lplyeqkk1

June 19, 2024

[1]:
```python
import os, shutil
train_dir = 'C:/Users/flavi/Desktop/Projeto-20240530/train'
validation_dir = 'C:/Users/flavi/Desktop/Projeto-20240530/validation'
test_dir = 'C:/Users/flavi/Desktop/Projeto-20240530/test'
```

[2]:
```python
from keras.utils import image_dataset_from_directory
IMG_SIZE = 150
train_dataset = image_dataset_from_directory(
train_dir,
image_size=(IMG_SIZE, IMG_SIZE),
batch_size=32)
validation_dataset = image_dataset_from_directory(
validation_dir,
image_size=(IMG_SIZE, IMG_SIZE),
batch_size=32)
test_dataset = image_dataset_from_directory(
test_dir,
image_size=(IMG_SIZE, IMG_SIZE),
batch_size=32)
```

```
Found 40000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
Found 10000 files belonging to 10 classes.
```

[3]:
```python
from tensorflow import keras
from keras import layers
from keras import models
from keras.preprocessing import image

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",␣
 ↪padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)
```

```
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu",␣
↪padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=128, kernel_size=3, activation="relu",␣
↪padding='same')(x)
x = layers.BatchNormalization()(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(512, activation="relu")(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

[4]:
```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

[5]:
```
from keras.callbacks import ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=2,
    min_lr=0.001
)
```

[6]:
```
from keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)
```

[7]:
```
from keras.callbacks import ModelCheckpoint

model_checkpoint = ModelCheckpoint(
    filepath='C:/Users/flavi/Desktop/projetoClassificaoDeImagens/
↪dl_project_2201707_2211044/ModelosS/ModelS_AdamOptimizer.keras',
    save_best_only=True,
    monitor='val_loss'
)
```

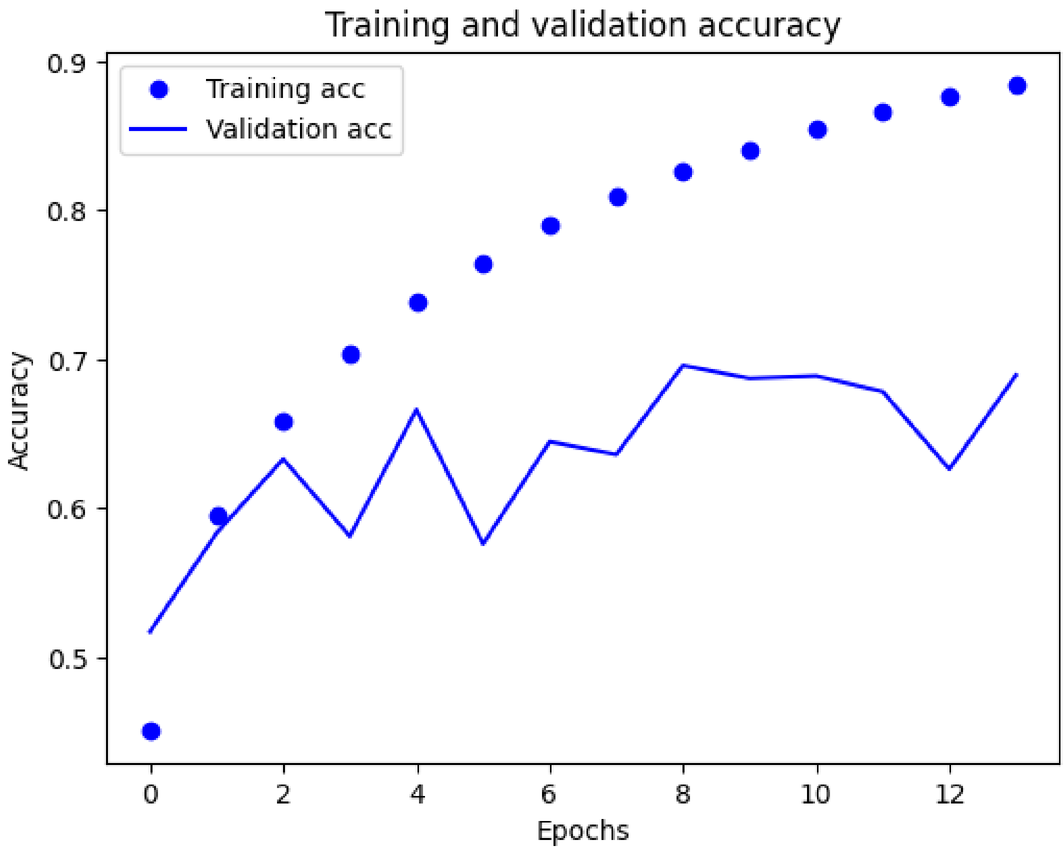
[8]:
```python
callbacks = [reduce_lr, early_stopping, model_checkpoint]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks
)
```

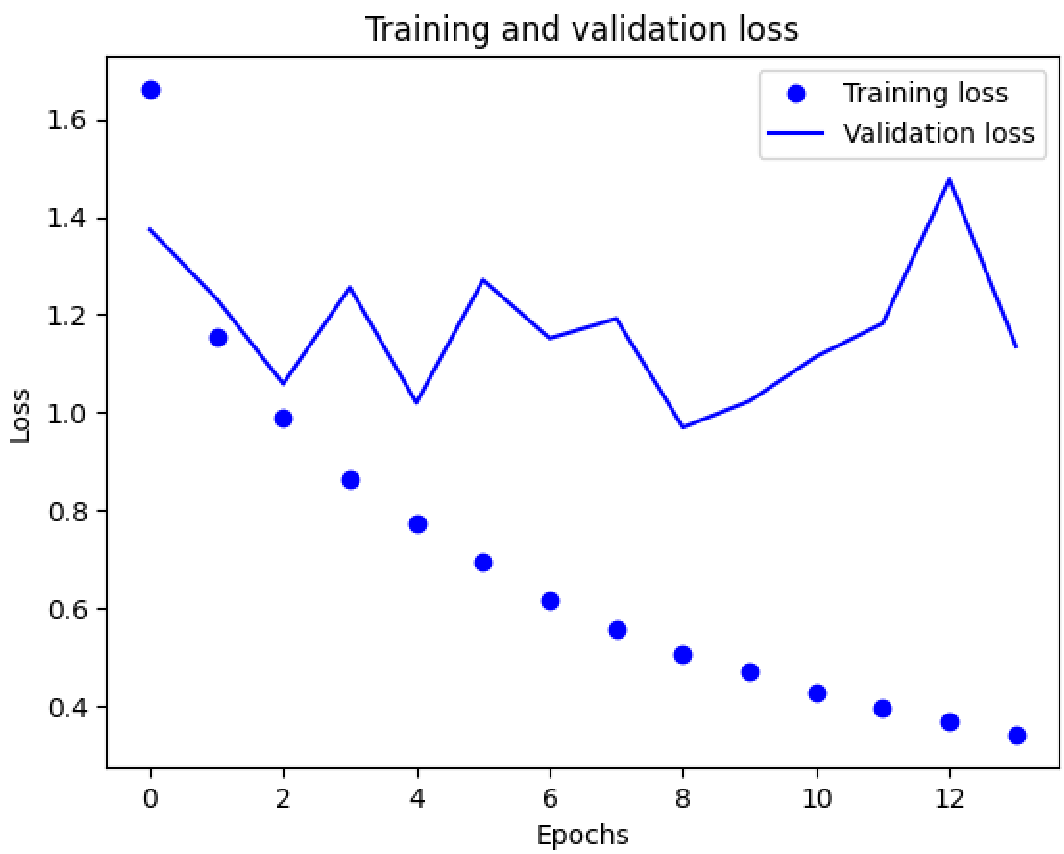
```
Epoch 1/30
1250/1250                673s 537ms/step
- accuracy: 0.3705 - loss: 2.4561 - val_accuracy: 0.5171 - val_loss: 1.3737 -
learning_rate: 0.0010
Epoch 2/30
1250/1250                685s 548ms/step
- accuracy: 0.5677 - loss: 1.2209 - val_accuracy: 0.5833 - val_loss: 1.2325 -
learning_rate: 0.0010
Epoch 3/30
1250/1250                685s 548ms/step
- accuracy: 0.6413 - loss: 1.0304 - val_accuracy: 0.6330 - val_loss: 1.0583 -
learning_rate: 0.0010
Epoch 4/30
1250/1250                711s 569ms/step
- accuracy: 0.6905 - loss: 0.8979 - val_accuracy: 0.5812 - val_loss: 1.2554 -
learning_rate: 0.0010
Epoch 5/30
1250/1250                686s 549ms/step
- accuracy: 0.7301 - loss: 0.7969 - val_accuracy: 0.6662 - val_loss: 1.0196 -
learning_rate: 0.0010
Epoch 6/30
1250/1250                685s 548ms/step
- accuracy: 0.7556 - loss: 0.7171 - val_accuracy: 0.5760 - val_loss: 1.2709 -
learning_rate: 0.0010
Epoch 7/30
1250/1250                681s 545ms/step
- accuracy: 0.7785 - loss: 0.6461 - val_accuracy: 0.6446 - val_loss: 1.1514 -
learning_rate: 0.0010
Epoch 8/30
1250/1250                671s 537ms/step
- accuracy: 0.8016 - loss: 0.5742 - val_accuracy: 0.6360 - val_loss: 1.1917 -
learning_rate: 0.0010
Epoch 9/30
1250/1250                678s 542ms/step
- accuracy: 0.8210 - loss: 0.5195 - val_accuracy: 0.6958 - val_loss: 0.9695 -
learning_rate: 0.0010
Epoch 10/30
1250/1250                698s 559ms/step
- accuracy: 0.8358 - loss: 0.4802 - val_accuracy: 0.6871 - val_loss: 1.0233 -
```

```
learning_rate: 0.0010
Epoch 11/30
1250/1250              680s 544ms/step
- accuracy: 0.8490 - loss: 0.4394 - val_accuracy: 0.6887 - val_loss: 1.1140 -
learning_rate: 0.0010
Epoch 12/30
1250/1250              686s 549ms/step
- accuracy: 0.8634 - loss: 0.4103 - val_accuracy: 0.6782 - val_loss: 1.1822 -
learning_rate: 0.0010
Epoch 13/30
1250/1250              1179s
944ms/step - accuracy: 0.8764 - loss: 0.3702 - val_accuracy: 0.6261 - val_loss:
1.4763 - learning_rate: 0.0010
Epoch 14/30
1250/1250              1404s 1s/step -
accuracy: 0.8839 - loss: 0.3423 - val_accuracy: 0.6893 - val_loss: 1.1352 -
learning_rate: 0.0010
```

[9]:
```python
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], 'bo', label='Training acc')
plt.plot(history.history['val_accuracy'], 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], 'bo', label='Training loss')
plt.plot(history.history['val_loss'], 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
0.9
0.8
0.7
0.6
0.5
0
2
4
6
8
10
12



[10]:
```
val_loss, val_acc = model.evaluate(validation_dataset)
print('Validation Accuracy:', val_acc)
```

```
313/313              59s 189ms/step -
accuracy: 0.6903 - loss: 0.9989
Validation Accuracy: 0.6958000063896179
```

[11]:
```
loss, accuracy = model.evaluate(test_dataset)
print(f"Loss: {loss}, Accuracy: {accuracy}")
```

```
313/313              60s 191ms/step -
accuracy: 0.6954 - loss: 0.9368
Loss: 0.9532666802406311, Accuracy: 0.695900022983551
```

[14]:
```
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Function to evaluate the model and get true and predicted labels
```

```python
def evaluate_model(model, dataset):
    all_labels = []
    all_predictions = []

    for images, labels in dataset:
        predictions = model.predict(images)
        predicted_labels = np.argmax(predictions, axis=1)
        true_labels = labels.numpy()  # Convert to numpy array if not already

        all_labels.extend(true_labels)
        all_predictions.extend(predicted_labels)

    return np.array(all_labels), np.array(all_predictions)

# Get true and predicted labels for the test dataset
true_labels, predicted_labels = evaluate_model(model, test_dataset)

# Compute the confusion matrix
conf_matrix = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
 ↪xticklabels=range(10), yticklabels=range(10))
plt.title('Matriz de Confusão')
plt.xlabel('Previsão')
plt.ylabel('Realidade')
plt.show()

# Print classification report
class_names = [str(i) for i in range(10)]  # Define class names based on your
 ↪dataset
print(classification_report(true_labels, predicted_labels,
 ↪target_names=class_names))

# Extract precision, recall, and F1-score for each class from classification
 ↪report
report = classification_report(true_labels, predicted_labels,
 ↪target_names=class_names, output_dict=True)

metrics = {'precision': [], 'recall': [], 'f1-score': []}
for cls in class_names:
    metrics['precision'].append(report[cls]['precision'])
    metrics['recall'].append(report[cls]['recall'])
    metrics['f1-score'].append(report[cls]['f1-score'])

# Plot precision, recall, and F1-score
```

```python
plt.figure(figsize=(10, 6))
bar_width = 0.2
index = np.arange(len(class_names))

plt.bar(index, metrics['precision'], bar_width, label='Precision')
plt.bar(index + bar_width, metrics['recall'], bar_width, label='Recall')
plt.bar(index + 2*bar_width, metrics['f1-score'], bar_width, label='F1-score')

plt.xlabel('Class')
plt.ylabel('Scores')
plt.title('Precision, Recall e F1-score para cada classe')
plt.xticks(index + bar_width, class_names)
plt.legend()

plt.tight_layout()
plt.show()
```

```
1/1          0s 378ms/step
1/1          0s 290ms/step
1/1          0s 273ms/step
1/1          0s 243ms/step
1/1          0s 362ms/step
1/1          0s 247ms/step
1/1          0s 253ms/step
1/1          0s 337ms/step
1/1          0s 272ms/step
1/1          0s 258ms/step
1/1          0s 310ms/step
1/1          0s 307ms/step
1/1          0s 344ms/step
1/1          0s 355ms/step
1/1          0s 249ms/step
1/1          0s 241ms/step
1/1          0s 310ms/step
1/1          0s 363ms/step
1/1          0s 211ms/step
1/1          0s 272ms/step
1/1          0s 252ms/step
1/1          0s 378ms/step
1/1          0s 249ms/step
1/1          0s 265ms/step
1/1          0s 240ms/step
1/1          0s 398ms/step
1/1          0s 334ms/step
1/1          0s 216ms/step
1/1          0s 242ms/step
1/1          0s 372ms/step
```

```
1/1              0s 193ms/step
1/1              0s 261ms/step
1/1              0s 305ms/step
1/1              0s 298ms/step
1/1              0s 183ms/step
1/1              0s 282ms/step
1/1              0s 400ms/step
1/1              0s 264ms/step
1/1              0s 340ms/step
1/1              0s 312ms/step
1/1              0s 283ms/step
1/1              0s 259ms/step
1/1              0s 378ms/step
1/1              0s 158ms/step
1/1              0s 260ms/step
1/1              0s 269ms/step
1/1              0s 302ms/step
1/1              0s 269ms/step
1/1              0s 256ms/step
1/1              0s 363ms/step
1/1              0s 334ms/step
1/1              0s 263ms/step
1/1              0s 204ms/step
1/1              0s 278ms/step
1/1              0s 367ms/step
1/1              0s 320ms/step
1/1              0s 310ms/step
1/1              0s 273ms/step
1/1              0s 237ms/step
1/1              0s 363ms/step
1/1              0s 279ms/step
1/1              0s 229ms/step
1/1              0s 286ms/step
1/1              0s 425ms/step
1/1              0s 441ms/step
1/1              0s 236ms/step
1/1              0s 226ms/step
1/1              0s 369ms/step
1/1              0s 305ms/step
1/1              0s 257ms/step
1/1              0s 312ms/step
1/1              0s 177ms/step
1/1              0s 247ms/step
1/1              0s 474ms/step
1/1              0s 423ms/step
1/1              0s 215ms/step
1/1              0s 262ms/step
1/1              0s 348ms/step
```

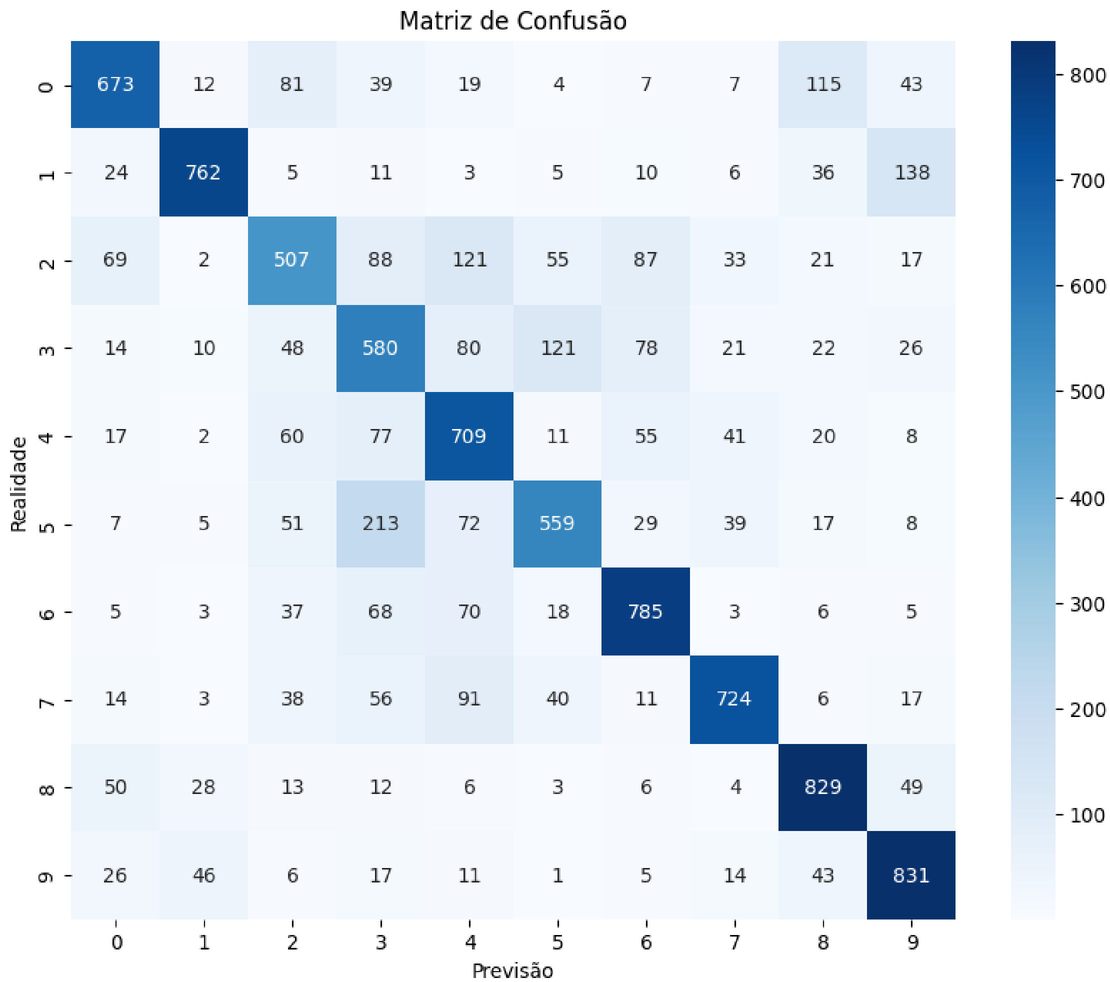
```
1/1                0s 318ms/step
1/1                0s 144ms/step
1/1                0s 251ms/step
1/1                0s 362ms/step
1/1                0s 167ms/step
1/1                0s 246ms/step
1/1                0s 363ms/step
1/1                0s 348ms/step
1/1                0s 172ms/step
1/1                0s 251ms/step
1/1                0s 414ms/step
1/1                0s 369ms/step
1/1                0s 311ms/step
1/1                0s 266ms/step
1/1                0s 314ms/step
1/1                0s 317ms/step
1/1                0s 357ms/step
1/1                0s 179ms/step
1/1                0s 258ms/step
1/1                0s 360ms/step
1/1                0s 287ms/step
1/1                0s 294ms/step
1/1                0s 250ms/step
1/1                0s 453ms/step
1/1                0s 466ms/step
1/1                0s 252ms/step
1/1                0s 289ms/step
1/1                0s 303ms/step
1/1                0s 368ms/step
1/1                0s 265ms/step
1/1                0s 260ms/step
1/1                0s 317ms/step
1/1                0s 338ms/step
1/1                0s 189ms/step
1/1                0s 221ms/step
1/1                0s 356ms/step
1/1                0s 311ms/step
1/1                0s 167ms/step
1/1                0s 258ms/step
1/1                0s 244ms/step
1/1                0s 357ms/step
1/1                0s 235ms/step
1/1                0s 315ms/step
1/1                0s 322ms/step
1/1                0s 355ms/step
1/1                0s 190ms/step
1/1                0s 254ms/step
1/1                0s 352ms/step
```

```
1/1            0s 175ms/step
1/1            0s 259ms/step
1/1            0s 296ms/step
1/1            1s 507ms/step
1/1            0s 392ms/step
1/1            0s 268ms/step
1/1            0s 244ms/step
1/1            0s 320ms/step
1/1            0s 367ms/step
1/1            0s 317ms/step
1/1            0s 165ms/step
1/1            0s 321ms/step
1/1            0s 318ms/step
1/1            0s 266ms/step
1/1            0s 224ms/step
1/1            1s 511ms/step
1/1            0s 425ms/step
1/1            0s 282ms/step
1/1            0s 164ms/step
1/1            0s 299ms/step
1/1            0s 364ms/step
1/1            0s 422ms/step
1/1            0s 270ms/step
1/1            0s 260ms/step
1/1            0s 313ms/step
1/1            0s 360ms/step
1/1            0s 313ms/step
1/1            0s 183ms/step
1/1            0s 331ms/step
1/1            0s 373ms/step
1/1            0s 404ms/step
1/1            0s 190ms/step
1/1            0s 271ms/step
1/1            0s 327ms/step
1/1            0s 163ms/step
1/1            0s 323ms/step
1/1            0s 401ms/step
1/1            0s 288ms/step
1/1            0s 221ms/step
1/1            0s 352ms/step
1/1            0s 262ms/step
1/1            0s 374ms/step
1/1            0s 333ms/step
1/1            0s 180ms/step
1/1            0s 247ms/step
1/1            0s 301ms/step
1/1            0s 309ms/step
1/1            0s 273ms/step
```

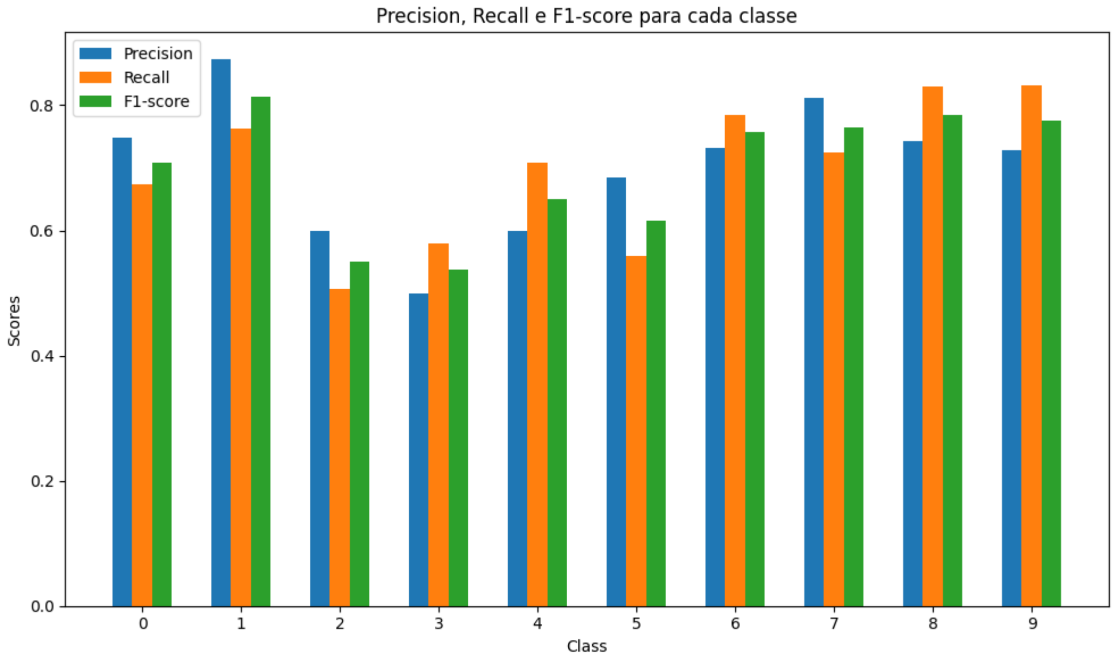
```
1/1                 0s 362ms/step
1/1                 0s 378ms/step
1/1                 0s 298ms/step
1/1                 0s 191ms/step
1/1                 0s 283ms/step
1/1                 0s 347ms/step
1/1                 0s 192ms/step
1/1                 0s 249ms/step
1/1                 0s 425ms/step
1/1                 0s 187ms/step
1/1                 0s 301ms/step
1/1                 0s 248ms/step
1/1                 0s 384ms/step
1/1                 0s 287ms/step
1/1                 0s 242ms/step
1/1                 0s 316ms/step
1/1                 0s 353ms/step
1/1                 0s 303ms/step
1/1                 0s 242ms/step
1/1                 0s 317ms/step
1/1                 0s 282ms/step
1/1                 0s 242ms/step
1/1                 0s 356ms/step
1/1                 0s 245ms/step
1/1                 0s 244ms/step
1/1                 0s 260ms/step
1/1                 0s 395ms/step
1/1                 0s 361ms/step
1/1                 0s 185ms/step
1/1                 0s 260ms/step
1/1                 0s 203ms/step
1/1                 0s 307ms/step
1/1                 0s 229ms/step
1/1                 0s 260ms/step
1/1                 0s 313ms/step
1/1                 0s 257ms/step
1/1                 0s 206ms/step
1/1                 0s 234ms/step
1/1                 0s 406ms/step
1/1                 0s 345ms/step
1/1                 0s 262ms/step
1/1                 0s 259ms/step
1/1                 0s 345ms/step
1/1                 0s 338ms/step
1/1                 0s 174ms/step
1/1                 0s 261ms/step
1/1                 0s 207ms/step
1/1                 0s 165ms/step
```

```
1/1              0s 272ms/step
1/1              0s 230ms/step
1/1              0s 276ms/step
1/1              0s 156ms/step
1/1              0s 285ms/step
1/1              0s 325ms/step
1/1              0s 303ms/step
1/1              0s 323ms/step
1/1              0s 150ms/step
1/1              0s 260ms/step
1/1              0s 422ms/step
1/1              0s 366ms/step
1/1              0s 261ms/step
1/1              0s 249ms/step
1/1              0s 332ms/step
1/1              0s 286ms/step
1/1              0s 260ms/step
1/1              0s 279ms/step
1/1              0s 301ms/step
1/1              0s 364ms/step
1/1              0s 332ms/step
1/1              0s 188ms/step
1/1              0s 250ms/step
1/1              0s 337ms/step
1/1              0s 210ms/step
1/1              0s 266ms/step
1/1              0s 240ms/step
1/1              0s 421ms/step
1/1              0s 315ms/step
1/1              0s 175ms/step
1/1              0s 261ms/step
1/1              0s 174ms/step
1/1              0s 350ms/step
1/1              0s 322ms/step
1/1              0s 170ms/step
1/1              0s 240ms/step
1/1              0s 299ms/step
1/1              0s 213ms/step
1/1              0s 237ms/step
1/1              0s 396ms/step
1/1              0s 322ms/step
1/1              0s 210ms/step
1/1              0s 261ms/step
1/1              0s 243ms/step
1/1              0s 339ms/step
1/1              0s 198ms/step
1/1              0s 281ms/step
1/1              0s 249ms/step
```

```
1/1                0s 384ms/step
1/1                0s 426ms/step
1/1                0s 457ms/step
1/1                0s 374ms/step
1/1                0s 279ms/step
1/1                0s 273ms/step
1/1                0s 315ms/step
1/1                0s 353ms/step
1/1                0s 211ms/step
1/1                0s 253ms/step
1/1                0s 321ms/step
1/1                0s 387ms/step
1/1                0s 278ms/step
1/1                0s 213ms/step
1/1                0s 328ms/step
1/1                0s 317ms/step
1/1                0s 285ms/step
1/1                0s 272ms/step
1/1                0s 318ms/step
1/1                0s 378ms/step
1/1                0s 297ms/step
1/1                0s 242ms/step
1/1                0s 259ms/step
1/1                0s 166ms/step
1/1                0s 310ms/step
1/1                0s 373ms/step
1/1                0s 311ms/step
1/1                0s 169ms/step
1/1                0s 256ms/step
1/1                0s 341ms/step
1/1                0s 356ms/step
1/1                0s 297ms/step
1/1                0s 231ms/step
1/1                0s 360ms/step
1/1                0s 422ms/step
1/1                0s 300ms/step
1/1                0s 189ms/step
1/1                0s 243ms/step
1/1                0s 323ms/step
1/1                0s 269ms/step
1/1                0s 265ms/step
1/1                0s 290ms/step
1/1                0s 326ms/step
```



	precision	recall	f1-score	support
0	0.75	0.67	0.71	1000
1	0.87	0.76	0.81	1000
2	0.60	0.51	0.55	1000
3	0.50	0.58	0.54	1000
4	0.60	0.71	0.65	1000
5	0.68	0.56	0.62	1000
6	0.73	0.79	0.76	1000
7	0.81	0.72	0.77	1000
8	0.74	0.83	0.78	1000
9	0.73	0.83	0.78	1000
accuracy			0.70	10000
macro avg	0.70	0.70	0.70	10000
weighted avg	0.70	0.70	0.70	10000


Precision, Recall e F1-score para cada classe
Precision
Recall
F1-score
Scores
0.0
0.2
0.4
0.6
0.8
Class
0
1
2
3
4
5
6
7
8
9