modelS_data_augmentation

June 12, 2025

[5]:
```python
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
import matplotlib.pyplot as plt
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 22:59:03.516266: 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 00:00:1749765543.577514  837034 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1749765543.596184  837034 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1749765543.682290  837034 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749765543.682315  837034 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749765543.682317  837034 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1749765543.682318  837034 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
```

```
target more than once.
2025-06-12 22:59:03.703494: 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.
```

1 Funções

[2]:
```
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)
```

1.1 Carregamento do dataset

Carrega o dataset distribuido pelos diferentes conjuntos de dados.

[8]:
```
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'
```

```python
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
```

1.2 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.

1.2.1 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**

1.2.2 Número total de imagens por conjunto de dados:

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

1.3 Processamento dos dados

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

[9]:
```
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.
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 com data augmentation

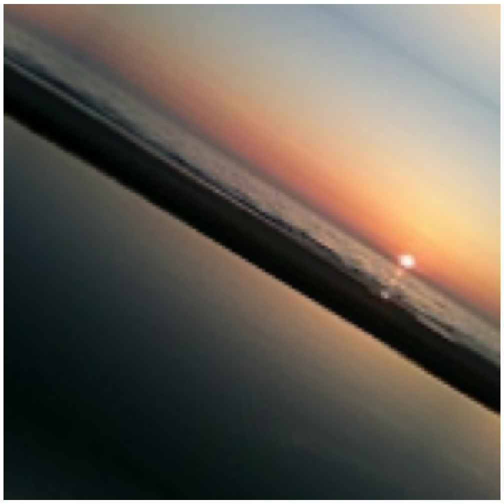
2.1 Data augmentation

Criação de data augmentation que aplica transformações aleatórias às imagens durante o treino, incluindo inversão horizontal, pequenas rotações e zoom, com o objetivo de aumentar a variabilidade dos dados e melhorar a generalização do modelo.

[19]:
```
data_augmentation = keras.Sequential(
 [
 layers.RandomFlip("horizontal"),
 layers.RandomRotation(0.1),
 layers.RandomZoom(0.2),
 ]
 )
```

Aplicação de data augmentation a um batch de imagens do conjunto de treino e visualiza quatro versões aumentadas da mesma imagem, permitindo observar os efeitos das transformações aleatórias aplicadas.

[20]:
```
plt.figure(figsize=(10, 10))
for images, _ in train_dataset.take(1):
  for i in range(4):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(2, 2, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```




2.2 Criação da CNN

Criação da CNN que irá receber imagens de 150x150 píxeis, aplica normalização e utiliza data augmentation com transformações aleatórias (como inversão horizontal, rotação e zoom) para aumentar a variabilidade dos dados de entrada. Em seguida, 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 multiclasse.

[21]:
```
inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
```

```python
x = layers.Conv2D(64, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(128, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Conv2D(128, 3, activation="relu")(x)
x = layers.MaxPooling2D(2)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(len(class_names), activation="softmax")(x)
model_dropout_dataAugmentation = keras.Model(inputs, outputs)

print(model_dropout_dataAugmentation.summary())
```

Model: "functional_5"

Layer (type)	Output Shape	Param #
input_layer_5 (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
dropout_2 (Dropout)	(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
```

2.3 Compilação da CNN

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

[22]:
```
model_dropout_dataAugmentation.compile(
loss='categorical_crossentropy',
optimizer=tf.keras.optimizers.RMSprop(learning_rate=1e-4),
metrics=['acc'])
```

2.4 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_CatCross_RMS_dropout_dataAugmentation.keras'
model_checkpoint_callback = keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_loss',
    save_best_only=True)
```

2.5 Treino da CNN

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

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

```
Epoch 1/50
345/345               88s 253ms/step -
acc: 0.4308 - loss: 1.4066 - val_acc: 0.6120 - val_loss: 1.0079
Epoch 2/50
345/345               87s 252ms/step -
acc: 0.6144 - loss: 0.9796 - val_acc: 0.6283 - val_loss: 0.9841
Epoch 3/50
```

```
345/345              87s 252ms/step -
acc: 0.6878 - loss: 0.8426 - val_acc: 0.6893 - val_loss: 0.8059
Epoch 4/50
345/345              87s 253ms/step -
acc: 0.7079 - loss: 0.7812 - val_acc: 0.7280 - val_loss: 0.7473
Epoch 5/50
345/345              88s 255ms/step -
acc: 0.7358 - loss: 0.7101 - val_acc: 0.7277 - val_loss: 0.7230
Epoch 6/50
345/345              88s 256ms/step -
acc: 0.7530 - loss: 0.6648 - val_acc: 0.7490 - val_loss: 0.6617
Epoch 7/50
345/345              89s 258ms/step -
acc: 0.7703 - loss: 0.6131 - val_acc: 0.7567 - val_loss: 0.6420
Epoch 8/50
345/345              95s 276ms/step -
acc: 0.7837 - loss: 0.5876 - val_acc: 0.7830 - val_loss: 0.5911
Epoch 9/50
345/345              91s 264ms/step -
acc: 0.7991 - loss: 0.5477 - val_acc: 0.7920 - val_loss: 0.5681
Epoch 10/50
345/345              86s 251ms/step -
acc: 0.8074 - loss: 0.5326 - val_acc: 0.7543 - val_loss: 0.6726
Epoch 11/50
345/345              90s 262ms/step -
acc: 0.8222 - loss: 0.4941 - val_acc: 0.7263 - val_loss: 0.7636
Epoch 12/50
345/345              87s 252ms/step -
acc: 0.8209 - loss: 0.4880 - val_acc: 0.8100 - val_loss: 0.5165
Epoch 13/50
345/345              86s 249ms/step -
acc: 0.8299 - loss: 0.4666 - val_acc: 0.7767 - val_loss: 0.6207
Epoch 14/50
345/345              86s 251ms/step -
acc: 0.8401 - loss: 0.4436 - val_acc: 0.8093 - val_loss: 0.5295
Epoch 15/50
345/345              86s 250ms/step -
acc: 0.8463 - loss: 0.4213 - val_acc: 0.8210 - val_loss: 0.4847
Epoch 16/50
345/345              86s 250ms/step -
acc: 0.8532 - loss: 0.4117 - val_acc: 0.8070 - val_loss: 0.5404
Epoch 17/50
345/345              86s 250ms/step -
acc: 0.8619 - loss: 0.3939 - val_acc: 0.8200 - val_loss: 0.5037
Epoch 18/50
345/345              86s 249ms/step -
acc: 0.8600 - loss: 0.3860 - val_acc: 0.8217 - val_loss: 0.5003
Epoch 19/50
```

```
345/345              86s 250ms/step -
acc: 0.8639 - loss: 0.3776 - val_acc: 0.8207 - val_loss: 0.4975
Epoch 20/50
345/345              87s 251ms/step -
acc: 0.8667 - loss: 0.3594 - val_acc: 0.8207 - val_loss: 0.5081
Epoch 21/50
345/345              86s 250ms/step -
acc: 0.8798 - loss: 0.3425 - val_acc: 0.8287 - val_loss: 0.4674
Epoch 22/50
345/345              86s 250ms/step -
acc: 0.8768 - loss: 0.3355 - val_acc: 0.8120 - val_loss: 0.5213
Epoch 23/50
345/345              86s 250ms/step -
acc: 0.8875 - loss: 0.3124 - val_acc: 0.8013 - val_loss: 0.5754
Epoch 24/50
345/345              86s 250ms/step -
acc: 0.8833 - loss: 0.3187 - val_acc: 0.8320 - val_loss: 0.4904
Epoch 25/50
345/345              86s 249ms/step -
acc: 0.8946 - loss: 0.2854 - val_acc: 0.8180 - val_loss: 0.5615
Epoch 26/50
345/345              86s 249ms/step -
acc: 0.8991 - loss: 0.2785 - val_acc: 0.8123 - val_loss: 0.5585
Epoch 27/50
345/345              86s 250ms/step -
acc: 0.9037 - loss: 0.2717 - val_acc: 0.8353 - val_loss: 0.4906
Epoch 28/50
345/345              87s 251ms/step -
acc: 0.9005 - loss: 0.2686 - val_acc: 0.8217 - val_loss: 0.5270
Epoch 29/50
345/345              87s 252ms/step -
acc: 0.9061 - loss: 0.2536 - val_acc: 0.8360 - val_loss: 0.4902
Epoch 30/50
345/345              87s 251ms/step -
acc: 0.9094 - loss: 0.2399 - val_acc: 0.8380 - val_loss: 0.4887
Epoch 31/50
345/345              86s 249ms/step -
acc: 0.9211 - loss: 0.2329 - val_acc: 0.8230 - val_loss: 0.5615
Epoch 32/50
345/345              86s 251ms/step -
acc: 0.9163 - loss: 0.2289 - val_acc: 0.8420 - val_loss: 0.4709
Epoch 33/50
345/345              86s 249ms/step -
acc: 0.9228 - loss: 0.2137 - val_acc: 0.8440 - val_loss: 0.4890
Epoch 34/50
345/345              87s 252ms/step -
acc: 0.9254 - loss: 0.2014 - val_acc: 0.8460 - val_loss: 0.4910
Epoch 35/50
```

```
345/345              86s 250ms/step -
acc: 0.9265 - loss: 0.2014 - val_acc: 0.8463 - val_loss: 0.4929
Epoch 36/50
345/345              87s 253ms/step -
acc: 0.9340 - loss: 0.1915 - val_acc: 0.8327 - val_loss: 0.5455
Epoch 37/50
345/345              86s 251ms/step -
acc: 0.9384 - loss: 0.1782 - val_acc: 0.8500 - val_loss: 0.4931
Epoch 38/50
345/345              86s 249ms/step -
acc: 0.9379 - loss: 0.1797 - val_acc: 0.8380 - val_loss: 0.5451
Epoch 39/50
345/345              86s 250ms/step -
acc: 0.9401 - loss: 0.1656 - val_acc: 0.8337 - val_loss: 0.5709
Epoch 40/50
345/345              86s 250ms/step -
acc: 0.9402 - loss: 0.1689 - val_acc: 0.8260 - val_loss: 0.6130
Epoch 41/50
345/345              87s 252ms/step -
acc: 0.9433 - loss: 0.1605 - val_acc: 0.8510 - val_loss: 0.5409
Epoch 42/50
345/345              86s 250ms/step -
acc: 0.9441 - loss: 0.1524 - val_acc: 0.8403 - val_loss: 0.5602
Epoch 43/50
345/345              87s 252ms/step -
acc: 0.9476 - loss: 0.1472 - val_acc: 0.8520 - val_loss: 0.5515
Epoch 44/50
345/345              86s 250ms/step -
acc: 0.9501 - loss: 0.1424 - val_acc: 0.8047 - val_loss: 0.7551
Epoch 45/50
345/345              86s 249ms/step -
acc: 0.9557 - loss: 0.1271 - val_acc: 0.8447 - val_loss: 0.5309
Epoch 46/50
345/345              86s 250ms/step -
acc: 0.9576 - loss: 0.1262 - val_acc: 0.8497 - val_loss: 0.5596
Epoch 47/50
345/345              86s 250ms/step -
acc: 0.9556 - loss: 0.1244 - val_acc: 0.8477 - val_loss: 0.5697
Epoch 48/50
345/345              87s 253ms/step -
acc: 0.9624 - loss: 0.1110 - val_acc: 0.8483 - val_loss: 0.5391
Epoch 49/50
345/345              86s 250ms/step -
acc: 0.9618 - loss: 0.1123 - val_acc: 0.8127 - val_loss: 0.7650
Epoch 50/50
345/345              86s 250ms/step -
acc: 0.9604 - loss: 0.1097 - val_acc: 0.8430 - val_loss: 0.6282
```

2.6 Carregamento do modelo e validação

Carregamento e avaliação do modelo através do valor da accuracy.

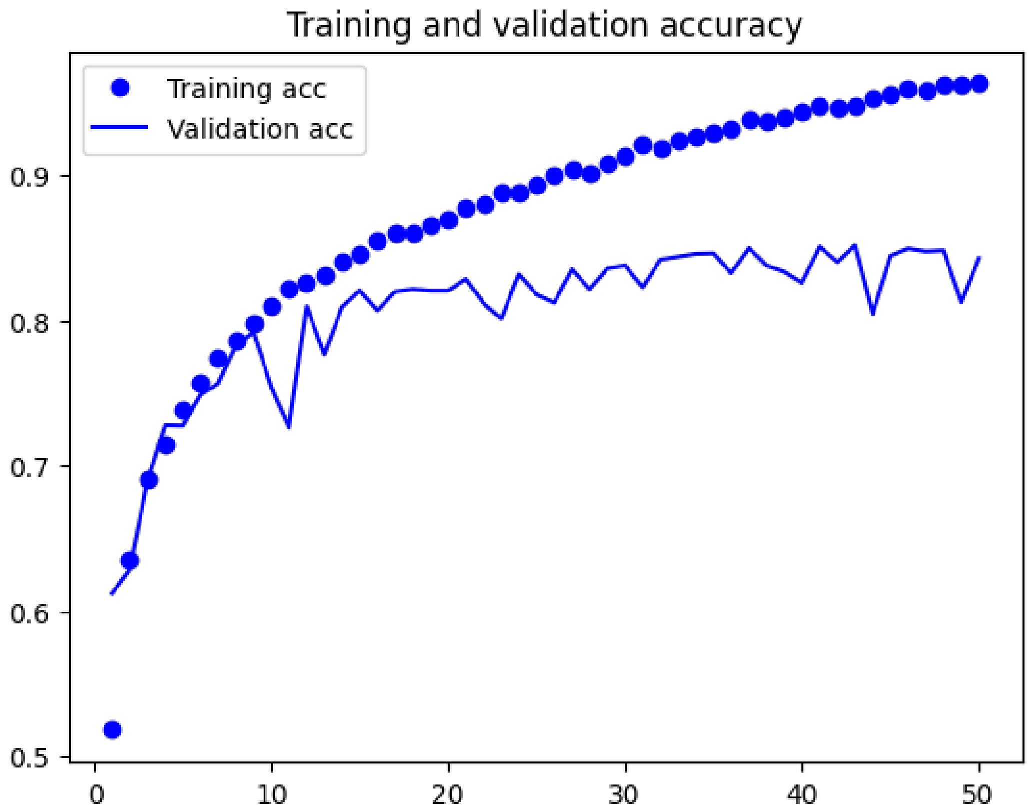
[10]:
```
modelS_CatCross_RMS_dropout_dataAugmentation = keras.models.
 ↪load_model('modelS_CatCross_RMS_dropout_dataAugmentation.keras')
val_loss, val_acc = modelS_CatCross_RMS_dropout_dataAugmentation.
 ↪evaluate(validation_dataset)
print('val_acc:', val_acc)
```

```
I0000 00:00:1749765605.991249  837172 cuda_dnn.cc:529] Loaded cuDNN version
90300
```

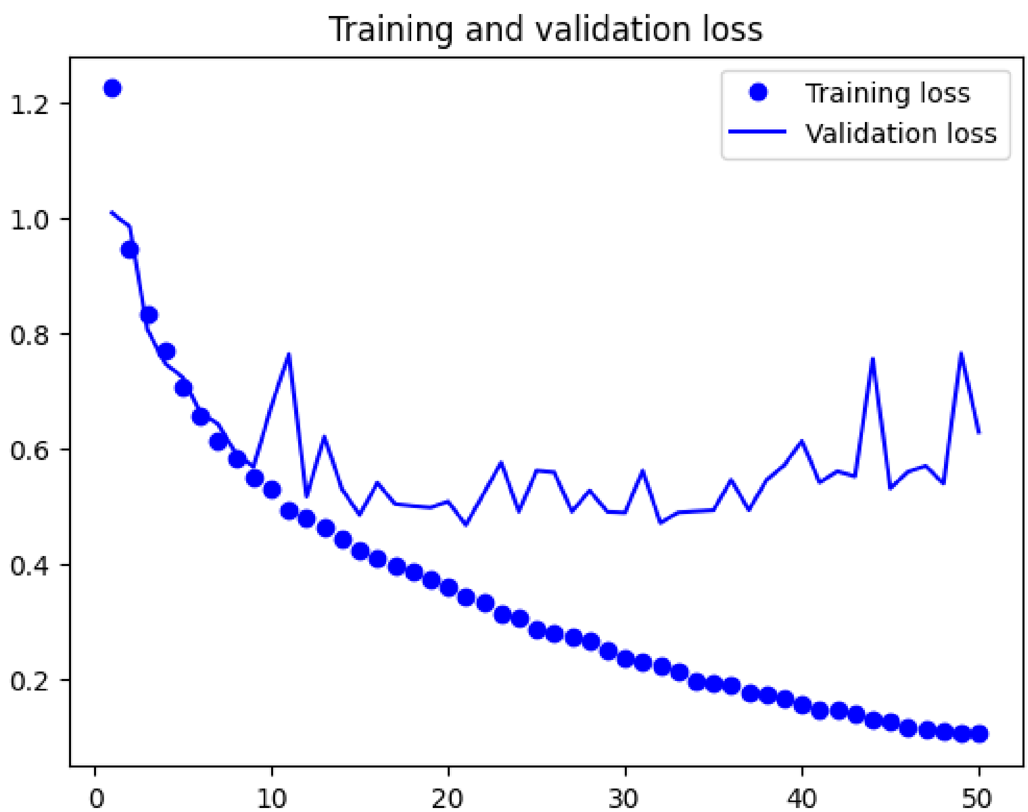
```
94/94            7s 22ms/step - acc:
0.8268 - loss: 0.4771
val_acc: 0.8286666870117188
```

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

[26]:
```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.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
0.9
0.8
0.7
0.6
0.5
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.

[6]:
```
y_true, y_pred = get_true_pred(modelS_CatCross_RMS_dropout_dataAugmentation,␣
 ↪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))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 y_true, y_pred =␣
 ↪get_true_pred(modelS_CatCross_RMS_dropout_dataAugmentation, test_dataset)
      2 report = classification_report(y_true, y_pred, target_names=class_names ␣
 ↪output_dict=True)
      3 class_only_report = {k: v for k, v in report.items() if k in class_name }
```

```
NameError: name 'modelS_CatCross_RMS_dropout_dataAugmentation' is not defined
```

3 Avaliação do melhor modelo (com data augmentation vs sem data augmentation)

Carregamento de modelo sem data augmentation.

[11]:
```
modelS_CatCross_RMS_dropout = keras.models.
    ↪load_model('modelS_CatCross_RMS_dropout.keras')
```

3.1 Comparação dos modelos utilizando a accuracy

[30]:
```
val_loss_CatCross_RMS_dropout, val_acc_CatCross_RMS_dropout =␣
    ↪modelS_CatCross_RMS_dropout.evaluate(validation_dataset)
val_loss_CatCross_RMS_dropout_DA, val_acc_CatCross_RMS_dropout_DA =␣
    ↪modelS_CatCross_RMS_dropout_dataAugmentation.evaluate(validation_dataset)


print("Validation Accuracy dos modelos:")
print(f"CatCross + RMSprop + Dropout: {val_acc_CatCross_RMS_dropout:.4f}")
print(f"CatCross + RMSprop + Dropout + Data augmentation:␣
    ↪{val_acc_CatCross_RMS_dropout_DA:.4f}")


results = {
    'CatCross_RMS_Dropout': val_acc_CatCross_RMS_dropout,
    'CatCross_RMS_Dropout_DataAugmentation': val_acc_CatCross_RMS_dropout_DA,
}

# Identificar o 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                 10s 106ms/step -
accuracy: 0.8448 - loss: 0.4687
94/94                 7s 79ms/step - acc:
0.8337 - loss: 0.4689
Validation Accuracy dos modelos:
CatCross + RMSprop + Dropout: 0.8457
CatCross + RMSprop + Dropout + Data augmentation: 0.8287

Melhor modelo: CatCross_RMS_Dropout com val_accuracy = 0.8457
```

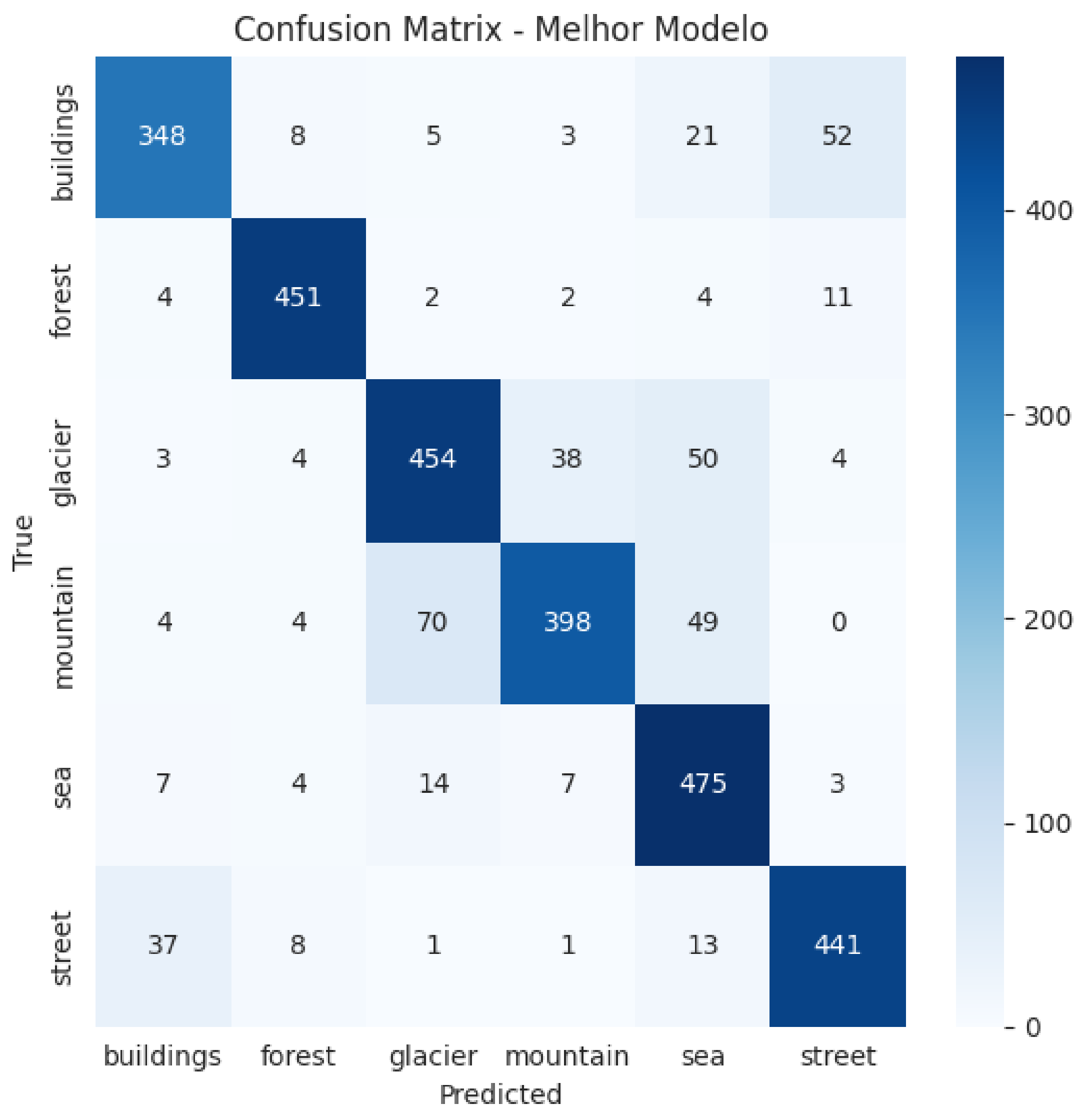
3.2 Matriz de confusão do melhor modelo

[12]:
```
y_true, y_pred = get_true_pred(modelS_CatCross_RMS_dropout, 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()
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1749765682.318136  837170 service.cc:152] XLA service 0x7d1d48003500
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1749765682.318269  837170 service.cc:160]   StreamExecutor device
(0): NVIDIA GeForce GTX 1070, Compute Capability 6.1
2025-06-12 23:01:22.505738: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
2025-06-12 23:01:23.225525: 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$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1_1/conv2d_4_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 23:01:23.289071: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.13 = (f32[1,64,72,72]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,32,74,74]{3,2,1,0} %bitcast.278, f32[64,32,3,3]{3,2,1,0}
%bitcast.285, f32[64]{0} %bitcast.287), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1_1/conv2d_5_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 23:01:23.344866: 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$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1_1/conv2d_6_1/convolution"
source_file="/home/diogo/.pyenv/versions/3.10.18/lib/python3.10/site-
packages/tensorflow/python/framework/ops.py" source_line=1200}, backend_config={
"operation_queue_id":"0","wait_on_operation_queues":[],"cudnn_conv_backend_confi
g":{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,"leakyr
elu_alpha":0},"force_earliest_schedule":false}
2025-06-12 23:01:23.407194: I
external/local_xla/xla/service/gpu/autotuning/conv_algorithm_picker.cc:549]
Omitted potentially buggy algorithm eng14{} for conv %cudnn-conv-bias-
activation.15 = (f32[1,128,15,15]{3,2,1,0}, u8[0]{0}) custom-
call(f32[1,128,17,17]{3,2,1,0} %bitcast.308, f32[128,128,3,3]{3,2,1,0}
%bitcast.315, f32[128]{0} %bitcast.317), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
metadata={op_type="Conv2D" op_name="functional_1_1/conv2d_7_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}
I0000 00:00:1749765683.783482  837170 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
2025-06-12 23:05:50.215519: I tensorflow/core/framework/local_rendezvous.cc:407]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```



3.3 Calcular saída do modelo para uma imagem

[13]:
```
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')
```

```python
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_dropout.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})")
```

Imagem de Teste

```
1/1              0s 39ms/step
Probabilidades por classe:
 buildings: 0.0069
    forest: 0.0004
   glacier: 0.0563
```

```
  mountain: 0.0155
       sea: 0.9200
    street: 0.0009

Classe prevista: sea (0.9200)
```