Modelo S sem data augmentation e optimizer RMSProp

Em todos os documentos deste folder "Optimizer RMSProp" a única diferença dos documentos do folder "Optimizer Adam" é o optimizer a ser utilizado, aqui nós decidimos escolher o optimizer RMSProp.

O treino que realizamos com o RMSProp foi somente nos melhores modelos obtidos com o otimizador ADAM, ou seja, não testamos diferentes metricas (BATCH_SIZE, IMG_SIZE...)

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten,
Dense, Conv2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
CSVLogger, ReduceLROnPlateau
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
# CONSTANTES
BATCH_SIZE = 32
IMG_SIZE = 32
NUM_CLASSES = 10  # nº classes para identificar
NUM_EPOCHS = 60
LEARNING_RATE = 0.001

# Folders do dataset
train_dirs = ['./dataset/train/train1', './dataset/train/train2',
'./dataset/train/train3', './dataset/train/train5']
validation_dir = './dataset/validation'
test_dir = './dataset/test'

# CRIAR OS GERADORES
train_datagen = ImageDataGenerator(rescale=1./255)

validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# training generators
train_generators = [train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
```

```
    class_mode='categorical') for train_dir in train_dirs]

# Necessário para junstar os trainning generators
def combined_generator(generators):
    while True:
        for generator in generators:
            yield next(generator)

train_generator = combined_generator(train_generators)

# Validation e test generators
validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical')
```

```
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
Found 10000 images belonging to 10 classes.
```

```
from tensorflow.keras import backend as K
from tensorflow.keras.metrics import Metric

class Precision(Metric):
    def __init__(self, name='precision', **kwargs):
        super(Precision, self).__init__(name=name, **kwargs)
        self.true_positives = self.add_weight(name='tp',
initializer='zeros')
        self.predicted_positives = self.add_weight(name='pp',
initializer='zeros')

    def update_state(self, y_true, y_pred, sample_weight=None):
        y_pred = K.round(y_pred)
        y_true = K.cast(y_true, 'float32')
        self.true_positives.assign_add(K.sum(y_true * y_pred))
        self.predicted_positives.assign_add(K.sum(y_pred))

    def result(self):
        return self.true_positives / (self.predicted_positives +
K.epsilon())
```

```
    def reset_states(self):
        self.true_positives.assign(0)
        self.predicted_positives.assign(0)

class Recall(Metric):
    def __init__(self, name='recall', **kwargs):
        super(Recall, self).__init__(name=name, **kwargs)
        self.true_positives = self.add_weight(name='tp',
initializer='zeros')
        self.actual_positives = self.add_weight(name='ap',
initializer='zeros')

    def update_state(self, y_true, y_pred, sample_weight=None):
        y_pred = K.round(y_pred)
        y_true = K.cast(y_true, 'float32')
        self.true_positives.assign_add(K.sum(y_true * y_pred))
        self.actual_positives.assign_add(K.sum(y_true))

    def result(self):
        return self.true_positives / (self.actual_positives +
K.epsilon())

    def reset_states(self):
        self.true_positives.assign(0)
        self.actual_positives.assign(0)

class F1Score(Metric):
    def __init__(self, name='f1_score', **kwargs):
        super(F1Score, self).__init__(name=name, **kwargs)
        self.precision = Precision()
        self.recall = Recall()

    def update_state(self, y_true, y_pred, sample_weight=None):
        self.precision.update_state(y_true, y_pred)
        self.recall.update_state(y_true, y_pred)

    def result(self):
        precision = self.precision.result()
        recall = self.recall.result()
        return 2 * ((precision * recall) / (precision + recall +
K.epsilon()))

    def reset_states(self):
        self.precision.reset_states()
        self.recall.reset_states()

model = Sequential([
    Conv2D(128, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    BatchNormalization(),
```

```
    Activation('relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    Conv2D(256, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.5),

    Conv2D(512, (3, 3)),
    BatchNormalization(),
    Activation('relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.5),

    Flatten(),
    Dense(512),
    BatchNormalization(),
    Activation('relu'),
    Dropout(0.5),

    Dense(NUM_CLASSES, activation='softmax')
])

# Compilar o modelo
model.compile(optimizer=RMSprop(learning_rate=LEARNING_RATE),
              loss='categorical_crossentropy',
              metrics=['accuracy', Precision(), Recall(), F1Score()])


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 128)       3584

 batch_normalization (BatchN  (None, 30, 30, 128)      512
 ormalization)

 activation (Activation)     (None, 30, 30, 128)       0

 max_pooling2d (MaxPooling2D  (None, 15, 15, 128)      0
 )

 dropout (Dropout)           (None, 15, 15, 128)       0

 conv2d_1 (Conv2D)           (None, 13, 13, 256)       295168
```

```
batch_normalization_1 (Batc  (None, 13, 13, 256)      1024
hNormalization)

activation_1 (Activation)   (None, 13, 13, 256)       0

max_pooling2d_1 (MaxPooling  (None, 6, 6, 256)        0
2D)

dropout_1 (Dropout)         (None, 6, 6, 256)         0

conv2d_2 (Conv2D)           (None, 4, 4, 512)         1180160

batch_normalization_2 (Batc  (None, 4, 4, 512)        2048
hNormalization)

activation_2 (Activation)   (None, 4, 4, 512)         0

max_pooling2d_2 (MaxPooling  (None, 2, 2, 512)        0
2D)

dropout_2 (Dropout)         (None, 2, 2, 512)         0

flatten (Flatten)           (None, 2048)              0

dense (Dense)               (None, 512)               1049088

batch_normalization_3 (Batc  (None, 512)              2048
hNormalization)

activation_3 (Activation)   (None, 512)               0

dropout_3 (Dropout)         (None, 512)               0

dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 2,538,762
Trainable params: 2,535,946
Non-trainable params: 2,816
_________________________________________________________________
```

```python
# Definir os Callbacks

# Para salvar o melhor modelo com base na acurácia de validação
checkpoint =
ModelCheckpoint("models/modelo_S_sem_data_augmentation_rmsprop.keras",
monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

# Parar o treinamento se não houver melhoria na loss após x epochs
```

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

# Salvar para csv
csv_logger =
CSVLogger(f'logs/modelo_S_sem_data_augmentation_rmsprop.csv',
append=True)

# Reduzir a learning rate se não houver melhoria na loss após x epochs
(lembrar de deixar este valor sempre menor que a patience no
early_stopping!!)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=3, verbose=1)

# calcular passos por epoch
steps_per_epoch = sum([gen.samples // BATCH_SIZE for gen in
train_generators])

# Treinar o modelo - Nao tirar os callbacks
history = model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=NUM_EPOCHS,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    callbacks=[checkpoint, early_stopping, csv_logger, reduce_lr]
)


# Avaliar o modelo no test generator
results = model.evaluate(test_generator)
loss, accuracy, precision, recall, f1_score = results[:5]
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
print(f"Test Precision: {precision}")
print(f"Test Recall: {recall}")
print(f"Test F1 Score: {f1_score}")
```

```
Epoch 1/60
1247/1248 [============================>.] - ETA: 0s - loss: 1.6425 -
accuracy: 0.4206 - precision: 0.5597 - recall: 0.2549 - f1_score:
0.3503

c:\Users\USER\.conda\envs\py310\lib\site-packages\keras\engine\
training.py:2319: UserWarning: Metric Precision implements a
`reset_states()` method; rename it to `reset_state()` (without the
final "s"). The name `reset_states()` has been deprecated to improve
API consistency.
  m.reset_state()
c:\Users\USER\.conda\envs\py310\lib\site-packages\keras\engine\
```

```
training.py:2319: UserWarning: Metric Recall implements a
`reset_states()` method; rename it to `reset_state()` (without the
final "s"). The name `reset_states()` has been deprecated to improve
API consistency.
  m.reset_state()
c:\Users\USER\.conda\envs\py310\lib\site-packages\keras\engine\
training.py:2319: UserWarning: Metric F1Score implements a
`reset_states()` method; rename it to `reset_state()` (without the
final "s"). The name `reset_states()` has been deprecated to improve
API consistency.
  m.reset_state()


Epoch 1: val_accuracy improved from -inf to 0.43329, saving model to
models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 39s 25ms/step - loss:
1.6422 - accuracy: 0.4207 - precision: 0.5599 - recall: 0.2551 -
f1_score: 0.3505 - val_loss: 1.7063 - val_accuracy: 0.4333 -
val_precision: 0.6726 - val_recall: 0.2424 - val_f1_score: 0.3564 -
lr: 0.0010
Epoch 2/60
1246/1248 [==============================>.] - ETA: 0s - loss: 1.2353 -
accuracy: 0.5596 - precision: 0.7066 - recall: 0.4059 - f1_score:
0.5156
Epoch 2: val_accuracy improved from 0.43329 to 0.57632, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
1.2350 - accuracy: 0.5598 - precision: 0.7069 - recall: 0.4062 -
f1_score: 0.5159 - val_loss: 1.2190 - val_accuracy: 0.5763 -
val_precision: 0.7236 - val_recall: 0.4224 - val_f1_score: 0.5334 -
lr: 0.0010
Epoch 3/60
1245/1248 [==============================>.] - ETA: 0s - loss: 1.1030 -
accuracy: 0.6076 - precision: 0.7383 - recall: 0.4739 - f1_score:
0.5772
Epoch 3: val_accuracy did not improve from 0.57632
1248/1248 [==============================] - 16s 13ms/step - loss:
1.1025 - accuracy: 0.6077 - precision: 0.7386 - recall: 0.4741 -
f1_score: 0.5775 - val_loss: 1.7925 - val_accuracy: 0.4097 -
val_precision: 0.4584 - val_recall: 0.3185 - val_f1_score: 0.3759 -
lr: 0.0010
Epoch 4/60
1246/1248 [==============================>.] - ETA: 0s - loss: 1.0075 -
accuracy: 0.6447 - precision: 0.7619 - recall: 0.5281 - f1_score:
0.6238
Epoch 4: val_accuracy improved from 0.57632 to 0.66176, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
1.0075 - accuracy: 0.6447 - precision: 0.7620 - recall: 0.5281 -
f1_score: 0.6238 - val_loss: 0.9536 - val_accuracy: 0.6618 -
```

```
val_precision: 0.7938 - val_recall: 0.5398 - val_f1_score: 0.6426 -
lr: 0.0010
Epoch 5/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.9459 -
accuracy: 0.6649 - precision: 0.7739 - recall: 0.5589 - f1_score:
0.6491
Epoch 5: val_accuracy did not improve from 0.66176
1248/1248 [==============================] - 16s 13ms/step - loss:
0.9464 - accuracy: 0.6648 - precision: 0.7737 - recall: 0.5588 -
f1_score: 0.6489 - val_loss: 1.0578 - val_accuracy: 0.6421 -
val_precision: 0.7592 - val_recall: 0.5373 - val_f1_score: 0.6292 -
lr: 0.0010
Epoch 6/60
1248/1248 [==============================] - ETA: 0s - loss: 0.8926 -
accuracy: 0.6867 - precision: 0.7879 - recall: 0.5868 - f1_score:
0.6726
Epoch 6: val_accuracy did not improve from 0.66176
1248/1248 [==============================] - 16s 13ms/step - loss:
0.8926 - accuracy: 0.6867 - precision: 0.7879 - recall: 0.5868 -
f1_score: 0.6726 - val_loss: 1.2481 - val_accuracy: 0.5860 -
val_precision: 0.6478 - val_recall: 0.5168 - val_f1_score: 0.5750 -
lr: 0.0010
Epoch 7/60
1244/1248 [============================>.] - ETA: 0s - loss: 0.8535 -
accuracy: 0.7027 - precision: 0.7992 - recall: 0.6102 - f1_score:
0.6920
Epoch 7: val_accuracy improved from 0.66176 to 0.67077, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.8534 - accuracy: 0.7028 - precision: 0.7993 - recall: 0.6102 -
f1_score: 0.6921 - val_loss: 0.9295 - val_accuracy: 0.6708 -
val_precision: 0.7523 - val_recall: 0.5935 - val_f1_score: 0.6636 -
lr: 0.0010
Epoch 8/60
1248/1248 [==============================] - ETA: 0s - loss: 0.8146 -
accuracy: 0.7153 - precision: 0.8056 - recall: 0.6298 - f1_score:
0.7069
Epoch 8: val_accuracy improved from 0.67077 to 0.68109, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.8146 - accuracy: 0.7153 - precision: 0.8056 - recall: 0.6298 -
f1_score: 0.7069 - val_loss: 0.9442 - val_accuracy: 0.6811 -
val_precision: 0.7595 - val_recall: 0.6135 - val_f1_score: 0.6787 -
lr: 0.0010
Epoch 9/60
1245/1248 [============================>.] - ETA: 0s - loss: 0.7845 -
accuracy: 0.7261 - precision: 0.8096 - recall: 0.6433 - f1_score:
0.7169
Epoch 9: val_accuracy did not improve from 0.68109
```

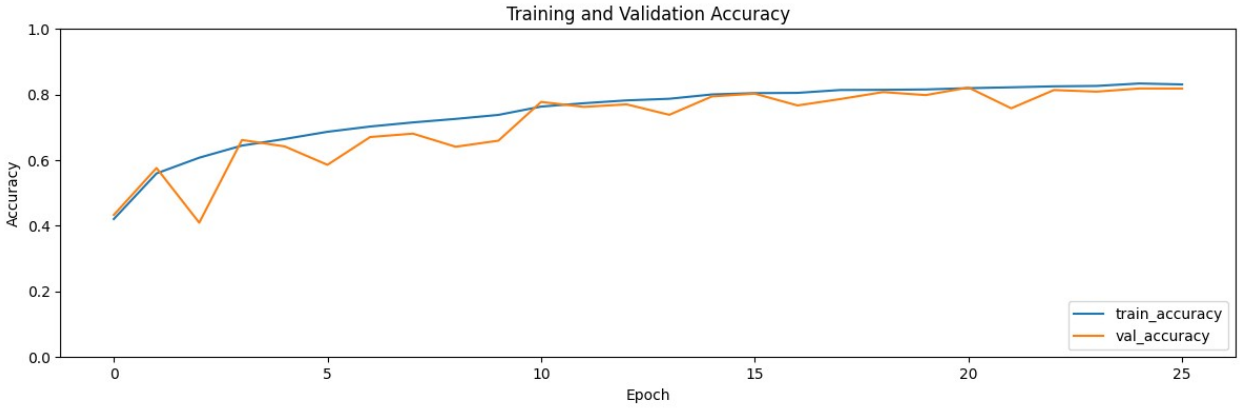
```
1248/1248 [==============================] - 16s 13ms/step - loss:
0.7848 - accuracy: 0.7260 - precision: 0.8095 - recall: 0.6432 -
f1_score: 0.7168 - val_loss: 1.0365 - val_accuracy: 0.6411 -
val_precision: 0.7427 - val_recall: 0.5482 - val_f1_score: 0.6308 -
lr: 0.0010
Epoch 10/60
1248/1248 [==============================] - ETA: 0s - loss: 0.7491 -
accuracy: 0.7382 - precision: 0.8188 - recall: 0.6634 - f1_score:
0.7329
Epoch 10: val_accuracy did not improve from 0.68109

Epoch 10: ReduceLROnPlateau reducing learning rate to
0.0005000000237487257.
1248/1248 [==============================] - 16s 13ms/step - loss:
0.7491 - accuracy: 0.7382 - precision: 0.8188 - recall: 0.6634 -
f1_score: 0.7329 - val_loss: 0.9825 - val_accuracy: 0.6597 -
val_precision: 0.7386 - val_recall: 0.5970 - val_f1_score: 0.6603 -
lr: 0.0010
Epoch 11/60
1248/1248 [==============================] - ETA: 0s - loss: 0.6765 -
accuracy: 0.7635 - precision: 0.8390 - recall: 0.6942 - f1_score:
0.7597
Epoch 11: val_accuracy improved from 0.68109 to 0.77774, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.6765 - accuracy: 0.7635 - precision: 0.8390 - recall: 0.6942 -
f1_score: 0.7597 - val_loss: 0.6389 - val_accuracy: 0.7777 -
val_precision: 0.8549 - val_recall: 0.7152 - val_f1_score: 0.7789 -
lr: 5.0000e-04
Epoch 12/60
1244/1248 [============================>.] - ETA: 0s - loss: 0.6457 -
accuracy: 0.7739 - precision: 0.8417 - recall: 0.7098 - f1_score:
0.7702
Epoch 12: val_accuracy did not improve from 0.77774
1248/1248 [==============================] - 17s 13ms/step - loss:
0.6462 - accuracy: 0.7738 - precision: 0.8415 - recall: 0.7096 -
f1_score: 0.7700 - val_loss: 0.6810 - val_accuracy: 0.7627 -
val_precision: 0.8348 - val_recall: 0.7048 - val_f1_score: 0.7643 -
lr: 5.0000e-04
Epoch 13/60
1244/1248 [============================>.] - ETA: 0s - loss: 0.6214 -
accuracy: 0.7824 - precision: 0.8483 - recall: 0.7243 - f1_score:
0.7814
Epoch 13: val_accuracy did not improve from 0.77774
1248/1248 [==============================] - 16s 13ms/step - loss:
0.6212 - accuracy: 0.7824 - precision: 0.8483 - recall: 0.7243 -
f1_score: 0.7814 - val_loss: 0.6669 - val_accuracy: 0.7703 -
val_precision: 0.8339 - val_recall: 0.7124 - val_f1_score: 0.7684 -
lr: 5.0000e-04
```

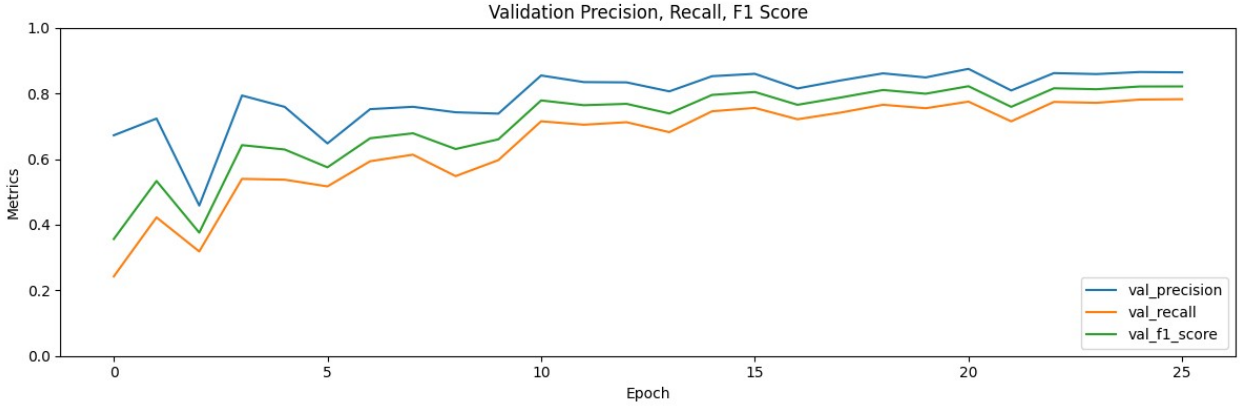
```
Epoch 14/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.6130 -
accuracy: 0.7874 - precision: 0.8488 - recall: 0.7270 - f1_score:
0.7832
Epoch 14: val_accuracy did not improve from 0.77774

Epoch 14: ReduceLROnPlateau reducing learning rate to
0.00025000000118743628.
1248/1248 [==============================] - 16s 13ms/step - loss:
0.6131 - accuracy: 0.7874 - precision: 0.8488 - recall: 0.7270 -
f1_score: 0.7832 - val_loss: 0.7669 - val_accuracy: 0.7385 -
val_precision: 0.8065 - val_recall: 0.6821 - val_f1_score: 0.7391 -
lr: 5.0000e-04
Epoch 15/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.5744 -
accuracy: 0.8006 - precision: 0.8587 - recall: 0.7433 - f1_score:
0.7969
Epoch 15: val_accuracy improved from 0.77774 to 0.79457, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5741 - accuracy: 0.8006 - precision: 0.8588 - recall: 0.7434 -
f1_score: 0.7969 - val_loss: 0.6000 - val_accuracy: 0.7946 -
val_precision: 0.8527 - val_recall: 0.7458 - val_f1_score: 0.7957 -
lr: 2.5000e-04
Epoch 16/60
1247/1248 [============================>.] - ETA: 0s - loss: 0.5594 -
accuracy: 0.8046 - precision: 0.8627 - recall: 0.7529 - f1_score:
0.8041
Epoch 16: val_accuracy improved from 0.79457 to 0.80288, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5593 - accuracy: 0.8046 - precision: 0.8627 - recall: 0.7530 -
f1_score: 0.8041 - val_loss: 0.5731 - val_accuracy: 0.8029 -
val_precision: 0.8600 - val_recall: 0.7560 - val_f1_score: 0.8046 -
lr: 2.5000e-04
Epoch 17/60
1243/1248 [============================>.] - ETA: 0s - loss: 0.5571 -
accuracy: 0.8053 - precision: 0.8610 - recall: 0.7541 - f1_score:
0.8040
Epoch 17: val_accuracy did not improve from 0.80288
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5572 - accuracy: 0.8053 - precision: 0.8612 - recall: 0.7541 -
f1_score: 0.8041 - val_loss: 0.6834 - val_accuracy: 0.7670 -
val_precision: 0.8154 - val_recall: 0.7215 - val_f1_score: 0.7655 -
lr: 2.5000e-04
Epoch 18/60
1247/1248 [============================>.] - ETA: 0s - loss: 0.5365 -
accuracy: 0.8140 - precision: 0.8653 - recall: 0.7621 - f1_score:
0.8104
```

```
Epoch 18: val_accuracy did not improve from 0.80288
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5363 - accuracy: 0.8141 - precision: 0.8653 - recall: 0.7622 -
f1_score: 0.8105 - val_loss: 0.6258 - val_accuracy: 0.7865 -
val_precision: 0.8396 - val_recall: 0.7416 - val_f1_score: 0.7876 -
lr: 2.5000e-04
Epoch 19/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.5280 -
accuracy: 0.8145 - precision: 0.8646 - recall: 0.7671 - f1_score:
0.8129
Epoch 19: val_accuracy improved from 0.80288 to 0.80769, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5276 - accuracy: 0.8146 - precision: 0.8647 - recall: 0.7673 -
f1_score: 0.8131 - val_loss: 0.5592 - val_accuracy: 0.8077 -
val_precision: 0.8614 - val_recall: 0.7656 - val_f1_score: 0.8107 -
lr: 2.5000e-04
Epoch 20/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.5260 -
accuracy: 0.8157 - precision: 0.8658 - recall: 0.7690 - f1_score:
0.8145
Epoch 20: val_accuracy did not improve from 0.80769
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5257 - accuracy: 0.8158 - precision: 0.8658 - recall: 0.7691 -
f1_score: 0.8146 - val_loss: 0.5878 - val_accuracy: 0.7986 -
val_precision: 0.8490 - val_recall: 0.7551 - val_f1_score: 0.7993 -
lr: 2.5000e-04
Epoch 21/60
1244/1248 [============================>.] - ETA: 0s - loss: 0.5155 -
accuracy: 0.8197 - precision: 0.8703 - recall: 0.7735 - f1_score:
0.8190
Epoch 21: val_accuracy improved from 0.80769 to 0.82161, saving model
to models\modelo_S_sem_data_augmentation.keras
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5155 - accuracy: 0.8197 - precision: 0.8703 - recall: 0.7735 -
f1_score: 0.8190 - val_loss: 0.5239 - val_accuracy: 0.8216 -
val_precision: 0.8752 - val_recall: 0.7752 - val_f1_score: 0.8222 -
lr: 2.5000e-04
Epoch 22/60
1248/1248 [==============================] - ETA: 0s - loss: 0.5068 -
accuracy: 0.8223 - precision: 0.8715 - recall: 0.7778 - f1_score:
0.8220
Epoch 22: val_accuracy did not improve from 0.82161
1248/1248 [==============================] - 16s 13ms/step - loss:
0.5068 - accuracy: 0.8223 - precision: 0.8715 - recall: 0.7778 -
f1_score: 0.8220 - val_loss: 0.7397 - val_accuracy: 0.7583 -
val_precision: 0.8092 - val_recall: 0.7150 - val_f1_score: 0.7592 -
lr: 2.5000e-04
Epoch 23/60
```

```
1246/1248 [============================>.] - ETA: 0s - loss: 0.4968 -
accuracy: 0.8252 - precision: 0.8722 - recall: 0.7822 - f1_score:
0.8247
Epoch 23: val_accuracy did not improve from 0.82161
1248/1248 [==============================] - 16s 13ms/step - loss:
0.4967 - accuracy: 0.8253 - precision: 0.8722 - recall: 0.7822 -
f1_score: 0.8248 - val_loss: 0.5479 - val_accuracy: 0.8139 -
val_precision: 0.8622 - val_recall: 0.7743 - val_f1_score: 0.8159 -
lr: 2.5000e-04
Epoch 24/60
1247/1248 [============================>.] - ETA: 0s - loss: 0.4947 -
accuracy: 0.8264 - precision: 0.8721 - recall: 0.7847 - f1_score:
0.8261
Epoch 24: val_accuracy did not improve from 0.82161

Epoch 24: ReduceLROnPlateau reducing learning rate to
0.0001250000059371814.
1248/1248 [==============================] - 18s 14ms/step - loss:
0.4946 - accuracy: 0.8265 - precision: 0.8722 - recall: 0.7848 -
f1_score: 0.8262 - val_loss: 0.5542 - val_accuracy: 0.8091 -
val_precision: 0.8592 - val_recall: 0.7715 - val_f1_score: 0.8130 -
lr: 2.5000e-04
Epoch 25/60
1246/1248 [============================>.] - ETA: 0s - loss: 0.4805 -
accuracy: 0.8338 - precision: 0.8778 - recall: 0.7917 - f1_score:
0.8325
Epoch 25: val_accuracy did not improve from 0.82161
1248/1248 [==============================] - 17s 14ms/step - loss:
0.4803 - accuracy: 0.8339 - precision: 0.8779 - recall: 0.7918 -
f1_score: 0.8326 - val_loss: 0.5360 - val_accuracy: 0.8187 -
val_precision: 0.8654 - val_recall: 0.7815 - val_f1_score: 0.8213 -
lr: 1.2500e-04
Epoch 26/60
1248/1248 [==============================] - ETA: 0s - loss: 0.4765 -
accuracy: 0.8311 - precision: 0.8764 - recall: 0.7906 - f1_score:
0.8313
Epoch 26: val_accuracy did not improve from 0.82161
1248/1248 [==============================] - 17s 13ms/step - loss:
0.4765 - accuracy: 0.8311 - precision: 0.8764 - recall: 0.7906 -
f1_score: 0.8313 - val_loss: 0.5241 - val_accuracy: 0.8185 -
val_precision: 0.8645 - val_recall: 0.7827 - val_f1_score: 0.8215 -
lr: 1.2500e-04
313/313 [==============================] - 6s 20ms/step - loss: 0.5055
- accuracy: 0.8244 - precision: 0.8788 - recall: 0.7808 - f1_score:
0.8269
Test Loss: 0.5054613947868347
Test Accuracy: 0.824400007724762
Test Precision: 0.8787844777107239
```

```
Test Recall: 0.7807999849319458
Test F1 Score: 0.8268995881080627

# Plots do treino
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(history.history['val_precision'], label='val_precision')
plt.plot(history.history['val_recall'], label='val_recall')
plt.plot(history.history['val_f1_score'], label='val_f1_score')
plt.xlabel('Epoch')
plt.ylabel('Metrics')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.title('Validation Precision, Recall, F1 Score')

plt.tight_layout()
plt.savefig(f'./plots/modelo_S_sem_data_augmentation_rmsprop_rmsprop.png')
plt.show()
```


Training and Validation Accuracy
Accuracy
Epoch
train_accuracy
val_accuracy
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25


Validation Precision, Recall, F1 Score
Metrics
Epoch
val_precision
val_recall
val_f1_score
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25