

 dense_4 (Dense)                   (None, 512)                3,211,776

 dense_5 (Dense)                   (None, 6)                      3,078


Total params: 3,455,686 (13.18 MB)

Trainable params: 3,455,686 (13.18 MB)

Non-trainable params: 0 (0.00 B)

None
```

4.2 Compilação da CNN

Compilação da CNN utilizando a loss **categorical crossentropy** e o optimizer **SGD**.

[34]:
```
model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.SGD(learning_rate=0.01),
    metrics=['acc'])
```

4.3 Definição do callback

Definição de um callback que guarda automaticamente o modelo com a menor perda (loss) de validação durante o treino.

[35]:
```
checkpoint_filepath = 'modelS_CatCross_SGD.keras'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss',
    save_best_only=True)
```

4.4 Treino da CNN

Treino da CNN durante 50 épocas utilizando o dataset de validação e o callback para guardar o melhor modelo.

[36]:
```
history_CatCross_SGD = model.fit(
train_dataset,
epochs=50,
validation_data=validation_dataset,
callbacks=[model_checkpoint_callback])
```

```
Epoch 1/50
345/345             17s 42ms/step -
acc: 0.3117 - loss: 1.6749 - val_acc: 0.5253 - val_loss: 1.2196
Epoch 2/50
345/345             16s 47ms/step -
acc: 0.5506 - loss: 1.1587 - val_acc: 0.5393 - val_loss: 1.1288
Epoch 3/50
345/345             12s 35ms/step -
acc: 0.6010 - loss: 1.0195 - val_acc: 0.6223 - val_loss: 0.9644
Epoch 4/50
345/345             16s 47ms/step -
acc: 0.6281 - loss: 0.9593 - val_acc: 0.6510 - val_loss: 0.9041
Epoch 5/50
345/345             14s 40ms/step -
acc: 0.6553 - loss: 0.8868 - val_acc: 0.6363 - val_loss: 0.9463
Epoch 6/50
345/345             12s 36ms/step -
acc: 0.6709 - loss: 0.8480 - val_acc: 0.7040 - val_loss: 0.7910
Epoch 7/50
345/345             15s 44ms/step -
acc: 0.7075 - loss: 0.7865 - val_acc: 0.7247 - val_loss: 0.7414
Epoch 8/50
345/345             12s 34ms/step -
acc: 0.7215 - loss: 0.7476 - val_acc: 0.7040 - val_loss: 0.7952
Epoch 9/50
345/345             15s 42ms/step -
acc: 0.7356 - loss: 0.7172 - val_acc: 0.6243 - val_loss: 1.0020
Epoch 10/50
345/345             12s 35ms/step -
acc: 0.7454 - loss: 0.6819 - val_acc: 0.7433 - val_loss: 0.6923
Epoch 11/50
345/345             13s 38ms/step -
acc: 0.7622 - loss: 0.6444 - val_acc: 0.6940 - val_loss: 0.8103
Epoch 12/50
345/345             17s 49ms/step -
acc: 0.7810 - loss: 0.6091 - val_acc: 0.5783 - val_loss: 1.1887
Epoch 13/50
345/345             13s 36ms/step -
acc: 0.7852 - loss: 0.5973 - val_acc: 0.7653 - val_loss: 0.6421
Epoch 14/50
345/345             15s 44ms/step -
acc: 0.8089 - loss: 0.5397 - val_acc: 0.7817 - val_loss: 0.6116
Epoch 15/50
345/345             12s 35ms/step -
acc: 0.8095 - loss: 0.5182 - val_acc: 0.7807 - val_loss: 0.6194
Epoch 16/50
345/345             12s 34ms/step -
acc: 0.8243 - loss: 0.4847 - val_acc: 0.7887 - val_loss: 0.6118
```

```
Epoch 17/50
345/345             15s 43ms/step -
acc: 0.8380 - loss: 0.4542 - val_acc: 0.7667 - val_loss: 0.6719
Epoch 18/50
345/345             12s 36ms/step -
acc: 0.8470 - loss: 0.4336 - val_acc: 0.7920 - val_loss: 0.5944
Epoch 19/50
345/345             12s 35ms/step -
acc: 0.8610 - loss: 0.3895 - val_acc: 0.7623 - val_loss: 0.6811
Epoch 20/50
345/345             15s 43ms/step -
acc: 0.8684 - loss: 0.3711 - val_acc: 0.7850 - val_loss: 0.6264
Epoch 21/50
345/345             12s 34ms/step -
acc: 0.8787 - loss: 0.3387 - val_acc: 0.7473 - val_loss: 0.7768
Epoch 22/50
345/345             15s 43ms/step -
acc: 0.8974 - loss: 0.3051 - val_acc: 0.7980 - val_loss: 0.6321
Epoch 23/50
345/345             12s 34ms/step -
acc: 0.9005 - loss: 0.2830 - val_acc: 0.7863 - val_loss: 0.6889
Epoch 24/50
345/345             12s 34ms/step -
acc: 0.9069 - loss: 0.2665 - val_acc: 0.7953 - val_loss: 0.6613
Epoch 25/50
345/345             15s 43ms/step -
acc: 0.9285 - loss: 0.2112 - val_acc: 0.5633 - val_loss: 1.6123
Epoch 26/50
345/345             12s 34ms/step -
acc: 0.9263 - loss: 0.2168 - val_acc: 0.7997 - val_loss: 0.7278
Epoch 27/50
345/345             12s 34ms/step -
acc: 0.9488 - loss: 0.1642 - val_acc: 0.7840 - val_loss: 0.7833
Epoch 28/50
345/345             15s 43ms/step -
acc: 0.9584 - loss: 0.1344 - val_acc: 0.8060 - val_loss: 0.7441
Epoch 29/50
345/345             12s 35ms/step -
acc: 0.9481 - loss: 0.1797 - val_acc: 0.7923 - val_loss: 0.8098
Epoch 30/50
345/345             12s 34ms/step -
acc: 0.9699 - loss: 0.1015 - val_acc: 0.8057 - val_loss: 0.8770
Epoch 31/50
345/345             15s 43ms/step -
acc: 0.9738 - loss: 0.0851 - val_acc: 0.7990 - val_loss: 0.8412
Epoch 32/50
345/345             11s 31ms/step -
acc: 0.9811 - loss: 0.0646 - val_acc: 0.7650 - val_loss: 1.0980
```

```
Epoch 33/50
345/345             11s 30ms/step -
acc: 0.9657 - loss: 0.1248 - val_acc: 0.8043 - val_loss: 0.8360
Epoch 34/50
345/345             14s 39ms/step -
acc: 0.9856 - loss: 0.0554 - val_acc: 0.7907 - val_loss: 0.9453
Epoch 35/50
345/345             11s 31ms/step -
acc: 0.9873 - loss: 0.0477 - val_acc: 0.7973 - val_loss: 0.9590
Epoch 36/50
345/345             11s 30ms/step -
acc: 0.9885 - loss: 0.0419 - val_acc: 0.8030 - val_loss: 0.9738
Epoch 37/50
345/345             13s 39ms/step -
acc: 0.9863 - loss: 0.0637 - val_acc: 0.7987 - val_loss: 0.9227
Epoch 38/50
345/345             11s 30ms/step -
acc: 0.9953 - loss: 0.0250 - val_acc: 0.8057 - val_loss: 1.0174
Epoch 39/50
345/345             11s 31ms/step -
acc: 0.9953 - loss: 0.0202 - val_acc: 0.8090 - val_loss: 1.0289
Epoch 40/50
345/345             13s 39ms/step -
acc: 0.9785 - loss: 0.0870 - val_acc: 0.8010 - val_loss: 1.0142
Epoch 41/50
345/345             11s 30ms/step -
acc: 0.9916 - loss: 0.0374 - val_acc: 0.8020 - val_loss: 0.9467
Epoch 42/50
345/345             11s 30ms/step -
acc: 0.9979 - loss: 0.0161 - val_acc: 0.8073 - val_loss: 1.0257
Epoch 43/50
345/345             13s 39ms/step -
acc: 0.9976 - loss: 0.0143 - val_acc: 0.8087 - val_loss: 1.0781
Epoch 44/50
345/345             11s 30ms/step -
acc: 0.9972 - loss: 0.0159 - val_acc: 0.8080 - val_loss: 1.1023
Epoch 45/50
345/345             10s 30ms/step -
acc: 0.9976 - loss: 0.0123 - val_acc: 0.8043 - val_loss: 1.1398
Epoch 46/50
345/345             13s 39ms/step -
acc: 0.9970 - loss: 0.0131 - val_acc: 0.8070 - val_loss: 1.1397
Epoch 47/50
345/345             11s 30ms/step -
acc: 0.9849 - loss: 0.0760 - val_acc: 0.8107 - val_loss: 1.0579
Epoch 48/50
345/345             11s 30ms/step -
acc: 0.9978 - loss: 0.0128 - val_acc: 0.8020 - val_loss: 1.1259
```

```
Epoch 49/50
345/345             13s 39ms/step -
acc: 0.9922 - loss: 0.0393 - val_acc: 0.8040 - val_loss: 1.1032
Epoch 50/50
345/345             11s 31ms/step -
acc: 0.9950 - loss: 0.0205 - val_acc: 0.8083 - val_loss: 1.1324
```

[37]:
```
best_epoch = np.argmin(history_CatCross_SGD.history['val_loss']) + 1
print(f"Melhor época (menor val_loss): {best_epoch}")
```

```
Melhor época (menor val_loss): 18
```

4.5 Carregamento do modelo e validação

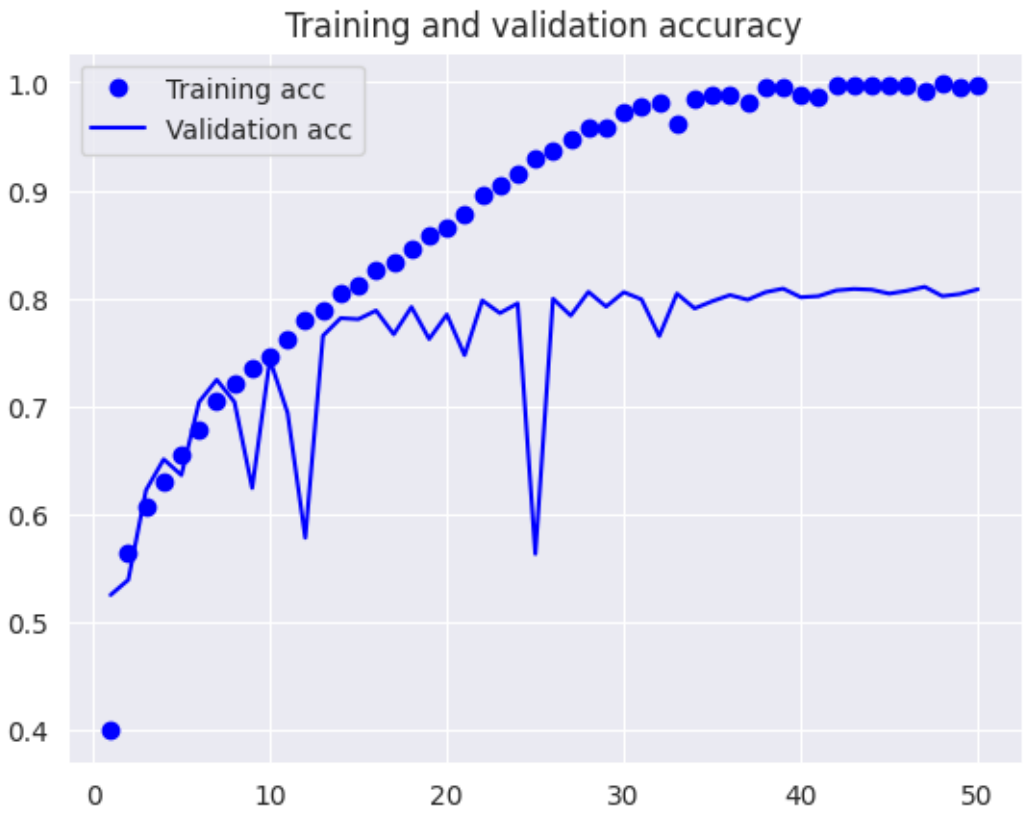
Carregamento e avaliação do modelo através do valor da accuracy.

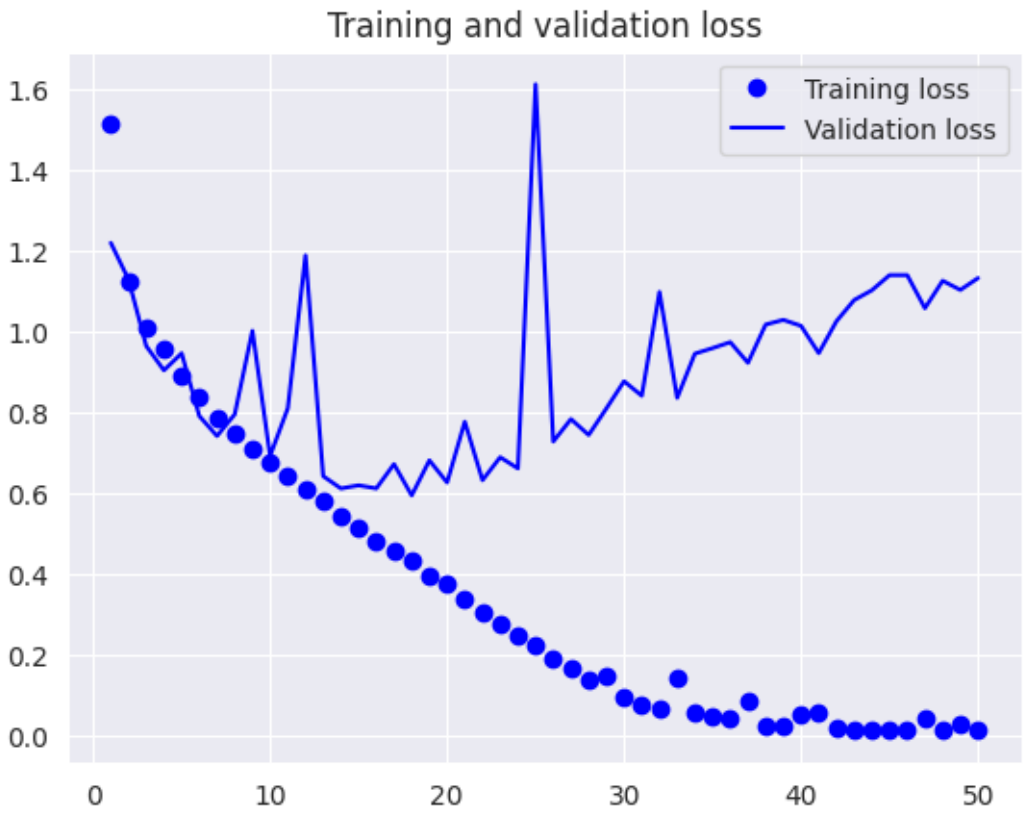
[38]:
```
modelS_CatCross_SGD = keras.models.load_model('modelS_CatCross_SGD.keras')
val_loss, val_acc = modelS_CatCross_SGD.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

```
94/94             3s 19ms/step - acc:
0.7904 - loss: 0.5882
val_acc: 0.7919999957084656
```

Representação gráfica dos valores da accuracy e da loss ao longo das épocas.

[39]:
```
acc = history_CatCross_SGD.history['acc']
val_acc = history_CatCross_SGD.history['val_acc']
loss = history_CatCross_SGD.history['loss']
val_loss = history_CatCross_SGD.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```


Training and validation accuracy
Training acc
Validation acc
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50



Avaliação da performance do modelo no conjunto de teste, utilizando o relatório de classificação. O relatório apresenta, para cada classe, as métricas precision, recall e F1-score, permitindo analisar detalhadamente os acertos e erros por classe.

[40]:
```
y_true, y_pred = get_true_pred(modelS_CatCross_SGD, test_dataset)
report = classification_report(y_true, y_pred, target_names=class_names,␣
  ↪output_dict=True)
class_only_report = {k: v for k, v in report.items() if k in class_names}
df = pd.DataFrame(class_only_report).T
print(df[['precision', 'recall', 'f1-score']].round(3))
```

```
           precision  recall  f1-score
buildings      0.773   0.773     0.773
forest         0.883   0.937     0.909
glacier        0.689   0.826     0.752
mountain       0.777   0.758     0.768
sea            0.833   0.731     0.779
street         0.872   0.760     0.812
```

5 Model (loss: KLDivergence, optimizer: RMSprop)

5.1 Criação da CNN

Criação da CNN que irá receber imagens de 150x150 píxeis, aplica normalização e passa por quatro camadas convolucionais com max pooling para extrair características, seguidas de uma camada densa com 512 unidades e uma camada de saída softmax para classificação.

[54]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

print(model.summary())
```

```
Model: "functional_7"
```

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 150, 150, 3)	0
rescaling_4 (Rescaling)	(None, 150, 150, 3)	0
conv2d_16 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_16 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_17 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_17 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_18 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_18 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_19 (Conv2D)	(None, 15, 15, 128)	147,584

```
 max_pooling2d_19 (MaxPooling2D)   (None, 7, 7, 128)                0

 flatten_4 (Flatten)               (None, 6272)                     0

 dense_8 (Dense)                   (None, 512)                3,211,776

 dense_9 (Dense)                   (None, 6)                      3,078


Total params: 3,455,686 (13.18 MB)

Trainable params: 3,455,686 (13.18 MB)

Non-trainable params: 0 (0.00 B)

None
```

5.2 Compilação da CNN

Compilação da CNN utilizando a loss **KLDivergence** e o optimizer **RMSprop**.

[55]:
```
model.compile(
    loss=tf.keras.losses.KLDivergence(),
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=1e-4),
    metrics=['acc'])
```

5.3 Definição do callback

Definição de um callback que guarda automaticamente o modelo com a menor perda (loss) de validação durante o treino.

[56]:
```
checkpoint_filepath = 'modelS_KLD_RMS.keras'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss',
    save_best_only=True)
```

5.4 Treino da CNN

Treino da CNN durante 50 épocas utilizando o dataset de validação e o callback para guardar o melhor modelo.

[57]:
```
history_KLD_RMS = model.fit(
train_dataset,
epochs=50,
validation_data=validation_dataset,
callbacks=[model_checkpoint_callback])
```

```
Epoch 1/50
345/345             20s 48ms/step -
acc: 0.4663 - loss: 1.3290 - val_acc: 0.5753 - val_loss: 1.0506
Epoch 2/50
345/345             12s 34ms/step -
acc: 0.6347 - loss: 0.9598 - val_acc: 0.6423 - val_loss: 0.9260
Epoch 3/50
345/345             12s 34ms/step -
acc: 0.6871 - loss: 0.8445 - val_acc: 0.7303 - val_loss: 0.7340
Epoch 4/50
345/345             15s 42ms/step -
acc: 0.7180 - loss: 0.7534 - val_acc: 0.6693 - val_loss: 0.9300
Epoch 5/50
345/345             12s 35ms/step -
acc: 0.7488 - loss: 0.6864 - val_acc: 0.7473 - val_loss: 0.6989
Epoch 6/50
345/345             12s 35ms/step -
acc: 0.7696 - loss: 0.6371 - val_acc: 0.7650 - val_loss: 0.6384
Epoch 7/50
345/345             15s 43ms/step -
acc: 0.7939 - loss: 0.5875 - val_acc: 0.7540 - val_loss: 0.6796
Epoch 8/50
345/345             12s 34ms/step -
acc: 0.8059 - loss: 0.5519 - val_acc: 0.7943 - val_loss: 0.5728
Epoch 9/50
345/345             12s 34ms/step -
acc: 0.8168 - loss: 0.5104 - val_acc: 0.7910 - val_loss: 0.5759
Epoch 10/50
345/345             15s 43ms/step -
acc: 0.8259 - loss: 0.4722 - val_acc: 0.8153 - val_loss: 0.5304
Epoch 11/50
345/345             12s 34ms/step -
acc: 0.8423 - loss: 0.4431 - val_acc: 0.8190 - val_loss: 0.5000
Epoch 12/50
345/345             15s 42ms/step -
acc: 0.8492 - loss: 0.4153 - val_acc: 0.7613 - val_loss: 0.6571
Epoch 13/50
345/345             12s 34ms/step -
acc: 0.8557 - loss: 0.3983 - val_acc: 0.8190 - val_loss: 0.4973
Epoch 14/50
345/345             12s 34ms/step -
acc: 0.8693 - loss: 0.3681 - val_acc: 0.8120 - val_loss: 0.5250
Epoch 15/50
345/345             15s 42ms/step -
acc: 0.8793 - loss: 0.3397 - val_acc: 0.8170 - val_loss: 0.5077
Epoch 16/50
345/345             12s 34ms/step -
acc: 0.8847 - loss: 0.3175 - val_acc: 0.7830 - val_loss: 0.6482
```

```
Epoch 17/50
345/345             13s 37ms/step -
acc: 0.8963 - loss: 0.2932 - val_acc: 0.8113 - val_loss: 0.5510
Epoch 18/50
345/345             16s 47ms/step -
acc: 0.9061 - loss: 0.2649 - val_acc: 0.8047 - val_loss: 0.5980
Epoch 19/50
345/345             13s 38ms/step -
acc: 0.9142 - loss: 0.2475 - val_acc: 0.8200 - val_loss: 0.5531
Epoch 20/50
345/345             18s 51ms/step -
acc: 0.9223 - loss: 0.2218 - val_acc: 0.8143 - val_loss: 0.5870
Epoch 21/50
345/345             15s 42ms/step -
acc: 0.9336 - loss: 0.1979 - val_acc: 0.8220 - val_loss: 0.5715
Epoch 22/50
345/345             17s 48ms/step -
acc: 0.9401 - loss: 0.1773 - val_acc: 0.8280 - val_loss: 0.5440
Epoch 23/50
345/345             15s 43ms/step -
acc: 0.9466 - loss: 0.1509 - val_acc: 0.8167 - val_loss: 0.6117
Epoch 24/50
345/345             14s 40ms/step -
acc: 0.9550 - loss: 0.1352 - val_acc: 0.8240 - val_loss: 0.6028
Epoch 25/50
345/345             16s 47ms/step -
acc: 0.9626 - loss: 0.1194 - val_acc: 0.8230 - val_loss: 0.6811
Epoch 26/50
345/345             15s 43ms/step -
acc: 0.9687 - loss: 0.1016 - val_acc: 0.8290 - val_loss: 0.6299
Epoch 27/50
345/345             17s 49ms/step -
acc: 0.9734 - loss: 0.0893 - val_acc: 0.8087 - val_loss: 0.7277
Epoch 28/50
345/345             14s 40ms/step -
acc: 0.9738 - loss: 0.0777 - val_acc: 0.8217 - val_loss: 0.7308
Epoch 29/50
345/345             16s 47ms/step -
acc: 0.9819 - loss: 0.0615 - val_acc: 0.8273 - val_loss: 0.7720
Epoch 30/50
345/345             13s 38ms/step -
acc: 0.9816 - loss: 0.0612 - val_acc: 0.8140 - val_loss: 0.8927
Epoch 31/50
345/345             12s 36ms/step -
acc: 0.9839 - loss: 0.0554 - val_acc: 0.8133 - val_loss: 0.8512
Epoch 32/50
345/345             15s 44ms/step -
acc: 0.9856 - loss: 0.0479 - val_acc: 0.8093 - val_loss: 0.8768
```

```
Epoch 33/50
345/345             12s 36ms/step -
acc: 0.9887 - loss: 0.0416 - val_acc: 0.8187 - val_loss: 0.8435
Epoch 34/50
345/345             16s 47ms/step -
acc: 0.9900 - loss: 0.0310 - val_acc: 0.8247 - val_loss: 0.8899
Epoch 35/50
345/345             13s 36ms/step -
acc: 0.9912 - loss: 0.0306 - val_acc: 0.7297 - val_loss: 1.7118
Epoch 36/50
345/345             12s 36ms/step -
acc: 0.9886 - loss: 0.0465 - val_acc: 0.8210 - val_loss: 0.9315
Epoch 37/50
345/345             15s 44ms/step -
acc: 0.9898 - loss: 0.0351 - val_acc: 0.8243 - val_loss: 0.9315
Epoch 38/50
345/345             12s 36ms/step -
acc: 0.9931 - loss: 0.0252 - val_acc: 0.8177 - val_loss: 1.0275
Epoch 39/50
345/345             15s 44ms/step -
acc: 0.9919 - loss: 0.0294 - val_acc: 0.8183 - val_loss: 1.0122
Epoch 40/50
345/345             12s 35ms/step -
acc: 0.9929 - loss: 0.0257 - val_acc: 0.8213 - val_loss: 1.0232
Epoch 41/50
345/345             12s 36ms/step -
acc: 0.9936 - loss: 0.0247 - val_acc: 0.8197 - val_loss: 1.0245
Epoch 42/50
345/345             15s 44ms/step -
acc: 0.9934 - loss: 0.0262 - val_acc: 0.8237 - val_loss: 1.0647
Epoch 43/50
345/345             12s 35ms/step -
acc: 0.9951 - loss: 0.0162 - val_acc: 0.8177 - val_loss: 1.0867
Epoch 44/50
345/345             12s 36ms/step -
acc: 0.9949 - loss: 0.0173 - val_acc: 0.8210 - val_loss: 1.1388
Epoch 45/50
345/345             15s 44ms/step -
acc: 0.9948 - loss: 0.0163 - val_acc: 0.8233 - val_loss: 1.0966
Epoch 46/50
345/345             12s 35ms/step -
acc: 0.9949 - loss: 0.0188 - val_acc: 0.8210 - val_loss: 1.1640
Epoch 47/50
345/345             16s 46ms/step -
acc: 0.9931 - loss: 0.0187 - val_acc: 0.8213 - val_loss: 1.1654
Epoch 48/50
345/345             13s 38ms/step -
acc: 0.9951 - loss: 0.0167 - val_acc: 0.8207 - val_loss: 1.1432
```

```
Epoch 49/50
345/345             12s 36ms/step -
acc: 0.9957 - loss: 0.0167 - val_acc: 0.8247 - val_loss: 1.1050
Epoch 50/50
345/345             17s 48ms/step -
acc: 0.9972 - loss: 0.0120 - val_acc: 0.8170 - val_loss: 1.1817
```

[58]:
```
best_epoch = np.argmin(history_KLD_RMS.history['val_loss']) + 1
print(f"Melhor época (menor val_loss): {best_epoch}")
```

```
Melhor época (menor val_loss): 13
```

5.5 Carregamento do modelo e validação

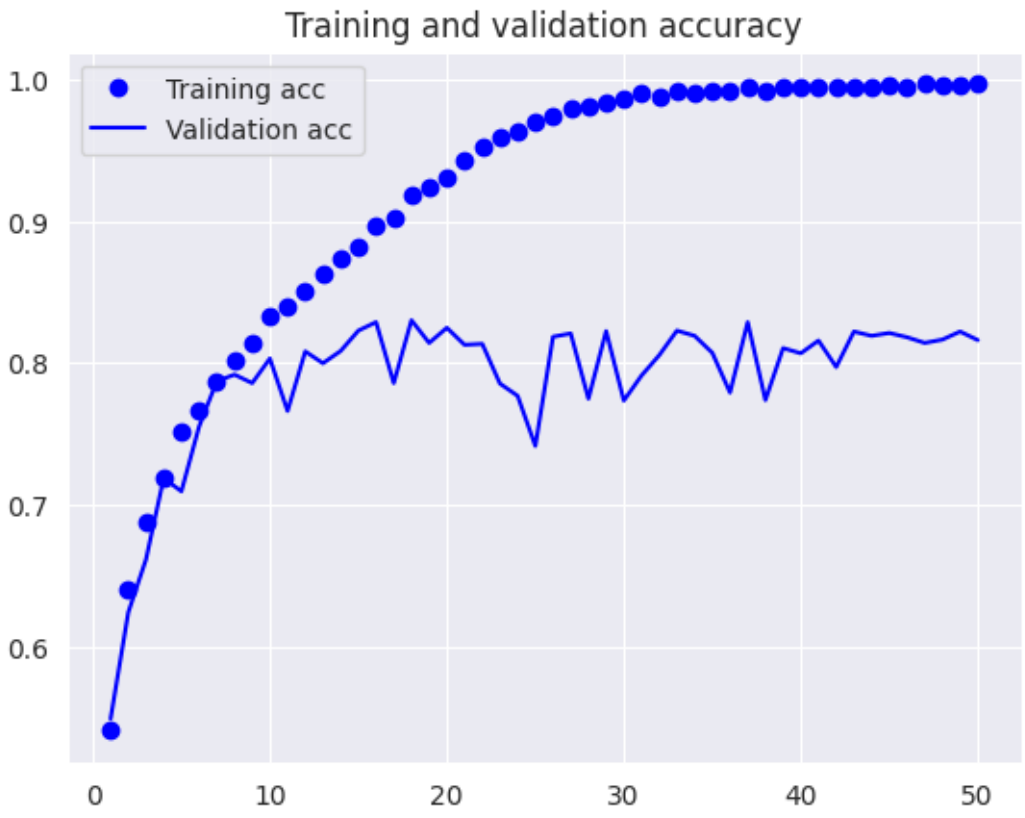
Carregamento e avaliação do modelo através do valor da accuracy.

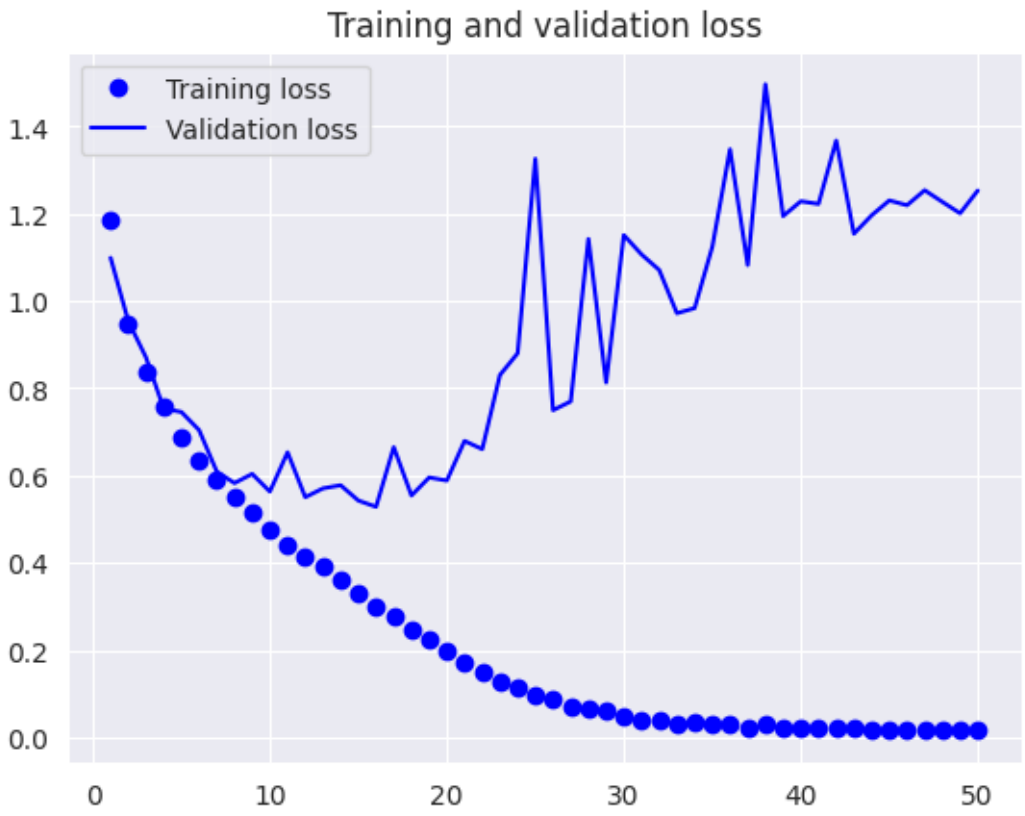
[59]:
```
modelS_KLD_RMS = keras.models.load_model('modelS_KLD_RMS.keras')
val_loss, val_acc = modelS_KLD_RMS.evaluate(validation_dataset)
print('val_acc:', val_acc)
```

```
94/94             3s 18ms/step - acc:
0.8298 - loss: 0.4612
val_acc: 0.8190000057220459
```

Representação gráfica dos valores da accuracy e da loss ao longo das épocas.

[47]:
```
acc = history_KLD_RMS.history['acc']
val_acc = history_KLD_RMS.history['val_acc']
loss = history_KLD_RMS.history['loss']
val_loss = history_KLD_RMS.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```


Training and validation accuracy
Training acc
Validation acc
1.0
0.9
0.8
0.7
0.6
0
10
20
30
40
50



Avaliação da performance do modelo no conjunto de teste, utilizando o relatório de classificação. O relatório apresenta, para cada classe, as métricas precision, recall e F1-score, permitindo analisar detalhadamente os acertos e erros por classe.

[48]:
```
y_true, y_pred = get_true_pred(modelS_KLD_RMS, test_dataset)
report = classification_report(y_true, y_pred, target_names=class_names,␣
 ↪output_dict=True)
class_only_report = {k: v for k, v in report.items() if k in class_names}
df = pd.DataFrame(class_only_report).T
print(df[['precision', 'recall', 'f1-score']].round(3))
```

```
           precision  recall  f1-score
buildings      0.856   0.762     0.806
forest         0.945   0.941     0.943
glacier        0.819   0.759     0.788
mountain       0.820   0.773     0.796
sea            0.757   0.924     0.832
street         0.862   0.876     0.869
```

```
2025-06-12 21:22:56.730716: I tensorflow/core/framework/local_rendezvous.cc:407]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

6 Avaliação do melhor modelo

6.1 Comparação dos modelos utilizando a accuracy

[60]:
```
val_loss_CatCross_RMS, val_acc_CatCross_RMS = modelS_CatCross_RMS.
 ↪evaluate(validation_dataset)
val_loss_CatCross_SGD, val_acc_CatCross_SGD = modelS_CatCross_SGD.
 ↪evaluate(validation_dataset)
val_loss_KLD_RMS, val_acc_KLD_RMS = modelS_KLD_RMS.evaluate(validation_dataset)
val_loss_KLD_SGD, val_acc_KLD_SGD = modelS_KLD_SGD.evaluate(validation_dataset)

print("Validation Accuracy dos modelos:")
print(f"CatCross + RMSprop: {val_acc_CatCross_RMS:.4f}")
print(f"CatCross + SGD    : {val_acc_CatCross_SGD:.4f}")
print(f"KLD      + RMSprop: {val_acc_KLD_RMS:.4f}")
print(f"KLD      + SGD    : {val_acc_KLD_SGD:.4f}")

results = {
    'CatCross_RMS': val_acc_CatCross_RMS,
    'CatCross_SGD': val_acc_CatCross_SGD,
    'KLD_RMS': val_acc_KLD_RMS,
    'KLD_SGD': val_acc_KLD_SGD
}

# Identificação do melhor modelo com base na maior val_accuracy
best_model = max(results, key=results.get)
best_accuracy = results[best_model]

print(f"\nMelhor modelo: {best_model} com val_accuracy = {best_accuracy:.4f}")
```

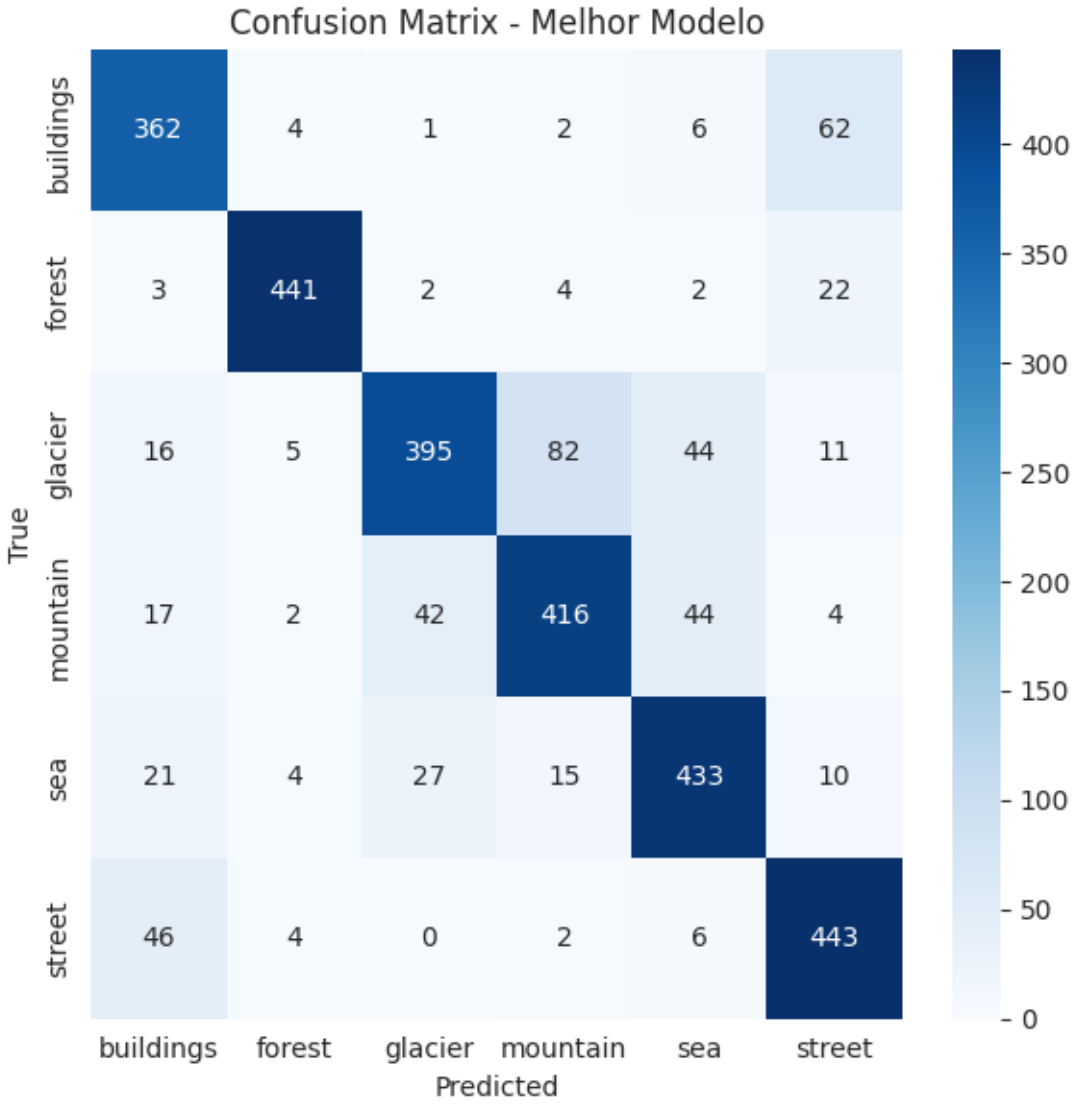
```
94/94            2s 16ms/step - acc:
0.8302 - loss: 0.4853
94/94            2s 19ms/step - acc:
0.7917 - loss: 0.5884
94/94            2s 18ms/step - acc:
0.8339 - loss: 0.4576
94/94            5s 52ms/step - acc:
0.7844 - loss: 0.5841
Validation Accuracy dos modelos:
CatCross + RMSprop: 0.8250
CatCross + SGD    : 0.7920
KLD      + RMSprop: 0.8190
KLD      + SGD    : 0.7810

Melhor modelo: CatCross_RMS com val_accuracy = 0.8250
```

6.2 Matriz de confusão do melhor modelo

[52]:
```
y_true, y_pred = get_true_pred(modelS_CatCross_RMS, test_dataset)
cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Melhor Modelo')
plt.tight_layout()
plt.show()
```



6.3 Calcular saída do modelo para uma imagem

[61]:
```
img_path = 'Dataset/archive/seg_test/sea/20072.jpg'

img = tf.keras.preprocessing.image.load_img(
    img_path,
    target_size=(150, 150),
    interpolation='bilinear'
)

plt.imshow(img)
plt.axis('off')
plt.title("Imagem de Teste")
plt.show()

img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

result = modelS_CatCross_RMS.predict(img_array)

class_names = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
print("Probabilidades por classe:")
for i, prob in enumerate(result[0]):
    print(f"{class_names[i]:>10s}: {prob:.4f}")


predicted_class = np.argmax(result)
print(f"\nClasse prevista: {class_names[predicted_class]}␣
 ↪({result[0][predicted_class]:.4f})")
```


```
1/1             0s 120ms/step
Probabilidades por classe:
 buildings: 0.1919
    forest: 0.0034
   glacier: 0.0611
  mountain: 0.0504
       sea: 0.6622
    street: 0.0311

Classe prevista: sea (0.6622)
```