In [29]:
```
import tensorflow as tf
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from keras import layers
from keras import models
from tensorflow.keras import optimizers
from sklearn.metrics import classification_report
import numpy as np
```

In [30]:
```
tf.__version__
```

Out[30]: '2.19.0'

In [31]:
```
train_dir = 'dataset_balanceado_final/train'
validation_dir = 'dataset_balanceado_final/validation'
test_dir = 'dataset_balanceado_final/test'
```

Definir batch_size e image_size

In [32]:
```
from tensorflow.keras.utils import image_dataset_from_directory

IMG_SIZE = 150
BATCH_SIZE = 32
```

Training set - É o conjunto de dados usado para treinar a rede

In [33]:
```
train_dataset = image_dataset_from_directory(
    train_dir,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    label_mode='categorical'
)
```

```
Found 4276 files belonging to 7 classes.
```

Validation set - Usado para 'testar' o modelo durante o processo de procura da melhor combinação de hiperparâmetros.

In [34]:
```
validation_dataset = image_dataset_from_directory(
    validation_dir,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    label_mode='categorical'
)
```

```
Found 1420 files belonging to 7 classes.
```

Test set - Usado para testar o modelo depois do processo de treino

In [35]:
```
test_dataset = image_dataset_from_directory(
    test_dir,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    label_mode='categorical'
)
```

```
Found 1420 files belonging to 7 classes.
```

Metricas para avaliar os modelos

In [36]:
```
# Utiliza uma função(do sckicit-learn) para avaliar o desempenho do modelo, indi
    # f1-score do modelo
    # accuracy do modelo
    # accuracy por classe

from sklearn.metrics import classification_report
import numpy as np

def print_classification_metrics(model, dataset, phase_name):
    y_true = []
    y_pred = []

    for images, labels in dataset:
        preds = model.predict(images)
        y_true.extend(np.argmax(labels.numpy(), axis=1))
        y_pred.extend(np.argmax(preds, axis=1))

    print(f"\n {phase_name}")
    print(classification_report(y_true, y_pred, digits=4))
```

MODELO 2 (VGG16)

Feature Extraction

Data Augmentation

In [37]:
```
from tensorflow import keras
from keras import layers
from keras.applications import VGG16 # Importa a arquitetura VGG16 pré-treinada

# Data augmentation
data_augmentation_vgg16 = keras.Sequential([
    layers.RandomFlip("vertical"),# Aplica inversão vertical nas imagens.
    layers.RandomFlip("horizontal"), # Aplica inversão horizontal nas imagens
    layers.RandomTranslation(0.1, 0.2),
    layers.RandomRotation(0.4), # Aplica rotação de 20%
    #layers.RandomZoom(0.1)
    ]) # Aplica zoom de 20%

# Carregar a base VGG16 pré-treinada
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(150, 150,
conv_base.trainable = False # Congela todas as camadas da VGG16, impedindo que o

# Usar data augmentation
inputs = layers.Input(shape=(150, 150, 3)) # Define a camada de entrada do novo
x = data_augmentation_vgg16(inputs) # Aplica as transformações de aumento de daa
x = keras.applications.vgg16.preprocess_input(x) # Aplica o pré-processamento es
x = conv_base(x) # Passa as imagens (pré-processadas e aumentadas) através da ba
x = layers.Flatten()(x) # Achata as características extraídas para um vetor 1D.
x = layers.Dense(256, activation='relu')(x) # Adiciona uma camada densa com 256
```

```
x = layers.Dropout(0.5)(x) # Aplica Dropout (50%) para regularização e prevenção
outputs = layers.Dense(7, activation='softmax')(x) # Adiciona a camada de saída

model_t = models.Model(inputs, outputs) # Cria o model_t

# Compilar e treinar (feature extraction)
model_t.compile( # Compila o modelo para configurar o processo de treino.
    loss='categorical_crossentropy', # Define a função de loss categorical cross
    optimizer=keras.optimizers.Adam(learning_rate=1e-3), # Configura o otimizado
    metrics=['accuracy'] # Define 'accuracy' (precisão) como a métrica a ser mon
)

history_t = model_t.fit( # Treina o modelo.
    train_dataset, # Usa o conjunto de dados de treino.
    validation_data=validation_dataset, # Usa o conjunto de dados de validação p
    epochs=15 # Treina o modelo por 10 épocas.
)
```

```
Epoch 1/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 18s 122ms/step - accuracy: 0.3233 - loss: 5.1769 - v
al_accuracy: 0.4718 - val_loss: 1.4558
Epoch 2/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 121ms/step - accuracy: 0.4248 - loss: 1.5484 - v
al_accuracy: 0.5134 - val_loss: 1.2832
Epoch 3/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 121ms/step - accuracy: 0.4333 - loss: 1.4760 - v
al_accuracy: 0.5303 - val_loss: 1.2436
Epoch 4/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 121ms/step - accuracy: 0.4524 - loss: 1.4186 - v
al_accuracy: 0.5824 - val_loss: 1.1935
Epoch 5/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 121ms/step - accuracy: 0.4836 - loss: 1.3557 - v
al_accuracy: 0.5577 - val_loss: 1.2682
Epoch 6/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.4890 - loss: 1.3777 - v
al_accuracy: 0.5880 - val_loss: 1.1366
Epoch 7/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5043 - loss: 1.2932 - v
al_accuracy: 0.5866 - val_loss: 1.1227
Epoch 8/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5086 - loss: 1.2801 - v
al_accuracy: 0.6190 - val_loss: 1.0998
Epoch 9/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5255 - loss: 1.2515 - v
al_accuracy: 0.6289 - val_loss: 1.0766
Epoch 10/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5218 - loss: 1.2273 - v
al_accuracy: 0.6014 - val_loss: 1.1036
Epoch 11/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5098 - loss: 1.2513 - v
al_accuracy: 0.5986 - val_loss: 1.1445
Epoch 12/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 122ms/step - accuracy: 0.5276 - loss: 1.1916 - v
al_accuracy: 0.6120 - val_loss: 1.1432
Epoch 13/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 123ms/step - accuracy: 0.5396 - loss: 1.2117 - v
al_accuracy: 0.6148 - val_loss: 1.1390
Epoch 14/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 123ms/step - accuracy: 0.5551 - loss: 1.2063 - v
al_accuracy: 0.6239 - val_loss: 1.0785
Epoch 15/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 16s 123ms/step - accuracy: 0.5538 - loss: 1.1864 - v
al_accuracy: 0.6239 - val_loss: 1.1262
```

In [38]:
```
print_classification_metrics(model_t, test_dataset, "Modelo 2 : VGG16 (Feature E
```

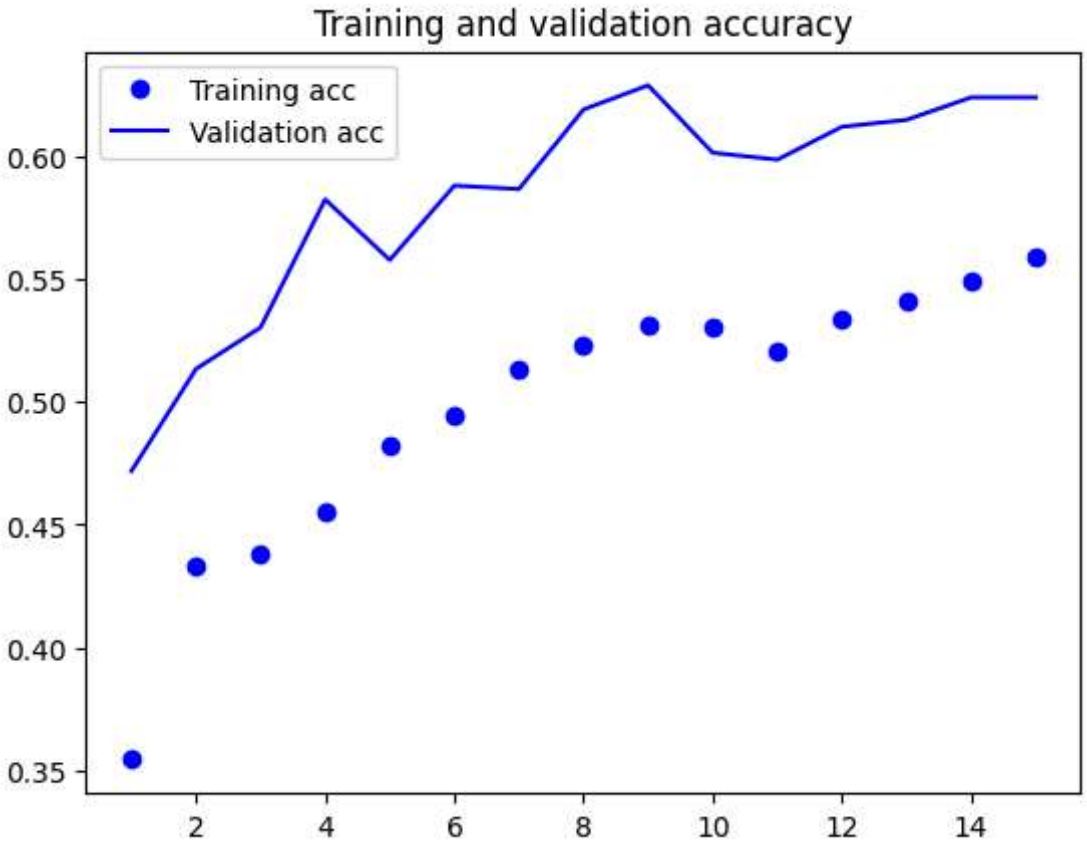
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step
```

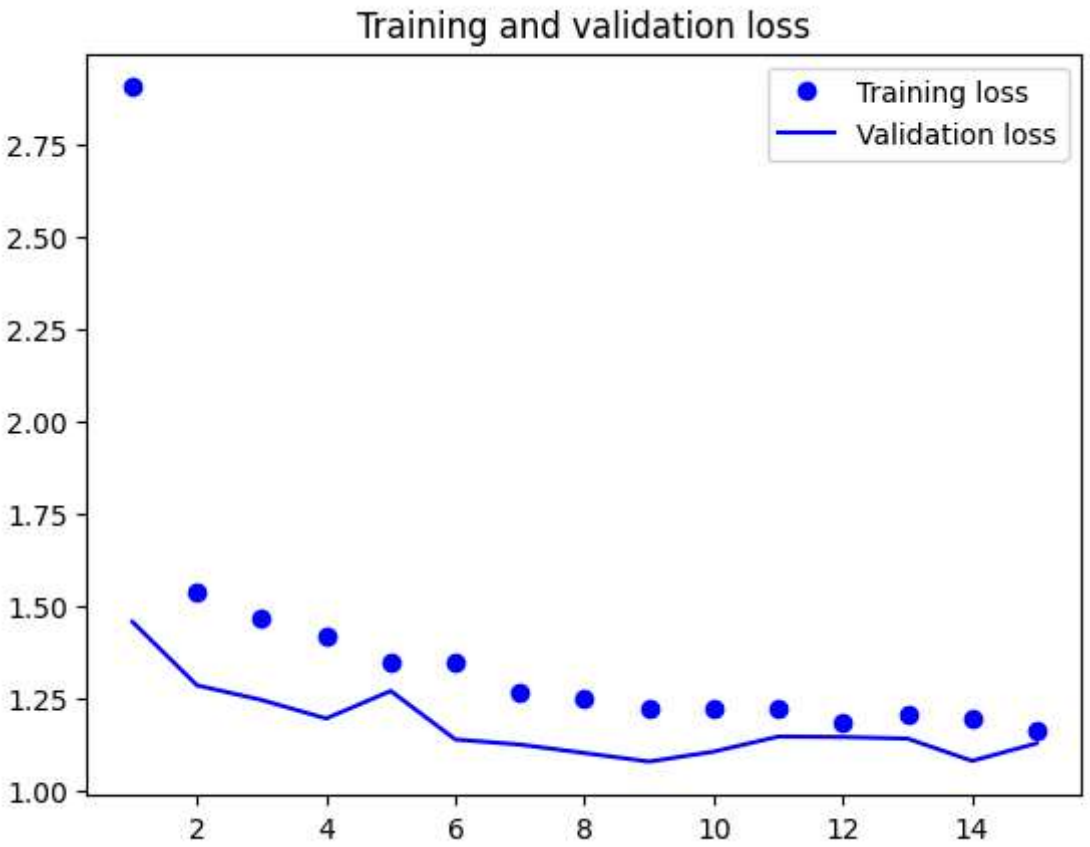
Modelo 2 : VGG16 (Feature Extraction com Augmentation)

	precision	recall	f1-score	support
0	0.7105	0.4154	0.5243	195
1	0.6807	0.5707	0.6209	198
2	0.4857	0.3881	0.4315	219
3	0.6885	0.6562	0.6720	192
4	0.4049	0.5180	0.4545	222
5	0.5166	0.7800	0.6215	200
6	0.9490	0.9588	0.9538	194
accuracy			0.6070	1420
macro avg	0.6337	0.6125	0.6112	1420

```
weighted avg     0.6262    0.6070    0.6049      1420
```

In [39]:
```
import matplotlib.pyplot as plt
accuracy = history_t.history['accuracy']
val_acc = history_t.history['val_accuracy']
loss = history_t.history['loss']
val_loss = history_t.history['val_loss']
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, '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()
```





Durante a feature extraction, observa-se uma subida constante na acurácia de treino (0.35 para 0.56) e validação (0.47 para 0.62), com curvas próximas e paralelas. Esse padrão indica que o modelo está a aprender de forma gradual e equilibrada, com boa capacidade de generalização.

As perdas de treino e validação diminuem progressivamente, com valores bastante próximos (1.0–1.5), o que sugere que o modelo está a convergir bem e sem sinais de overfitting nesta etapa inicial.

Fine Tuning

In [40]:
```
# Descongelar parte da VGG16 (últimas camadas)
conv_base.trainable = True
for layer in conv_base.layers[:-4]:  # Descongelar as ultimas -8
    layer.trainable = False # manter as primeiras camadas congeladas

# Recompilar o modelo com learning rate menor
model_t.compile(
    loss='categorical_crossentropy',
    optimizer=keras.optimizers.Adam(learning_rate=1e-5),
    metrics=['accuracy']
)

# Treinar novamente
history_t = model_t.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=15
)
```

```
Epoch 1/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 21s 141ms/step - accuracy: 0.5550 - loss: 1.1736 - v
al_accuracy: 0.6535 - val_loss: 1.0091
Epoch 2/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 139ms/step - accuracy: 0.5877 - loss: 1.0485 - v
al_accuracy: 0.6613 - val_loss: 1.0168
Epoch 3/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6067 - loss: 1.0131 - v
al_accuracy: 0.6493 - val_loss: 1.0757
Epoch 4/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6187 - loss: 0.9626 - v
al_accuracy: 0.6634 - val_loss: 1.0168
Epoch 5/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6246 - loss: 0.9596 - v
al_accuracy: 0.6634 - val_loss: 1.0332
Epoch 6/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6427 - loss: 0.9297 - v
al_accuracy: 0.6690 - val_loss: 0.9782
Epoch 7/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6357 - loss: 0.9442 - v
al_accuracy: 0.6908 - val_loss: 0.9736
Epoch 8/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6476 - loss: 0.8951 - v
al_accuracy: 0.6930 - val_loss: 0.9054
Epoch 9/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6414 - loss: 0.9262 - v
al_accuracy: 0.6634 - val_loss: 1.0338
Epoch 10/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6565 - loss: 0.8900 - v
al_accuracy: 0.6824 - val_loss: 0.9270
Epoch 11/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6637 - loss: 0.8588 - v
al_accuracy: 0.6739 - val_loss: 0.9859
Epoch 12/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6578 - loss: 0.8427 - v
al_accuracy: 0.6873 - val_loss: 0.9574
Epoch 13/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6667 - loss: 0.8453 - v
al_accuracy: 0.6796 - val_loss: 1.0026
Epoch 14/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6788 - loss: 0.8244 - v
al_accuracy: 0.6930 - val_loss: 0.9527
Epoch 15/15
134/134 ━━━━━━━━━━━━━━━━━━━━ 19s 140ms/step - accuracy: 0.6898 - loss: 0.7990 - v
al_accuracy: 0.6965 - val_loss: 0.9479
```

In [41]:
```
print_classification_metrics(model_t, test_dataset, "Modelo 2 : Fine-tuning")
```

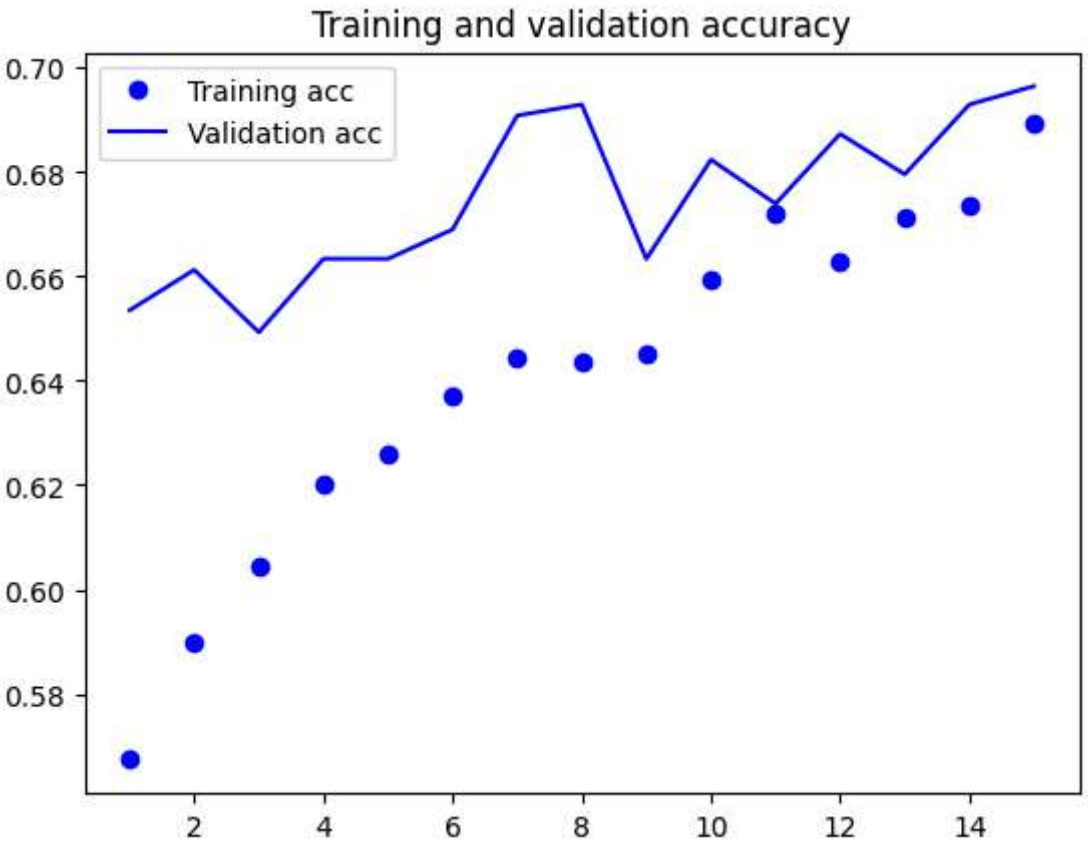
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 219ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 121ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step

 Modelo 2 : Fine-tuning
              precision    recall  f1-score   support

           0     0.7883    0.5538    0.6506       195
           1     0.8298    0.5909    0.6903       198
           2     0.5090    0.6484    0.5703       219
           3     0.8557    0.8958    0.8753       192
           4     0.5402    0.6351    0.5839       222
           5     0.7163    0.7450    0.7304       200
           6     0.9793    0.9742    0.9767       194

    accuracy                         0.7169      1420
   macro avg     0.7455    0.7205    0.7254      1420
```

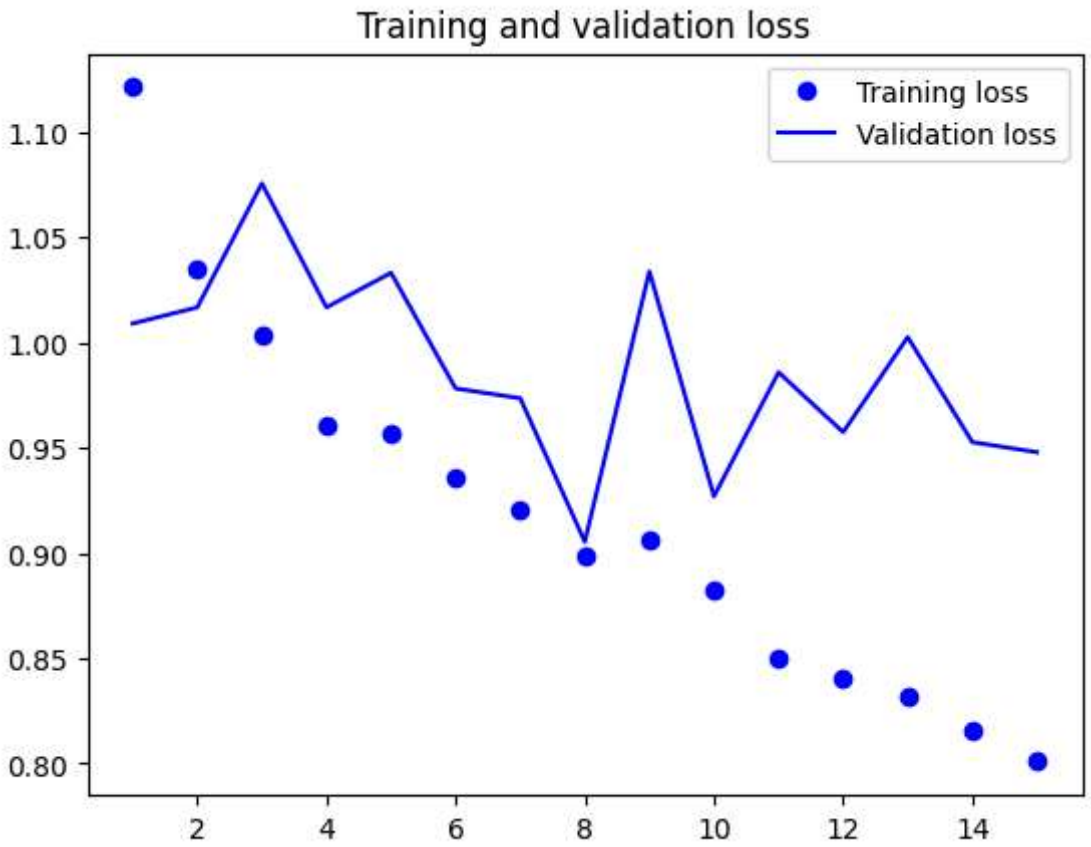
```
weighted avg      0.7373    0.7169    0.7195       1420
```

```
2025-06-12 01:51:08.115895: I tensorflow/core/framework/local_rendezvous.cc:407]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

Curvas de Loss e de Accuracy

In [42]:
```
import matplotlib.pyplot as plt
accuracy = history_t.history['accuracy']
val_acc = history_t.history['val_accuracy']
loss = history_t.history['loss']
val_loss = history_t.history['val_loss']
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, '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()
```





No fine-tuning, as acurácias de treino e validação aumentam de forma consistente, alcançando cerca de 0.69. As curvas estão próximas, o que indica que o modelo está a melhorar seu desempenho sem sinais evidentes de overfitting, o que reflete um ajuste mais equilibrado.

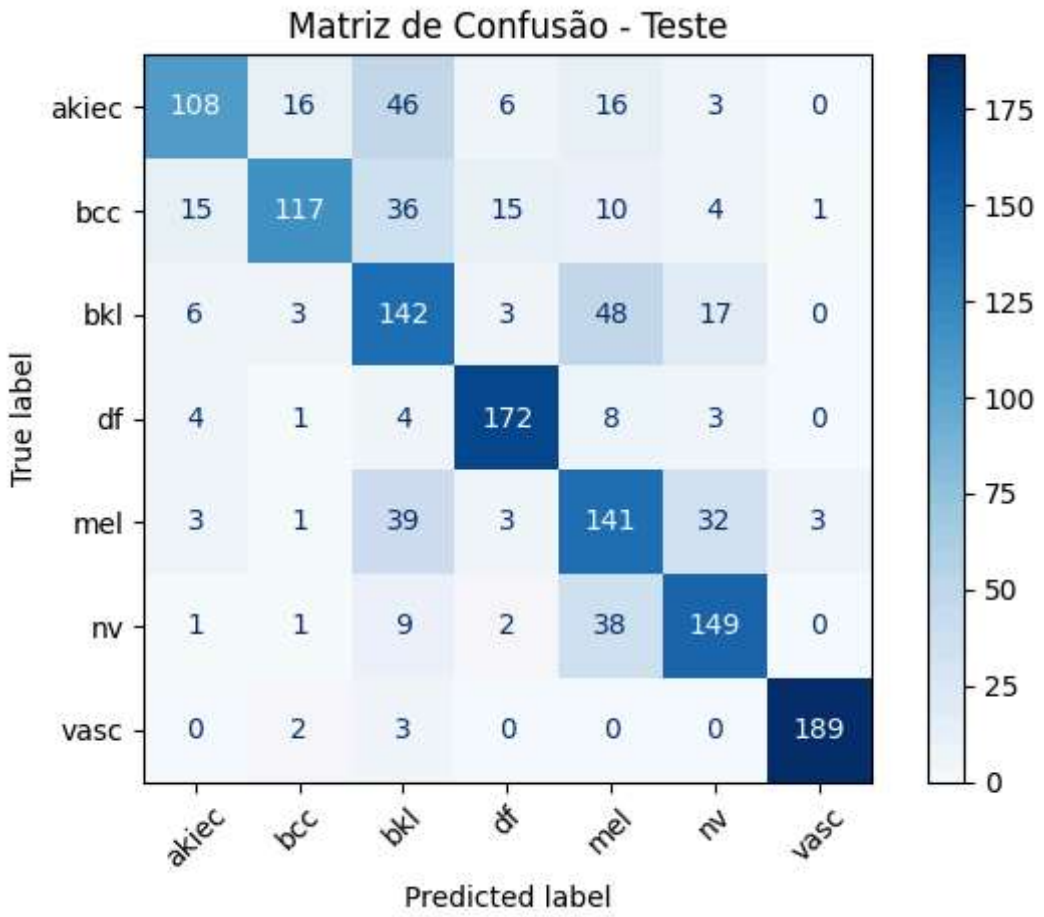
As perdas de treino e validação diminuem ao longo das épocas. Apesar de alguma oscilação na loss de validação, ambas as curvas seguem uma tendência de queda e permanecem próximas, sugerindo convergência estável e boa generalização durante o tuning.

Matriz de Confusão

In [ ]:
```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Obter predições no test_dataset
y_true = []
y_pred = []

for images, labels in test_dataset:
    preds = model_t.predict(images)
    y_true.extend(np.argmax(labels.numpy(), axis=1))
    y_pred.extend(np.argmax(preds, axis=1))

class_names = test_dataset.class_names

# Criar e mostrar a matriz de confusão
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
```

```
plt.figure(figsize=(10, 8))
disp.plot(cmap=plt.cm.Blues, xticks_rotation=45, values_format='d')
plt.title("Matriz de Confusão - Teste")
plt.tight_layout()
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 121ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 114ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
<Figure size 1000x800 with 0 Axes>
```




```
The Kernel crashed while executing code in the current cell or a previous cell.

Please review the code in the cell(s) to identify a possible cause of the failur
e.

Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more info.

View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.
```

O modelo apresenta bom desempenho geral, com alta taxa de acerto nas classes 'df', 'nv' e 'vasc', que foram corretamente classificadas na maioria dos casos. No entanto, há confusão significativa entre as classes 'akiec', 'bcc' e 'bkl', o que sugere que o modelo ainda tem dificuldade em distinguir essas lesões, possivelmente devido à semelhança visual entre elas.

Salvar o Modelo

In [44]:
```
model_t.save("modelT_3A_com_data_aug_adam_cat_cross_best_acc.keras")
```

Definir a Função para obter Computed Features

In [45]:
```
import numpy as np
from tensorflow.keras.applications.vgg16 import preprocess_input

def extract_computed_features(conv_base, dataset, augment_layer=None):
    all_features = []
    all_labels = []
```

```
    for images, labels in dataset:
        # Aplica data augmentation se fornecido
        if augment_layer:
            images = augment_layer(images)

        # Pre-processamento obrigatório para VGG16
        preprocessed = preprocess_input(images)

        # Extrair as computed features
        features = conv_base.predict(preprocessed)

        all_features.append(features)
        all_labels.append(labels)

    # Junta tudo num único array
    features_array = np.concatenate(all_features)
    labels_array = np.concatenate(all_labels)

    return features_array, labels_array
```

Usar a funçao "extract_computed_features"

In [46]:
```
train_features, train_labels = extract_computed_features(conv_base, train_datase
val_features, val_labels = extract_computed_features(conv_base, validation_datas
test_features, test_labels = extract_computed_features(conv_base, test_dataset,
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 518ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 129ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 492ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 138ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 354ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step
```

Salvar as Computed Features

In [47]:
```python
from numpy import save

save("modelt_3A_train_features.npy", train_features)
save("modelt_3A_train_labels.npy", train_labels)

save("modelt_3A_val_features.npy", val_features)
save("modelt_3A_val_labels.npy", val_labels)

save("modelt_3A_test_features.npy", test_features)
save("modelt_3A_test_labels.npy", test_labels)
```