mrphn4b9i

June 19, 2024

[1]:
```
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]:
```
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]:
```
from tensorflow import keras
from keras import layers
from keras import models
from keras.preprocessing import image

data_augmentation = keras.Sequential(
    [
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
    ]
)

inputs = keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
```

```
x = data_augmentation(inputs)
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='sgd',
    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_SGDOptimizerComData.keras',
    save_best_only=True,
    monitor='val_loss'
)
```

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

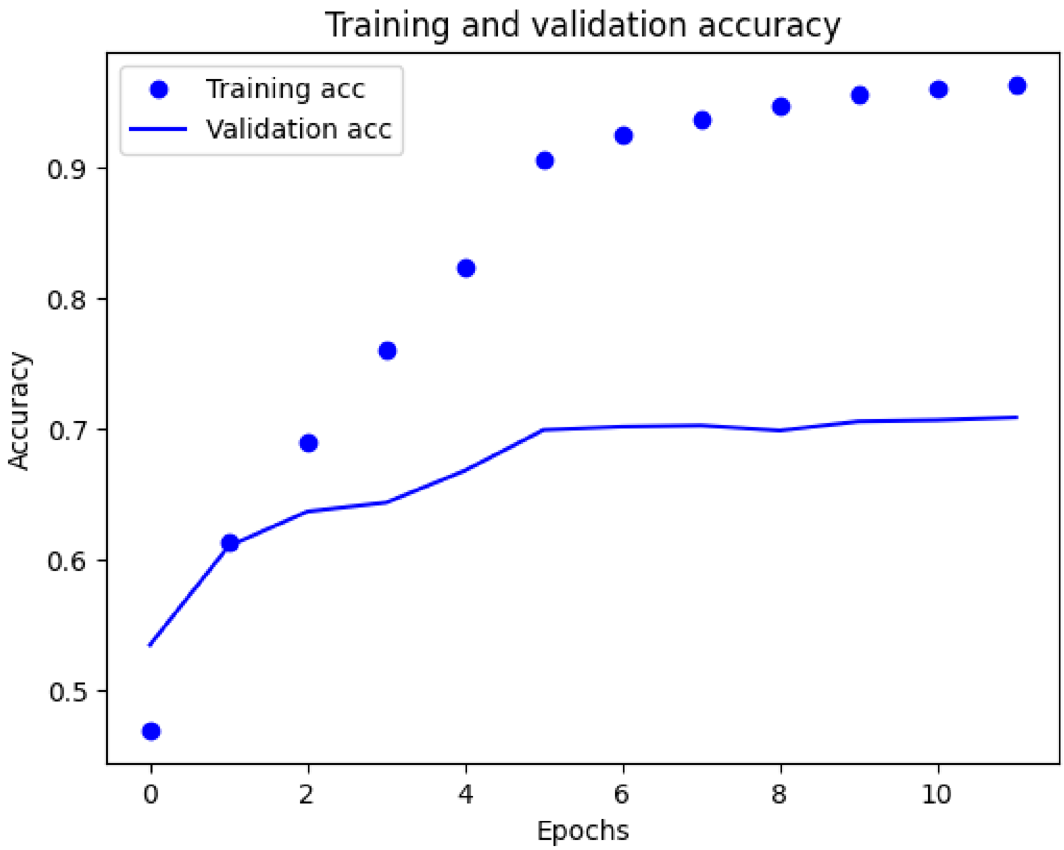
```
Epoch 1/30
1250/1250            542s 433ms/step
- accuracy: 0.3991 - loss: 1.9175 - val_accuracy: 0.5346 - val_loss: 1.2933 -
learning_rate: 0.0100
Epoch 2/30
1250/1250            535s 428ms/step
- accuracy: 0.5934 - loss: 1.1395 - val_accuracy: 0.6101 - val_loss: 1.1168 -
learning_rate: 0.0100
Epoch 3/30
1250/1250            534s 427ms/step
- accuracy: 0.6758 - loss: 0.9134 - val_accuracy: 0.6366 - val_loss: 1.0462 -
learning_rate: 0.0100
Epoch 4/30
1250/1250            532s 426ms/step
- accuracy: 0.7444 - loss: 0.7164 - val_accuracy: 0.6434 - val_loss: 1.1034 -
learning_rate: 0.0100
Epoch 5/30
1250/1250            536s 429ms/step
- accuracy: 0.8133 - loss: 0.5312 - val_accuracy: 0.6679 - val_loss: 1.0957 -
learning_rate: 0.0100
Epoch 6/30
1250/1250            861s 689ms/step
- accuracy: 0.8834 - loss: 0.3384 - val_accuracy: 0.6990 - val_loss: 1.0021 -
learning_rate: 0.0020
Epoch 7/30
1250/1250            1088s
871ms/step - accuracy: 0.9125 - loss: 0.2605 - val_accuracy: 0.7015 - val_loss:
0.9978 - learning_rate: 0.0020
Epoch 8/30
1250/1250            1094s
```

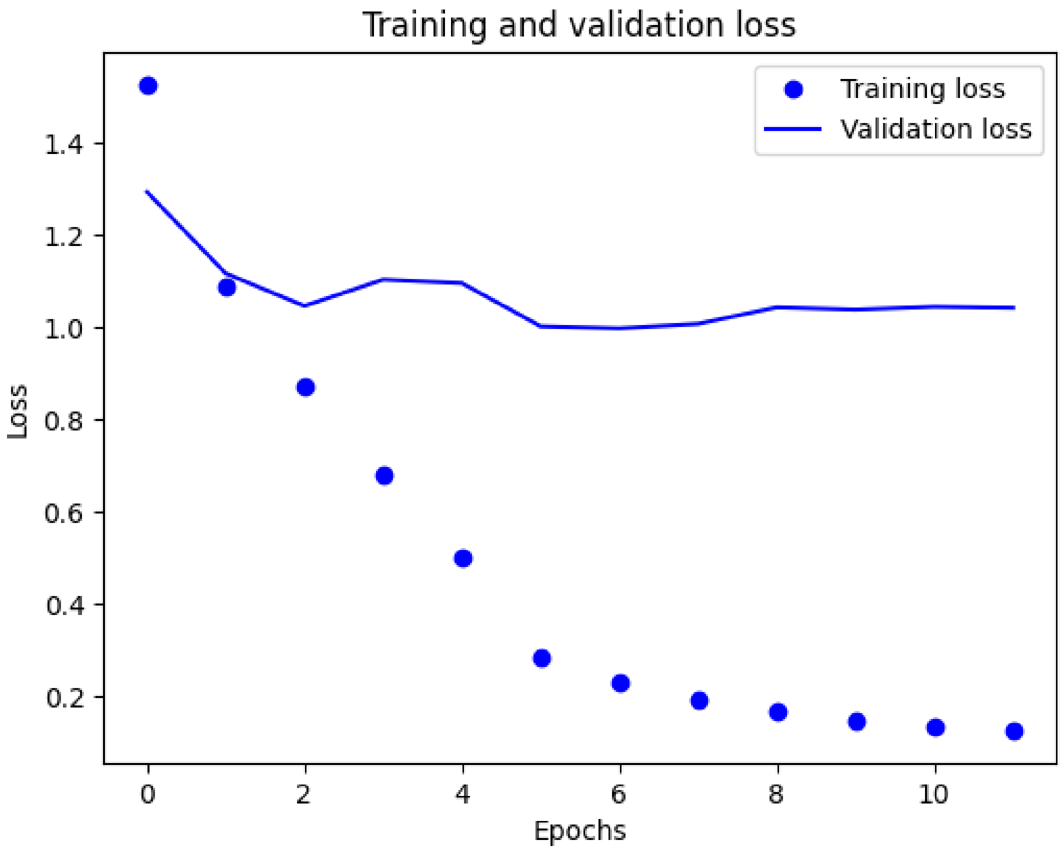
```
875ms/step - accuracy: 0.9282 - loss: 0.2191 - val_accuracy: 0.7024 - val_loss:
1.0075 - learning_rate: 0.0020
Epoch 9/30
1250/1250            1131s
905ms/step - accuracy: 0.9396 - loss: 0.1884 - val_accuracy: 0.6985 - val_loss:
1.0431 - learning_rate: 0.0020
Epoch 10/30
1250/1250            1169s
935ms/step - accuracy: 0.9502 - loss: 0.1643 - val_accuracy: 0.7055 - val_loss:
1.0382 - learning_rate: 0.0010
Epoch 11/30
1250/1250            1160s
928ms/step - accuracy: 0.9528 - loss: 0.1483 - val_accuracy: 0.7065 - val_loss:
1.0444 - learning_rate: 0.0010
Epoch 12/30
1250/1250            1127s
901ms/step - accuracy: 0.9586 - loss: 0.1372 - val_accuracy: 0.7086 - val_loss:
1.0421 - learning_rate: 0.0010
```

[9]:

```
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



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

```
313/313              54s 173ms/step -
accuracy: 0.7053 - loss: 0.9751
Validation Accuracy: 0.7014999985694885
```

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

```
313/313              53s 170ms/step -
accuracy: 0.6966 - loss: 0.9801
Loss: 0.9928498864173889, Accuracy: 0.6980999708175659
```

[12]:
```
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             1s 834ms/step
1/1             0s 308ms/step
1/1             0s 226ms/step
1/1             0s 328ms/step
1/1             0s 218ms/step
1/1             0s 240ms/step
1/1             0s 258ms/step
1/1             0s 287ms/step
1/1             0s 220ms/step
1/1             0s 243ms/step
1/1             0s 264ms/step
1/1             0s 324ms/step
1/1             0s 396ms/step
1/1             0s 253ms/step
1/1             0s 296ms/step
1/1             0s 248ms/step
1/1             0s 180ms/step
1/1             0s 229ms/step
1/1             0s 333ms/step
1/1             0s 395ms/step
1/1             0s 331ms/step
1/1             0s 263ms/step
1/1             0s 271ms/step
1/1             0s 210ms/step
1/1             0s 272ms/step
1/1             0s 278ms/step
1/1             0s 243ms/step
1/1             0s 231ms/step
1/1             0s 220ms/step
1/1             0s 259ms/step
```

```
1/1                0s 336ms/step
1/1                0s 332ms/step
1/1                0s 277ms/step
1/1                0s 261ms/step
1/1                0s 225ms/step
1/1                0s 291ms/step
1/1                0s 304ms/step
1/1                0s 257ms/step
1/1                0s 321ms/step
1/1                0s 293ms/step
1/1                0s 296ms/step
1/1                0s 247ms/step
1/1                0s 228ms/step
1/1                0s 292ms/step
1/1                0s 296ms/step
1/1                0s 344ms/step
1/1                0s 290ms/step
1/1                0s 279ms/step
1/1                0s 247ms/step
1/1                0s 239ms/step
1/1                0s 273ms/step
1/1                0s 295ms/step
1/1                0s 251ms/step
1/1                0s 252ms/step
1/1                0s 317ms/step
1/1                0s 311ms/step
1/1                0s 306ms/step
1/1                0s 275ms/step
1/1                0s 258ms/step
1/1                0s 295ms/step
1/1                0s 278ms/step
1/1                0s 340ms/step
1/1                0s 429ms/step
1/1                0s 274ms/step
1/1                0s 247ms/step
1/1                0s 251ms/step
1/1                0s 255ms/step
1/1                0s 424ms/step
1/1                0s 352ms/step
1/1                0s 250ms/step
1/1                0s 189ms/step
1/1                0s 287ms/step
1/1                0s 259ms/step
1/1                0s 348ms/step
1/1                0s 268ms/step
1/1                0s 207ms/step
1/1                0s 245ms/step
1/1                0s 247ms/step
```

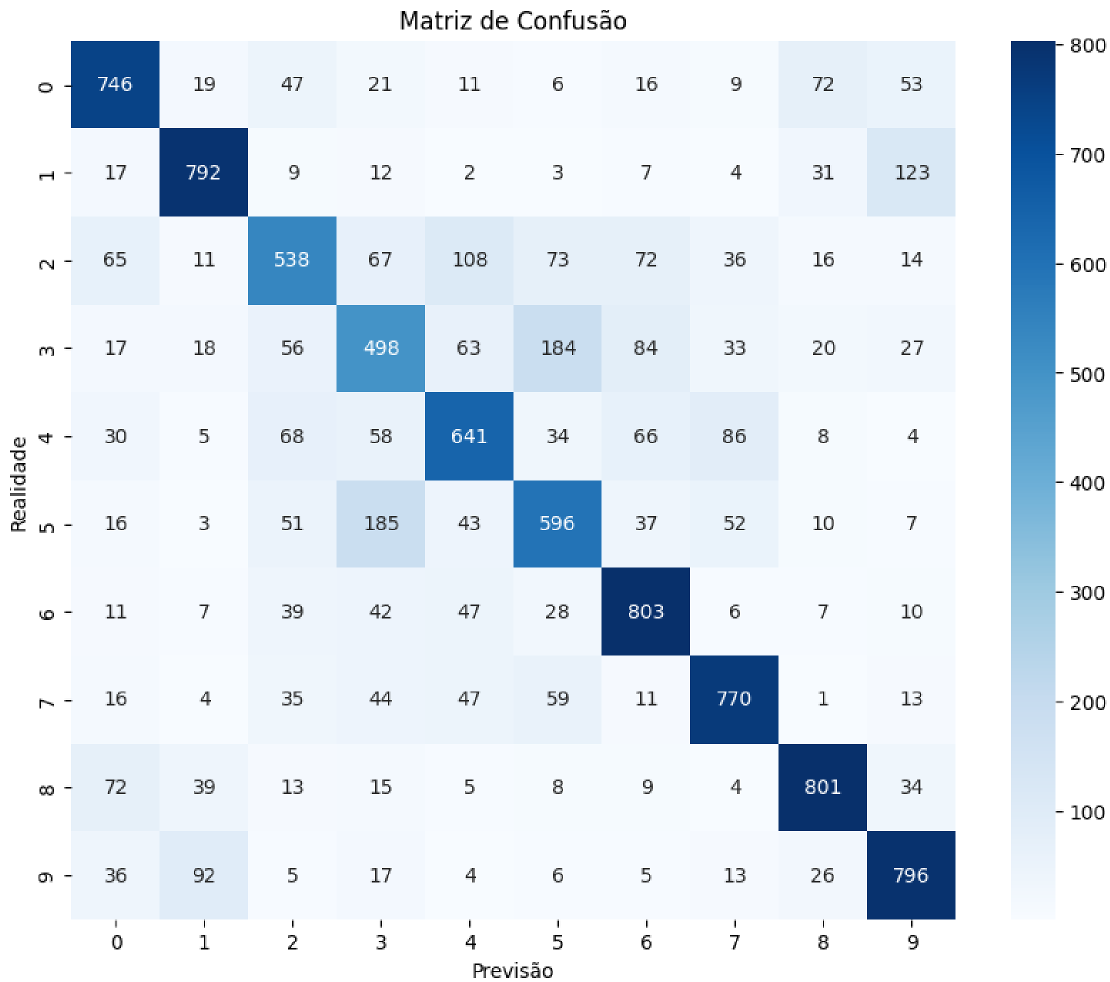
```
1/1                0s 267ms/step
1/1                0s 294ms/step
1/1                0s 283ms/step
1/1                0s 359ms/step
1/1                0s 330ms/step
1/1                0s 308ms/step
1/1                0s 252ms/step
1/1                0s 252ms/step
1/1                0s 320ms/step
1/1                0s 396ms/step
1/1                0s 268ms/step
1/1                0s 271ms/step
1/1                0s 203ms/step
1/1                0s 291ms/step
1/1                0s 260ms/step
1/1                0s 315ms/step
1/1                0s 291ms/step
1/1                0s 231ms/step
1/1                0s 319ms/step
1/1                0s 280ms/step
1/1                0s 258ms/step
1/1                0s 219ms/step
1/1                0s 233ms/step
1/1                0s 257ms/step
1/1                0s 346ms/step
1/1                0s 220ms/step
1/1                0s 228ms/step
1/1                0s 248ms/step
1/1                0s 208ms/step
1/1                0s 277ms/step
1/1                0s 365ms/step
1/1                0s 307ms/step
1/1                0s 239ms/step
1/1                0s 215ms/step
1/1                0s 270ms/step
1/1                0s 342ms/step
1/1                0s 347ms/step
1/1                0s 343ms/step
1/1                0s 242ms/step
1/1                0s 260ms/step
1/1                0s 283ms/step
1/1                0s 291ms/step
1/1                0s 264ms/step
1/1                0s 294ms/step
1/1                0s 253ms/step
1/1                0s 264ms/step
1/1                0s 266ms/step
1/1                0s 267ms/step
```

```
1/1                 0s 263ms/step
1/1                 0s 341ms/step
1/1                 0s 314ms/step
1/1                 0s 267ms/step
1/1                 0s 262ms/step
1/1                 0s 194ms/step
1/1                 0s 258ms/step
1/1                 0s 342ms/step
1/1                 0s 370ms/step
1/1                 0s 305ms/step
1/1                 0s 254ms/step
1/1                 0s 240ms/step
1/1                 0s 279ms/step
1/1                 0s 260ms/step
1/1                 0s 313ms/step
1/1                 0s 238ms/step
1/1                 0s 196ms/step
1/1                 0s 280ms/step
1/1                 0s 326ms/step
1/1                 0s 271ms/step
1/1                 0s 222ms/step
1/1                 0s 200ms/step
1/1                 0s 242ms/step
1/1                 0s 317ms/step
1/1                 0s 327ms/step
1/1                 0s 269ms/step
1/1                 0s 221ms/step
1/1                 0s 278ms/step
1/1                 0s 159ms/step
1/1                 0s 267ms/step
1/1                 0s 304ms/step
1/1                 0s 397ms/step
1/1                 0s 263ms/step
1/1                 0s 217ms/step
1/1                 0s 217ms/step
1/1                 0s 236ms/step
1/1                 0s 260ms/step
1/1                 0s 256ms/step
1/1                 0s 197ms/step
1/1                 0s 308ms/step
1/1                 0s 280ms/step
1/1                 0s 249ms/step
1/1                 0s 253ms/step
1/1                 0s 265ms/step
1/1                 0s 206ms/step
1/1                 0s 265ms/step
1/1                 0s 316ms/step
1/1                 0s 268ms/step
```

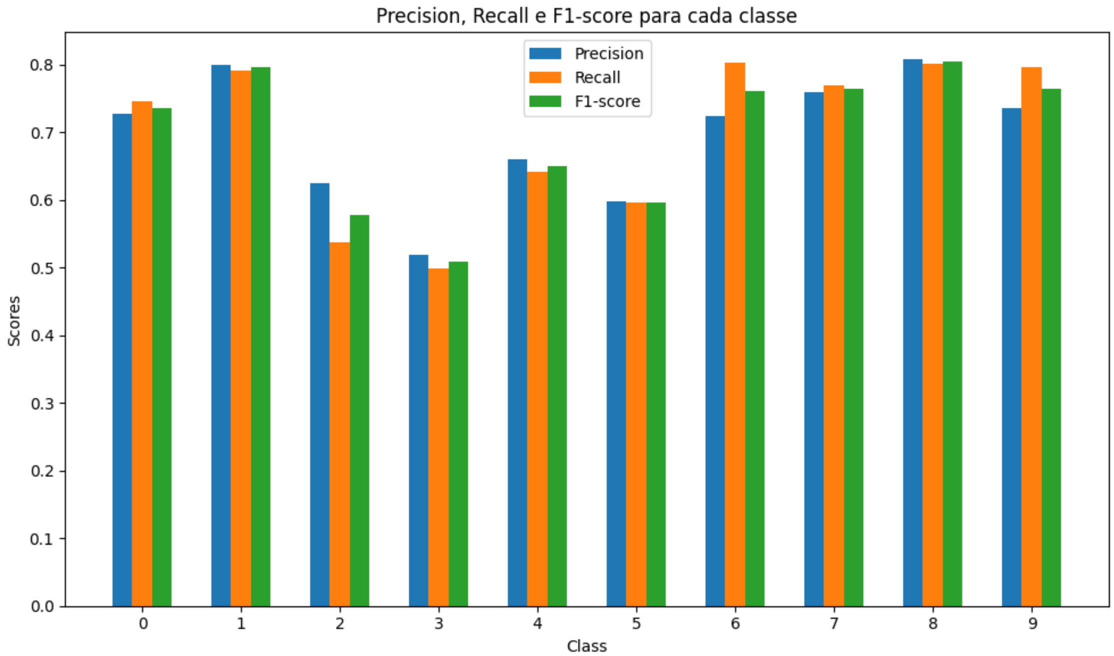
```
1/1                0s 205ms/step
1/1                0s 267ms/step
1/1                0s 284ms/step
1/1                0s 327ms/step
1/1                0s 283ms/step
1/1                0s 285ms/step
1/1                0s 235ms/step
1/1                0s 217ms/step
1/1                0s 301ms/step
1/1                0s 300ms/step
1/1                0s 317ms/step
1/1                0s 267ms/step
1/1                0s 234ms/step
1/1                0s 251ms/step
1/1                0s 321ms/step
1/1                0s 315ms/step
1/1                0s 250ms/step
1/1                0s 270ms/step
1/1                0s 253ms/step
1/1                0s 263ms/step
1/1                0s 295ms/step
1/1                0s 257ms/step
1/1                0s 221ms/step
1/1                0s 170ms/step
1/1                0s 169ms/step
1/1                0s 266ms/step
1/1                0s 305ms/step
1/1                0s 254ms/step
1/1                0s 208ms/step
1/1                0s 228ms/step
1/1                0s 228ms/step
1/1                0s 223ms/step
1/1                0s 207ms/step
1/1                0s 227ms/step
1/1                0s 302ms/step
1/1                0s 172ms/step
1/1                0s 392ms/step
1/1                0s 247ms/step
1/1                0s 217ms/step
1/1                0s 264ms/step
1/1                0s 295ms/step
1/1                0s 315ms/step
1/1                0s 239ms/step
1/1                0s 186ms/step
1/1                0s 254ms/step
1/1                0s 304ms/step
1/1                0s 402ms/step
1/1                0s 237ms/step
```

```
1/1          0s 257ms/step
1/1          0s 250ms/step
1/1          0s 269ms/step
1/1          0s 322ms/step
1/1          0s 318ms/step
1/1          0s 239ms/step
1/1          0s 159ms/step
1/1          0s 289ms/step
1/1          0s 310ms/step
1/1          0s 323ms/step
1/1          0s 342ms/step
1/1          0s 319ms/step
1/1          0s 252ms/step
1/1          0s 203ms/step
1/1          0s 249ms/step
1/1          0s 315ms/step
1/1          0s 234ms/step
1/1          0s 153ms/step
1/1          0s 249ms/step
1/1          0s 252ms/step
1/1          0s 306ms/step
1/1          0s 333ms/step
1/1          0s 227ms/step
1/1          0s 164ms/step
1/1          0s 268ms/step
1/1          0s 312ms/step
1/1          0s 249ms/step
1/1          0s 308ms/step
1/1          0s 224ms/step
1/1          0s 230ms/step
1/1          0s 296ms/step
1/1          0s 352ms/step
1/1          0s 323ms/step
1/1          0s 253ms/step
1/1          0s 208ms/step
1/1          0s 200ms/step
1/1          0s 243ms/step
1/1          0s 340ms/step
1/1          0s 239ms/step
1/1          0s 220ms/step
1/1          0s 244ms/step
1/1          0s 232ms/step
1/1          0s 320ms/step
1/1          0s 178ms/step
1/1          0s 317ms/step
1/1          0s 282ms/step
1/1          0s 209ms/step
1/1          0s 253ms/step
```

```
1/1                0s 244ms/step
1/1                0s 299ms/step
1/1                0s 242ms/step
1/1                0s 213ms/step
1/1                0s 219ms/step
1/1                0s 257ms/step
1/1                0s 240ms/step
1/1                0s 211ms/step
1/1                0s 171ms/step
1/1                0s 239ms/step
1/1                0s 280ms/step
1/1                0s 228ms/step
1/1                0s 184ms/step
1/1                0s 267ms/step
1/1                0s 290ms/step
1/1                0s 257ms/step
1/1                0s 261ms/step
1/1                0s 176ms/step
1/1                0s 222ms/step
1/1                0s 291ms/step
1/1                0s 300ms/step
1/1                0s 218ms/step
1/1                0s 205ms/step
1/1                0s 255ms/step
1/1                0s 237ms/step
1/1                0s 259ms/step
1/1                0s 262ms/step
1/1                0s 183ms/step
1/1                0s 231ms/step
1/1                0s 228ms/step
1/1                0s 251ms/step
1/1                0s 240ms/step
1/1                0s 205ms/step
1/1                0s 217ms/step
1/1                0s 287ms/step
1/1                0s 394ms/step
1/1                0s 259ms/step
1/1                0s 188ms/step
1/1                0s 238ms/step
1/1                0s 305ms/step
1/1                0s 336ms/step
1/1                0s 228ms/step
1/1                1s 509ms/step
```



	precision	recall	f1-score	support
0	0.73	0.75	0.74	1000
1	0.80	0.79	0.80	1000
2	0.62	0.54	0.58	1000
3	0.52	0.50	0.51	1000
4	0.66	0.64	0.65	1000
5	0.60	0.60	0.60	1000
6	0.72	0.80	0.76	1000
7	0.76	0.77	0.77	1000
8	0.81	0.80	0.80	1000
9	0.74	0.80	0.77	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
Class
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6
7
8
9