Final Project Image Classification

February 4, 2024

1. Import the Required Libraries

```
[1]: import tensorflow as tf
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.preprocessing import image
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
       ↪Dropout
     from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
       ↪ReduceLROnPlateau
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.utils import plot_model
     from sklearn.metrics import classification_report, confusion_matrix,␣
       ↪ConfusionMatrixDisplay
     from google.colab import files
     import matplotlib.pyplot as plt
     import matplotlib.image as mpimg
     import numpy as np
     import os
     import zipfile
     import time
```

2. Load Dataset

```
[2]: !wget --no-check-certificate \
       https://github.com/dicodingacademy/assets/releases/download/release/
       ↪rockpaperscissors.zip \
       -O /content/rockpaperscissors.zip

     local_zip = '/content/rockpaperscissors.zip'
     zip_ref = zipfile.ZipFile(local_zip, 'r')
     zip_ref.extractall('/content')
     zip_ref.close()
```

```
--2024-02-05 11:04:26--  https://github.com/dicodingacademy/assets/releases/down
load/release/rockpaperscissors.zip
```

```
Resolving github.com (github.com)… 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443… connected.
HTTP request sent, awaiting response… 302 Found
Location: https://objects.githubusercontent.com/github-production-release-
asset-2e65be/391417272/7eb836f2-695b-4a46-9c78-b65867166957?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAVCODYLSA53PQK4ZA%2F20240205%2Fus-
east-1%2Fs3%2Faws4_request&X-Amz-Date=20240205T110426Z&X-Amz-Expires=300&X-Amz-S
ignature=846f21039a0b85251966040d86a655c0f8b0eea5f74fb8faaa4e20cfe3058921&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=391417272&response-content-
disposition=attachment%3B%20filename%3Drockpaperscissors.zip&response-content-
type=application%2Foctet-stream [following]
--2024-02-05 11:04:26--  https://objects.githubusercontent.com/github-
production-release-
asset-2e65be/391417272/7eb836f2-695b-4a46-9c78-b65867166957?X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAVCODYLSA53PQK4ZA%2F20240205%2Fus-
east-1%2Fs3%2Faws4_request&X-Amz-Date=20240205T110426Z&X-Amz-Expires=300&X-Amz-S
ignature=846f21039a0b85251966040d86a655c0f8b0eea5f74fb8faaa4e20cfe3058921&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=391417272&response-content-
disposition=attachment%3B%20filename%3Drockpaperscissors.zip&response-content-
type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to objects.githubusercontent.com
(objects.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 322873683 (308M) [application/octet-stream]
Saving to: ‘/content/rockpaperscissors.zip’

/content/rockpapers 100%[===================>] 307.92M   331MB/s    in 0.9s

2024-02-05 11:04:28 (331 MB/s) - ‘/content/rockpaperscissors.zip’ saved
[322873683/322873683]
```

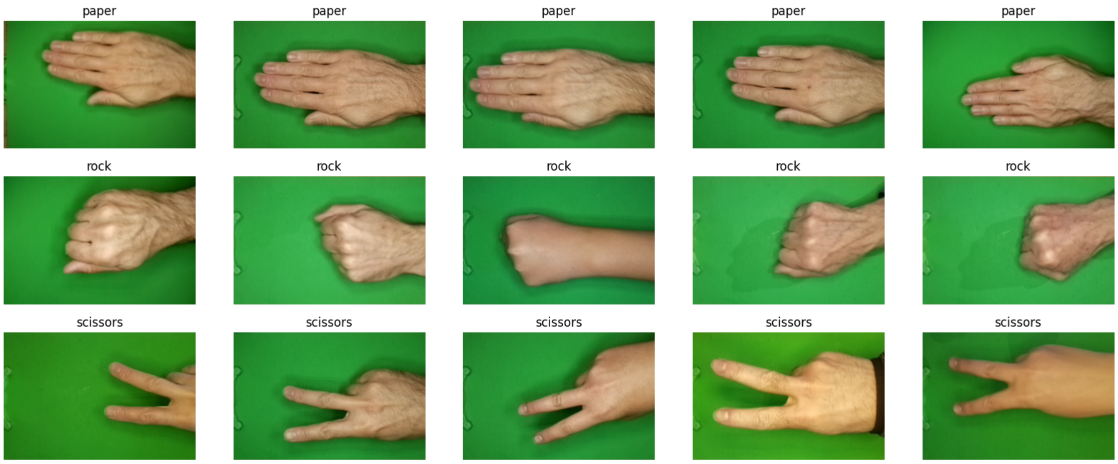
[3]:

```python
base_dir = '/content/rockpaperscissors/rps-cv-images'

paper_dir = os.path.join(base_dir, 'paper')
rock_dir = os.path.join(base_dir, 'rock')
scissors_dir = os.path.join(base_dir, 'scissors')

paper_imgs = os.listdir(paper_dir)
rock_imgs = os.listdir(rock_dir)
scissors_imgs = os.listdir(scissors_dir)
```

3. Check Image Dataset

[4]:

```
list_dir = [paper_dir, rock_dir, scissors_dir]
list_folders = [paper_imgs, rock_imgs, scissors_imgs]

plt.figure(figsize=(20, 8))
for x in range(3):
  for i, img_path in enumerate(list_folders[x][:5]):
    sp = plt.subplot(3, 5, x*5 + i + 1)
    img = mpimg.imread(os.path.join(list_dir[x], img_path))
    plt.imshow(img)
    plt.axis("off")
    plt.title(list_dir[x].split('/')[-1])

plt.show()
```



4. Augmented Image Dataset and Split Image Dataset

[5]:

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    shear_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.4)

validation_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.4)
```

[6]:
```
train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='training')

validation_generator = validation_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation')
```

```
Found 1314 images belonging to 3 classes.
Found 874 images belonging to 3 classes.
```

5. Building Model

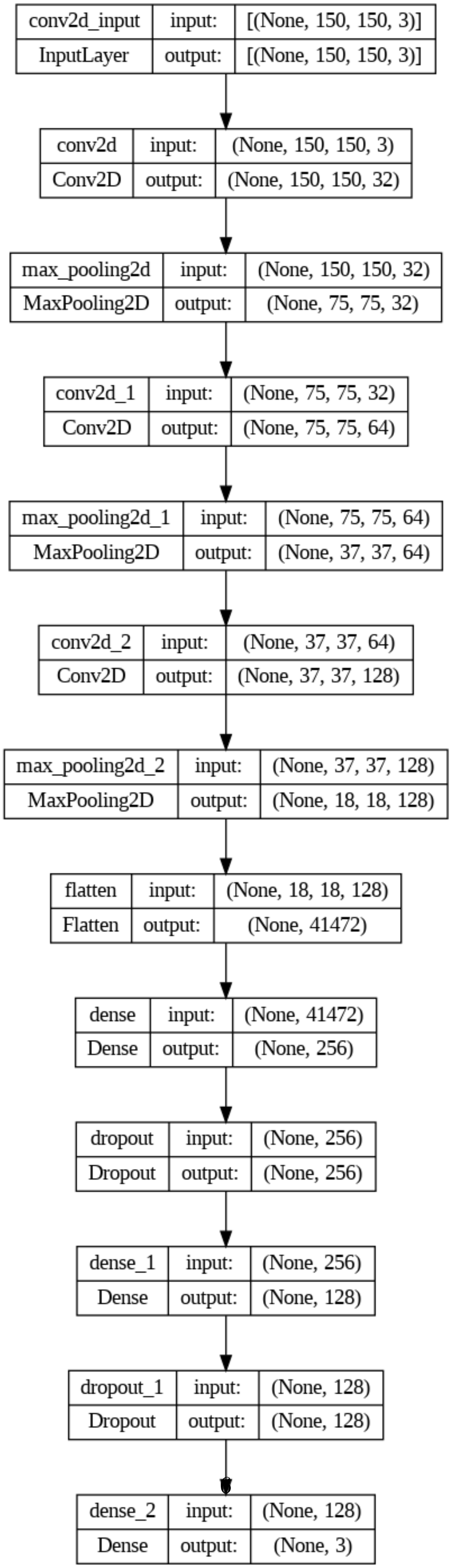
[7]:
```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', padding='same', input_shape=(150, 150,␣
 ↪3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu', padding='same'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu', padding='same'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 max_pooling2d (MaxPooling2  (None, 75, 75, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496
 max_pooling2d_1 (MaxPoolin  (None, 37, 37, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 18, 18, 128)       0
 g2D)

 flatten (Flatten)           (None, 41472)             0

 dense (Dense)               (None, 256)               10617088

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 3)                 387

=================================================================
Total params: 10743619 (40.98 MB)
Trainable params: 10743619 (40.98 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[8]:

```
plot_model(
    model,
    show_shapes=True,
    show_layer_names=True,
)
```

[8]:


conv2d_input | input: | [(None, 150, 150, 3)]
InputLayer | output: | [(None, 150, 150, 3)]
conv2d | input: | (None, 150, 150, 3)
Conv2D | output: | (None, 150, 150, 32)
max_pooling2d | input: | (None, 150, 150, 32)
MaxPooling2D | output: | (None, 75, 75, 32)
conv2d_1 | input: | (None, 75, 75, 32)
Conv2D | output: | (None, 75, 75, 64)
max_pooling2d_1 | input: | (None, 75, 75, 64)
MaxPooling2D | output: | (None, 37, 37, 64)
conv2d_2 | input: | (None, 37, 37, 64)
Conv2D | output: | (None, 37, 37, 128)
max_pooling2d_2 | input: | (None, 37, 37, 128)
MaxPooling2D | output: | (None, 18, 18, 128)
flatten | input: | (None, 18, 18, 128)
Flatten | output: | (None, 41472)
dense | input: | (None, 41472)
Dense | output: | (None, 256)
dropout | input: | (None, 256)
Dropout | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 128)
dropout_1 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 3)

6. Compile Model

[9]:

```
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

7. Fit Model

[10]:

```
def train_model(model, train_generator, validation_generator, steps_per_epoch,␣
 ↪validation_steps, epochs=40):
  callbacks = [
      ModelCheckpoint('best_model.h5', monitor='val_accuracy',␣
 ↪save_best_only=True, mode='max'),
      EarlyStopping(monitor='val_loss', patience=3, mode='min'),
      ReduceLROnPlateau(monitor='val_accuracy', patience=3, verbose=1, factor=0.
 ↪5, min_lr=0.000003)
  ]

  start_time = time.time()

  history = model.fit(
      train_generator,
      steps_per_epoch=steps_per_epoch,
      epochs=epochs,
      validation_data=validation_generator,
      validation_steps=validation_steps,
      verbose=2,
      callbacks=callbacks)

  end_time = time.time()

  training_time = (end_time - start_time)/60
  print(f'Training Time: {training_time} Minute')

  return history

def evaluate_model(model, train_generator, validation_generator):
  train_loss, train_acc = model.evaluate(train_generator)
  val_loss, val_acc = model.evaluate(validation_generator)
  if train_acc >= 0.96 and val_acc >= 0.96 and train_loss <= 0.3 and val_loss␣
 ↪<= 0.3:
      print('The best model with the specified criteria')
```

```
        return True
    else:
        print('Not the best model with the specified criteria')
        return False

while True:
    history = train_model(
        model,
        train_generator,
        validation_generator,
        steps_per_epoch=train_generator.samples//train_generator.batch_size,
        validation_steps=validation_generator.samples//validation_generator.
  ↪batch_size
        )
    best_model = tf.keras.models.load_model('best_model.h5')
    if evaluate_model(best_model, train_generator, validation_generator):
        best_model.save('final_model.h5')
        break
```

```
Epoch 1/40
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
41/41 - 19s - loss: 1.0806 - accuracy: 0.4150 - val_loss: 0.7546 - val_accuracy:
0.7465 - lr: 0.0010 - 19s/epoch - 458ms/step
Epoch 2/40
41/41 - 12s - loss: 0.6418 - accuracy: 0.7207 - val_loss: 0.3707 - val_accuracy:
0.8877 - lr: 0.0010 - 12s/epoch - 301ms/step
Epoch 3/40
41/41 - 12s - loss: 0.4058 - accuracy: 0.8460 - val_loss: 0.2799 - val_accuracy:
0.9109 - lr: 0.0010 - 12s/epoch - 302ms/step
Epoch 4/40
41/41 - 12s - loss: 0.3446 - accuracy: 0.8690 - val_loss: 0.1865 - val_accuracy:
0.9421 - lr: 0.0010 - 12s/epoch - 300ms/step
Epoch 5/40
41/41 - 12s - loss: 0.3234 - accuracy: 0.8721 - val_loss: 0.2514 - val_accuracy:
0.9097 - lr: 0.0010 - 12s/epoch - 299ms/step
Epoch 6/40
41/41 - 13s - loss: 0.2555 - accuracy: 0.8970 - val_loss: 0.1649 - val_accuracy:
0.9583 - lr: 0.0010 - 13s/epoch - 308ms/step
Epoch 7/40
41/41 - 14s - loss: 0.2236 - accuracy: 0.9173 - val_loss: 0.1347 - val_accuracy:
0.9595 - lr: 0.0010 - 14s/epoch - 347ms/step
Epoch 8/40
```

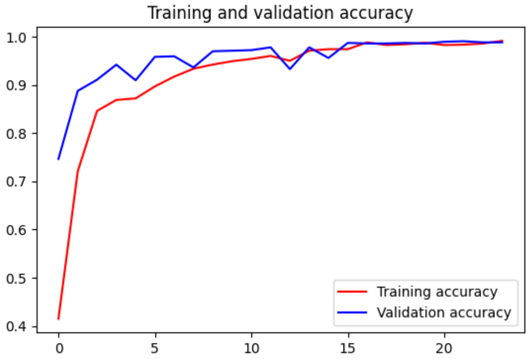
```
41/41 - 12s - loss: 0.1901 - accuracy: 0.9337 - val_loss: 0.2264 - val_accuracy:
0.9363 - lr: 0.0010 - 12s/epoch - 296ms/step
Epoch 9/40
41/41 - 12s - loss: 0.1712 - accuracy: 0.9423 - val_loss: 0.1010 - val_accuracy:
0.9699 - lr: 0.0010 - 12s/epoch - 292ms/step
Epoch 10/40
41/41 - 12s - loss: 0.1524 - accuracy: 0.9493 - val_loss: 0.0918 - val_accuracy:
0.9711 - lr: 0.0010 - 12s/epoch - 289ms/step
Epoch 11/40
41/41 - 12s - loss: 0.1318 - accuracy: 0.9540 - val_loss: 0.0802 - val_accuracy:
0.9722 - lr: 0.0010 - 12s/epoch - 295ms/step
Epoch 12/40
41/41 - 12s - loss: 0.1240 - accuracy: 0.9602 - val_loss: 0.0792 - val_accuracy:
0.9780 - lr: 0.0010 - 12s/epoch - 298ms/step
Epoch 13/40
41/41 - 12s - loss: 0.1387 - accuracy: 0.9501 - val_loss: 0.1911 - val_accuracy:
0.9329 - lr: 0.0010 - 12s/epoch - 298ms/step
Epoch 14/40
41/41 - 12s - loss: 0.0892 - accuracy: 0.9711 - val_loss: 0.0696 - val_accuracy:
0.9780 - lr: 0.0010 - 12s/epoch - 287ms/step
Epoch 15/40

Epoch 15: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
41/41 - 12s - loss: 0.0851 - accuracy: 0.9743 - val_loss: 0.1423 - val_accuracy:
0.9560 - lr: 0.0010 - 12s/epoch - 298ms/step
Epoch 16/40
41/41 - 12s - loss: 0.0705 - accuracy: 0.9743 - val_loss: 0.0567 - val_accuracy:
0.9873 - lr: 5.0000e-04 - 12s/epoch - 304ms/step
Epoch 17/40
41/41 - 11s - loss: 0.0551 - accuracy: 0.9883 - val_loss: 0.0565 - val_accuracy:
0.9861 - lr: 5.0000e-04 - 11s/epoch - 274ms/step
Epoch 18/40
41/41 - 12s - loss: 0.0501 - accuracy: 0.9828 - val_loss: 0.0550 - val_accuracy:
0.9861 - lr: 5.0000e-04 - 12s/epoch - 287ms/step
Epoch 19/40

Epoch 19: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
41/41 - 12s - loss: 0.0533 - accuracy: 0.9844 - val_loss: 0.0695 - val_accuracy:
0.9873 - lr: 5.0000e-04 - 12s/epoch - 281ms/step
Epoch 20/40
41/41 - 12s - loss: 0.0415 - accuracy: 0.9875 - val_loss: 0.0649 - val_accuracy:
0.9861 - lr: 2.5000e-04 - 12s/epoch - 296ms/step
Epoch 21/40
41/41 - 13s - loss: 0.0473 - accuracy: 0.9828 - val_loss: 0.0513 - val_accuracy:
0.9896 - lr: 2.5000e-04 - 13s/epoch - 309ms/step
Epoch 22/40
41/41 - 12s - loss: 0.0404 - accuracy: 0.9836 - val_loss: 0.0589 - val_accuracy:
0.9907 - lr: 2.5000e-04 - 12s/epoch - 298ms/step
```

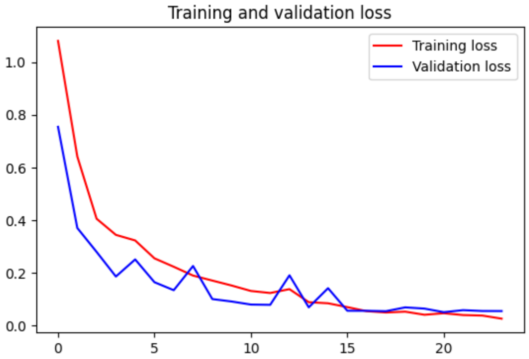
```
Epoch 23/40
41/41 - 12s - loss: 0.0385 - accuracy: 0.9860 - val_loss: 0.0558 - val_accuracy:
0.9884 - lr: 2.5000e-04 - 12s/epoch - 298ms/step
Epoch 24/40
41/41 - 14s - loss: 0.0267 - accuracy: 0.9914 - val_loss: 0.0555 - val_accuracy:
0.9884 - lr: 2.5000e-04 - 14s/epoch - 347ms/step
Training Time: 6.590774953365326 Minute
42/42 [==============================] - 10s 230ms/step - loss: 0.0162 -
accuracy: 0.9962
28/28 [==============================] - 2s 87ms/step - loss: 0.0583 - accuracy:
0.9908
The best model with the specified criteria
```

8. Plot Accuracy and Loss

[14]:

```python
def plotaccloss(history):
  acc = history.history['accuracy']
  val_acc = history.history['val_accuracy']
  loss = history.history['loss']
  val_loss = history.history['val_loss']

  epochs = range(len(acc))

  plt.figure(figsize=(14, 4))

  plt.subplot(1, 2, 1)
  plt.plot(epochs, acc, 'r', label='Training accuracy')
  plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
  plt.title('Training and validation accuracy')
  plt.legend(loc=0)

  plt.subplot(1, 2, 2)
  plt.plot(epochs, loss, 'r', label='Training loss')
  plt.plot(epochs, val_loss, 'b', label='Validation loss')
  plt.title('Training and validation loss')
  plt.legend(loc=0)

  plt.show()

plotaccloss(history)
```





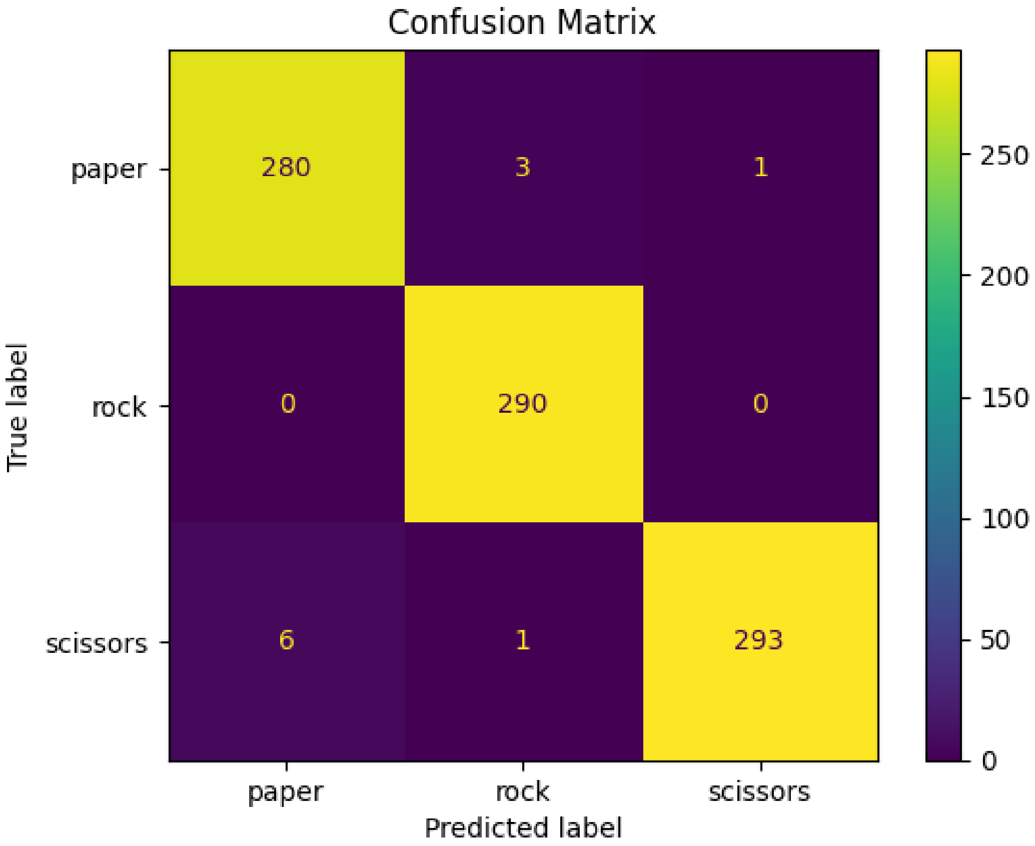
9. Evaluate Model

[12]:

```
def evaluate(model):
  validation_generator = train_datagen.flow_from_directory(
          base_dir,
          target_size=(150, 150),
          batch_size=32,
          class_mode='categorical',
          shuffle = False,
          subset='validation')

  batch_size = 32
  num_of_test_samples = len(validation_generator.filenames)

  Y_pred = model.predict(validation_generator, num_of_test_samples //␣
 ↪batch_size+1)
  y_pred = np.argmax(Y_pred, axis=1)

  print('\n\nClassification Report\n')
  target_names = list(validation_generator.class_indices.keys())
  print(classification_report(validation_generator.classes, y_pred,␣
 ↪target_names=target_names))

  print('\nConfusion Matrix\n')
  cm = confusion_matrix(validation_generator.classes, y_pred)
  print(cm)

  disp = ConfusionMatrixDisplay(confusion_matrix=cm,␣
 ↪display_labels=validation_generator.class_indices.keys())
  disp.plot()
  disp.ax_.set_title("Confusion Matrix")

evaluate(model)
```

```
Found 874 images belonging to 3 classes.
28/28 [==============================] - 7s 244ms/step


Classification Report

              precision    recall  f1-score   support

       paper       0.98      0.99      0.98       284
        rock       0.99      1.00      0.99       290
    scissors       1.00      0.98      0.99       300

    accuracy                           0.99       874
   macro avg       0.99      0.99      0.99       874
weighted avg       0.99      0.99      0.99       874
```

```
Confusion Matrix

[[280   3   1]
 [  0 290   0]
 [  6   1 293]]
```



10. Image Prediction

```
[15]: def imagepredict(model):
        uploaded = files.upload()

        for fn in uploaded.keys():

          path = fn
          img = image.load_img(path, target_size=(150,150))

          imgplot = plt.imshow(img)
```

```
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])

    classes = model.predict(images, batch_size=10)
    print(fn)
    if classes[0][0]>0.5:
      print(f'This is Paper!. With Probability {classes[0][0]*100}%')
    elif classes[0][1]>0.5:
      print(f'This is Rock!. With Probability {classes[0][1]*100}%')
    else:
      print(f'This is Scissors!. With Probability {classes[0][2]*100}%')

imagepredict(model)
```

```
<IPython.core.display.HTML object>
```

```
Saving scissors 1.jpeg to scissors 1.jpeg
1/1 [==============================] - 0s 17ms/step
scissors 1.jpeg
This is Scissors!. With Probability 100.0%
```