Modelo T sem data augmentation e optimizer RMSProp

```
from tensorflow.keras.metrics import Metric
from tensorflow.keras import backend as K
import json
import os
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout,
Dense, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau, CSVLogger
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.regularizers import l2
from tensorflow.keras.mixed_precision import set_global_policy

# MIX precision training -- facilita no treino!
set_global_policy('mixed_float16')

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# CONSTANTES
BATCH_SIZE = 64
IMG_SIZE = 150
NUM_CLASSES = 10  # nº classes para identificar
NUM_EPOCHS = 60
LEARNING_RATE = 0.0001
DENSE_LAYERS = [1024, 512, 256, 128]
```

```
INFO:tensorflow:Mixed precision compatibility check (mixed_float16):
OK
Your GPU will likely run quickly with dtype policy mixed_float16 as it
has compute capability of at least 7.0. Your GPU: NVIDIA GeForce RTX
4070, compute capability 8.9
```

```
# 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'

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

```python
# 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 juntar os trainning generators and repeat


def combined_generator(generators):
    while True:
        for generator in generators:
            for batch in generator:
                yield batch


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')

# Load the pre-trained ResNet50 model without the top layer and adjust
input shape
base_model = ResNet50(weights='imagenet', include_top=False,
                      input_shape=(IMG_SIZE, IMG_SIZE, 3))

# Descongelar camadas (nao meter valores demasiado altos)
for layer in base_model.layers[-50:]:
    layer.trainable = True
```

```
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.
```

```python
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()

# Definir as layers do modelo com parametros ajustados para reduzir o 
overfitting
model = Sequential([
    base_model,
    BatchNormalization(),
    GlobalAveragePooling2D(),
    # Increase model complexity
    Dense(DENSE_LAYERS[0], activation='relu', 
kernel_regularizer=l2(0.03)),
    Dropout(0.5),  # High dropout rate for regularization
    BatchNormalization(),
    Dense(DENSE_LAYERS[1], activation='relu', 
kernel_regularizer=l2(0.03)),
    Dropout(0.5),
    BatchNormalization(),
    Dense(DENSE_LAYERS[2], activation='relu', 
kernel_regularizer=l2(0.03)),
    Dropout(0.5),
    Dense(DENSE_LAYERS[3], activation='relu', 
kernel_regularizer=l2(0.03)),
    Dropout(0.5),
    BatchNormalization(),
    Dense(NUM_CLASSES, activation='softmax', dtype='float32')
])

# 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 #   
```

```
=================================================================
 resnet50 (Functional)       (None, 5, 5, 2048)        23587712

 batch_normalization (BatchN  (None, 5, 5, 2048)       8192
 ormalization)

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 1024)              2098176

 dropout (Dropout)           (None, 1024)              0

 batch_normalization_1 (Batc  (None, 1024)             4096
 hNormalization)

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 batch_normalization_2 (Batc  (None, 512)              2048
 hNormalization)

 dense_2 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_3 (Dropout)         (None, 128)               0

 batch_normalization_3 (Batc  (None, 128)              512
 hNormalization)

 dense_4 (Dense)             (None, 10)                1290

=================================================================
Total params: 26,391,050
Trainable params: 26,330,506
Non-trainable params: 60,544
_________________________________________________________________
```

```
# CALLBACKS
os.makedirs('logs', exist_ok=True)
checkpoint = 
ModelCheckpoint(f'models/modelo_T_sem_data_augmentation_rmsprop.keras'
,
                             monitor='val_accuracy', verbose=1, 
save_best_only=True, mode='max')
early_stopping = EarlyStopping(
    monitor='val_loss', patience=10, restore_best_weights=True)  # 
Increased patience
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss', factor=0.2, patience=4, min_lr=1e-7, 
verbose=1)  # More aggressive schedule
csv_logger = CSVLogger(
    f'logs/modelo_T_sem_data_augmentation_rmsprop.csv', separator=',', 
append=False)

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

# calcular passos por epoch
# 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_steps,
    callbacks=[checkpoint, early_stopping, reduce_lr, csv_logger]
)

# Avaliar o modelo no test generator
# 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
624/624 [==============================] - ETA: 0s - loss: 53.1007 - 
accuracy: 0.4440 - precision: 0.6692 - recall: 0.2674 - f1_score: 
0.3821
```

```
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 0.10086, saving model to
models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 74s 95ms/step - loss:
53.1007 - accuracy: 0.4440 - precision: 0.6692 - recall: 0.2674 -
f1_score: 0.3821 - val_loss: 32.8211 - val_accuracy: 0.1009 -
val_precision: 0.1093 - val_recall: 0.0965 - val_f1_score: 0.1025 -
lr: 1.0000e-04
Epoch 2/60
624/624 [==============================] - ETA: 0s - loss: 17.4746 -
accuracy: 0.9086 - precision: 0.9539 - recall: 0.8405 - f1_score:
0.8936
Epoch 2: val_accuracy improved from 0.10086 to 0.78476, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 57s 92ms/step - loss:
17.4746 - accuracy: 0.9086 - precision: 0.9539 - recall: 0.8405 -
f1_score: 0.8936 - val_loss: 8.7829 - val_accuracy: 0.7848 -
val_precision: 0.8591 - val_recall: 0.7413 - val_f1_score: 0.7958 -
lr: 1.0000e-04
Epoch 3/60
624/624 [==============================] - ETA: 0s - loss: 4.4318 -
accuracy: 0.9613 - precision: 0.9743 - recall: 0.9431 - f1_score:
0.9584
Epoch 3: val_accuracy improved from 0.78476 to 0.82252, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 57s 92ms/step - loss:
4.4318 - accuracy: 0.9613 - precision: 0.9743 - recall: 0.9431 -
f1_score: 0.9584 - val_loss: 2.5695 - val_accuracy: 0.8225 -
val_precision: 0.8608 - val_recall: 0.7964 - val_f1_score: 0.8273 -
lr: 1.0000e-04
Epoch 4/60
624/624 [==============================] - ETA: 0s - loss: 1.2353 -
accuracy: 0.9681 - precision: 0.9757 - recall: 0.9582 - f1_score:
0.9669
Epoch 4: val_accuracy improved from 0.82252 to 0.86899, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 57s 92ms/step - loss:
1.2353 - accuracy: 0.9681 - precision: 0.9757 - recall: 0.9582 -
f1_score: 0.9669 - val_loss: 1.1237 - val_accuracy: 0.8690 -
val_precision: 0.8831 - val_recall: 0.8592 - val_f1_score: 0.8710 -
lr: 1.0000e-04
```

```
Epoch 5/60
624/624 [==============================] - ETA: 0s - loss: 0.5197 -
accuracy: 0.9742 - precision: 0.9786 - recall: 0.9676 - f1_score:
0.9731
Epoch 5: val_accuracy did not improve from 0.86899
624/624 [==============================] - 56s 90ms/step - loss:
0.5197 - accuracy: 0.9742 - precision: 0.9786 - recall: 0.9676 -
f1_score: 0.9731 - val_loss: 0.9921 - val_accuracy: 0.8468 -
val_precision: 0.8577 - val_recall: 0.8385 - val_f1_score: 0.8480 -
lr: 1.0000e-04
Epoch 6/60
624/624 [==============================] - ETA: 0s - loss: 0.3272 -
accuracy: 0.9778 - precision: 0.9817 - recall: 0.9728 - f1_score:
0.9772
Epoch 6: val_accuracy did not improve from 0.86899
624/624 [==============================] - 58s 93ms/step - loss:
0.3272 - accuracy: 0.9778 - precision: 0.9817 - recall: 0.9728 -
f1_score: 0.9772 - val_loss: 0.7588 - val_accuracy: 0.8684 -
val_precision: 0.8810 - val_recall: 0.8595 - val_f1_score: 0.8701 -
lr: 1.0000e-04
Epoch 7/60
624/624 [==============================] - ETA: 0s - loss: 0.2628 -
accuracy: 0.9800 - precision: 0.9835 - recall: 0.9765 - f1_score:
0.9800
Epoch 7: val_accuracy did not improve from 0.86899
624/624 [==============================] - 57s 92ms/step - loss:
0.2628 - accuracy: 0.9800 - precision: 0.9835 - recall: 0.9765 -
f1_score: 0.9800 - val_loss: 0.8225 - val_accuracy: 0.8573 -
val_precision: 0.8697 - val_recall: 0.8521 - val_f1_score: 0.8608 -
lr: 1.0000e-04
Epoch 8/60
624/624 [==============================] - ETA: 0s - loss: 0.2503 -
accuracy: 0.9803 - precision: 0.9831 - recall: 0.9763 - f1_score:
0.9797
Epoch 8: val_accuracy did not improve from 0.86899
624/624 [==============================] - 57s 92ms/step - loss:
0.2503 - accuracy: 0.9803 - precision: 0.9831 - recall: 0.9763 -
f1_score: 0.9797 - val_loss: 0.8837 - val_accuracy: 0.8489 -
val_precision: 0.8638 - val_recall: 0.8376 - val_f1_score: 0.8505 -
lr: 1.0000e-04
Epoch 9/60
624/624 [==============================] - ETA: 0s - loss: 0.2296 -
accuracy: 0.9823 - precision: 0.9850 - recall: 0.9791 - f1_score:
0.9820
Epoch 9: val_accuracy did not improve from 0.86899
624/624 [==============================] - 57s 92ms/step - loss:
0.2296 - accuracy: 0.9823 - precision: 0.9850 - recall: 0.9791 -
f1_score: 0.9820 - val_loss: 0.8584 - val_accuracy: 0.8569 -
val_precision: 0.8652 - val_recall: 0.8507 - val_f1_score: 0.8579 -
```

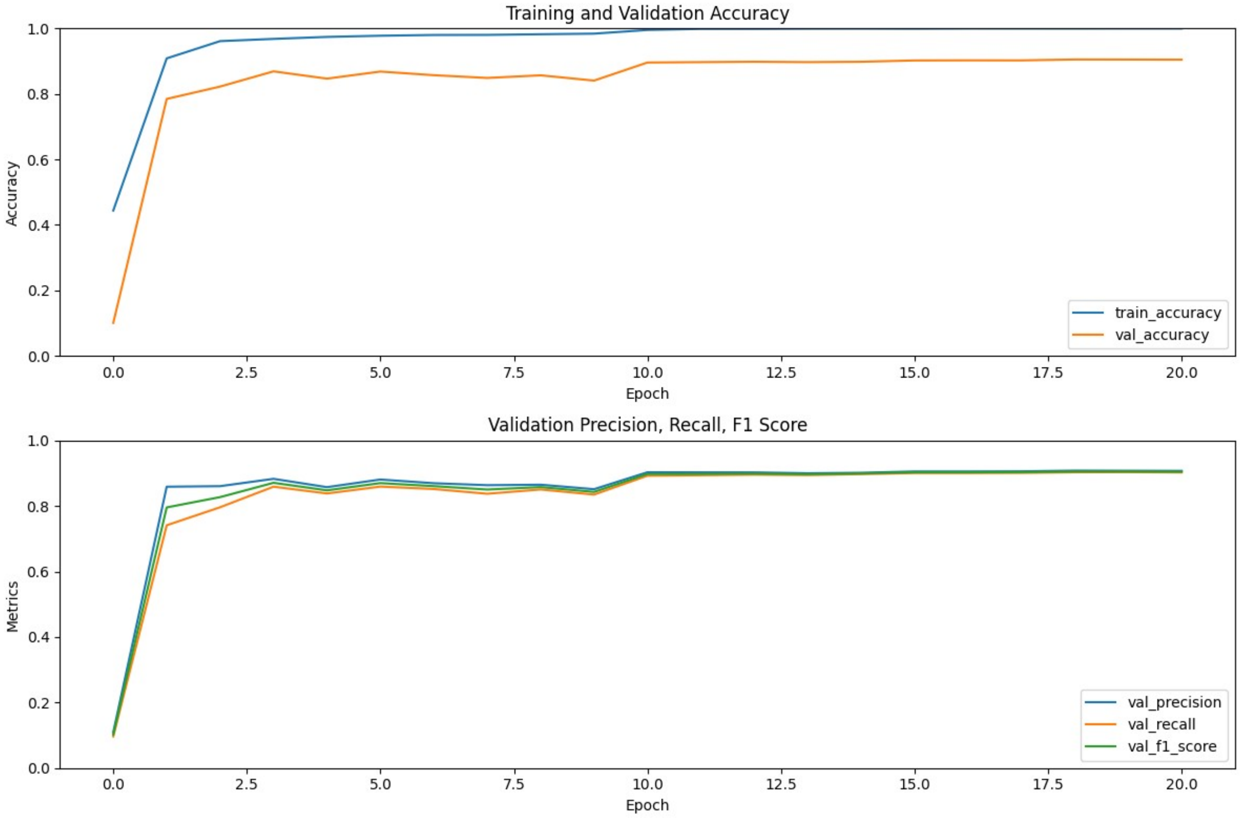
```
lr: 1.0000e-04
Epoch 10/60
624/624 [==============================] - ETA: 0s - loss: 0.2213 -
accuracy: 0.9841 - precision: 0.9865 - recall: 0.9815 - f1_score:
0.9840
Epoch 10: val_accuracy did not improve from 0.86899

Epoch 10: ReduceLROnPlateau reducing learning rate to
1.9999999494757503e-05.
624/624 [==============================] - 57s 91ms/step - loss:
0.2213 - accuracy: 0.9841 - precision: 0.9865 - recall: 0.9815 -
f1_score: 0.9840 - val_loss: 0.9745 - val_accuracy: 0.8407 -
val_precision: 0.8515 - val_recall: 0.8351 - val_f1_score: 0.8432 -
lr: 1.0000e-04
Epoch 11/60
624/624 [==============================] - ETA: 0s - loss: 0.1416 -
accuracy: 0.9953 - precision: 0.9961 - recall: 0.9942 - f1_score:
0.9952
Epoch 11: val_accuracy improved from 0.86899 to 0.89593, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 58s 92ms/step - loss:
0.1416 - accuracy: 0.9953 - precision: 0.9961 - recall: 0.9942 -
f1_score: 0.9952 - val_loss: 0.6085 - val_accuracy: 0.8959 -
val_precision: 0.9032 - val_recall: 0.8927 - val_f1_score: 0.8979 -
lr: 2.0000e-05
Epoch 12/60
624/624 [==============================] - ETA: 0s - loss: 0.0861 -
accuracy: 0.9986 - precision: 0.9987 - recall: 0.9980 - f1_score:
0.9984
Epoch 12: val_accuracy improved from 0.89593 to 0.89714, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 57s 92ms/step - loss:
0.0861 - accuracy: 0.9986 - precision: 0.9987 - recall: 0.9980 -
f1_score: 0.9984 - val_loss: 0.6235 - val_accuracy: 0.8971 -
val_precision: 0.9030 - val_recall: 0.8941 - val_f1_score: 0.8985 -
lr: 2.0000e-05
Epoch 13/60
624/624 [==============================] - ETA: 0s - loss: 0.0698 -
accuracy: 0.9986 - precision: 0.9988 - recall: 0.9982 - f1_score:
0.9985
Epoch 13: val_accuracy improved from 0.89714 to 0.89834, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 58s 92ms/step - loss:
0.0698 - accuracy: 0.9986 - precision: 0.9988 - recall: 0.9982 -
f1_score: 0.9985 - val_loss: 0.6265 - val_accuracy: 0.8983 -
val_precision: 0.9028 - val_recall: 0.8958 - val_f1_score: 0.8993 -
lr: 2.0000e-05
Epoch 14/60
624/624 [==============================] - ETA: 0s - loss: 0.0515 -
```

```
accuracy: 0.9993 - precision: 0.9994 - recall: 0.9991 - f1_score:
0.9992
Epoch 14: val_accuracy did not improve from 0.89834
624/624 [==============================] - 56s 90ms/step - loss:
0.0515 - accuracy: 0.9993 - precision: 0.9994 - recall: 0.9991 -
f1_score: 0.9992 - val_loss: 0.6487 - val_accuracy: 0.8971 -
val_precision: 0.9002 - val_recall: 0.8946 - val_f1_score: 0.8974 -
lr: 2.0000e-05
Epoch 15/60
624/624 [==============================] - ETA: 0s - loss: 0.0438 -
accuracy: 0.9994 - precision: 0.9996 - recall: 0.9992 - f1_score:
0.9994
Epoch 15: val_accuracy did not improve from 0.89834

Epoch 15: ReduceLROnPlateau reducing learning rate to
3.999999898951501e-06.
624/624 [==============================] - 57s 91ms/step - loss:
0.0438 - accuracy: 0.9994 - precision: 0.9996 - recall: 0.9992 -
f1_score: 0.9994 - val_loss: 0.6856 - val_accuracy: 0.8982 -
val_precision: 0.9016 - val_recall: 0.8975 - val_f1_score: 0.8996 -
lr: 2.0000e-05
Epoch 16/60
624/624 [==============================] - ETA: 0s - loss: 0.0421 -
accuracy: 0.9992 - precision: 0.9993 - recall: 0.9989 - f1_score:
0.9991
Epoch 16: val_accuracy improved from 0.89834 to 0.90224, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 58s 92ms/step - loss:
0.0421 - accuracy: 0.9992 - precision: 0.9993 - recall: 0.9989 -
f1_score: 0.9991 - val_loss: 0.6374 - val_accuracy: 0.9022 -
val_precision: 0.9056 - val_recall: 0.9008 - val_f1_score: 0.9032 -
lr: 4.0000e-06
Epoch 17/60
624/624 [==============================] - ETA: 0s - loss: 0.0369 -
accuracy: 0.9998 - precision: 0.9998 - recall: 0.9997 - f1_score:
0.9997
Epoch 17: val_accuracy improved from 0.90224 to 0.90264, saving model
to models\modelo_T_sem_data_augmentation.keras
624/624 [==============================] - 58s 92ms/step - loss:
0.0369 - accuracy: 0.9998 - precision: 0.9998 - recall: 0.9997 -
f1_score: 0.9997 - val_loss: 0.6391 - val_accuracy: 0.9026 -
val_precision: 0.9056 - val_recall: 0.9008 - val_f1_score: 0.9032 -
lr: 4.0000e-06
Epoch 18/60
624/624 [==============================] - ETA: 0s - loss: 0.0348 -
accuracy: 0.9998 - precision: 0.9998 - recall: 0.9996 - f1_score:
0.9997
Epoch 18: val_accuracy did not improve from 0.90264
624/624 [==============================] - 56s 90ms/step - loss:
```

```
0.0348 - accuracy: 0.9998 - precision: 0.9998 - recall: 0.9996 -
f1_score: 0.9997 - val_loss: 0.6350 - val_accuracy: 0.9024 -
val_precision: 0.9061 - val_recall: 0.9013 - val_f1_score: 0.9037 -
lr: 4.0000e-06
Epoch 19/60
624/624 [==============================] - ETA: 0s - loss: 0.0332 -
accuracy: 0.9997 - precision: 0.9998 - recall: 0.9996 - f1_score:
0.9997
Epoch 19: val_accuracy improved from 0.90264 to 0.90525, saving model
to models\modelo_T_sem_data_augmentation.keras

Epoch 19: ReduceLROnPlateau reducing learning rate to
7.999999979801942e-07.
624/624 [==============================] - 57s 92ms/step - loss:
0.0332 - accuracy: 0.9997 - precision: 0.9998 - recall: 0.9996 -
f1_score: 0.9997 - val_loss: 0.6223 - val_accuracy: 0.9052 -
val_precision: 0.9081 - val_recall: 0.9031 - val_f1_score: 0.9056 -
lr: 4.0000e-06
Epoch 20/60
624/624 [==============================] - ETA: 0s - loss: 0.0321 -
accuracy: 0.9999 - precision: 0.9999 - recall: 0.9998 - f1_score:
0.9999
Epoch 20: val_accuracy did not improve from 0.90525
624/624 [==============================] - 56s 90ms/step - loss:
0.0321 - accuracy: 0.9999 - precision: 0.9999 - recall: 0.9998 -
f1_score: 0.9999 - val_loss: 0.6295 - val_accuracy: 0.9050 -
val_precision: 0.9078 - val_recall: 0.9035 - val_f1_score: 0.9057 -
lr: 8.0000e-07
Epoch 21/60
624/624 [==============================] - ETA: 0s - loss: 0.0314 -
accuracy: 0.9999 - precision: 0.9999 - recall: 0.9998 - f1_score:
0.9999
Epoch 21: val_accuracy did not improve from 0.90525
624/624 [==============================] - 57s 92ms/step - loss:
0.0314 - accuracy: 0.9999 - precision: 0.9999 - recall: 0.9998 -
f1_score: 0.9999 - val_loss: 0.6305 - val_accuracy: 0.9046 -
val_precision: 0.9074 - val_recall: 0.9026 - val_f1_score: 0.9050 -
lr: 8.0000e-07
157/157 [==============================] - 8s 51ms/step - loss: 0.6593
- accuracy: 0.8892 - precision: 0.8959 - recall: 0.8857 - f1_score:
0.8908
Test Loss: 0.6593149900436401
Test Accuracy: 0.88919997215271
Test Precision: 0.8959134221076965
Test Recall: 0.885699987411499
Test F1 Score: 0.8907773494720459
```

```
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
plt.plot(history.history['accuracy'], label='train_accuracy')
```

```python
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.savefig(f'./plots/modelo_T_sem_data_augmentation_rmsprop.png')
plt.tight_layout()
# plt.show()
```



Training and Validation Accuracy

Validation Precision, Recall, F1 Score