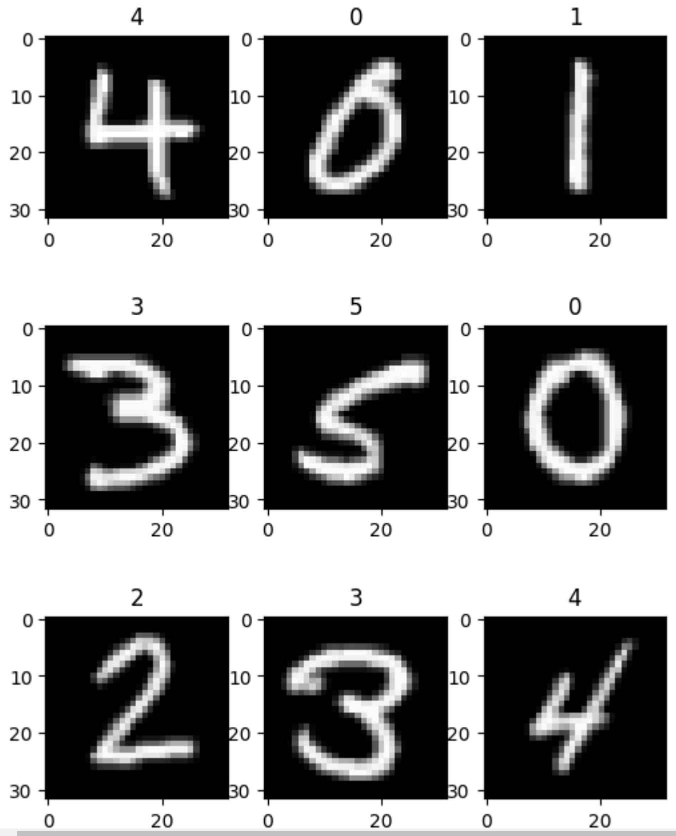
```
import os
import cv2
import time
import tensorflow as tf
import time
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras import (models,layers)
from tensorflow.keras.layers import (Dense,Flatten)
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.applications.vgg19 import VGG19
from matplotlib import pyplot as plt
import numpy as np
import random
from sklearn.metrics import confusion_matrix, classification_report
```

```
model_directory = 'models'
class RandomIntegers():
  def __init__(self):
      pass
  def generate(self, n, length):
    random_integers = random.sample(range(length), n)
    return random_integers
(x_train,y_train), (x_test, y_test)= mnist.load_data()
IMG_SIZE = 32
def resize(img_array):
  tmp = np.empty((img_array.shape[0], IMG_SIZE, IMG_SIZE))
  for i in range(len(img_array)):
    img = img_array[i].reshape(28, 28).astype('uint8')
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = img.astype('float32')/255
    tmp[i] = img
  return tmp
x_train = resize(x_train)
x_test = resize(x_test)

# Stack the images to create 3 channels
x_train = np.stack((x_train,)*3, axis=-1)
x_test = np.stack((x_test,)*3, axis=-1)

# Print the shapes to confirm the resizing
print(x_train.shape)
print(x_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
(60000, 32, 32, 3)
(10000, 32, 32, 3)
```

```
def display_images():
  random_integers = RandomIntegers().generate(9, len(x_train))
  plt.figure(figsize=(6, 8))
  counter = 0
  for i in random_integers:
    plt.subplot(330 + 1 + counter)
    counter += 1
    plt.imshow(x_train[i])
    plt.title(str(y_train[i]))
  plt.show()
display_images()
```



```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_train
vgg19 = VGG19(weights='imagenet', include_top= False,input_shape=(32,32,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.
80134624/80134624 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
model = Sequential()
model.add(layers.Input(shape=(32, 32, 3)))
model.add(vgg19)
model.add(Flatten())
model.add(Dense(10, activation ='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='sgd',metrics=['accuracy'])
model.summary()
```

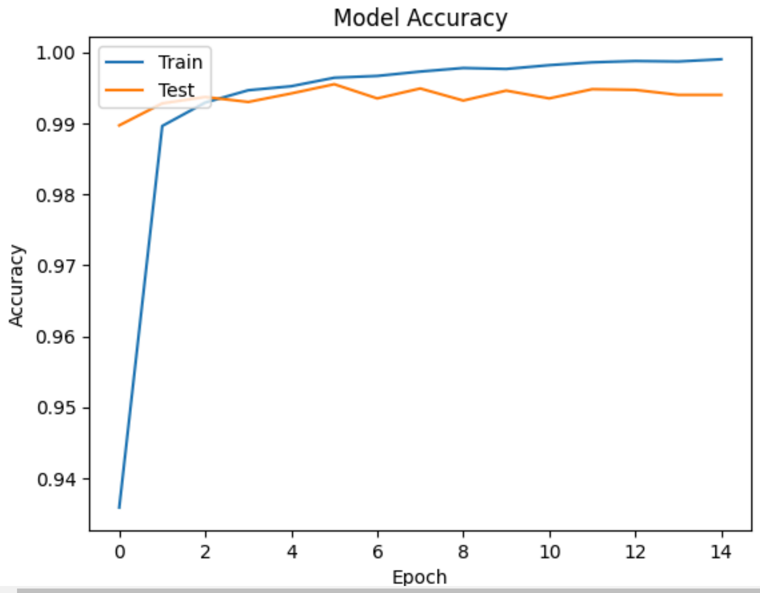
```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ vgg19 (Functional)                   │ (None, 1, 1, 512)           │      20,024,384 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 512)                 │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 10)                  │           5,130 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 20,029,514 (76.41 MB)
 Trainable params: 20,029,514 (76.41 MB)
```

```
import time
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

es = EarlyStopping(monitor='accuracy', verbose=1, patience=5)
mc = ModelCheckpoint(filepath='mnist-vgg19.keras', verbose=1, save_best_only=True, monitor='accuracy', mode='max')
cb = [es, mc]

start_time = time.time()
history = model.fit(
    x_train,
    y_train,
    epochs=15,
    batch_size=128,
    validation_data=(x_test, y_test),
    callbacks=cb
)
end_time = time.time()
```

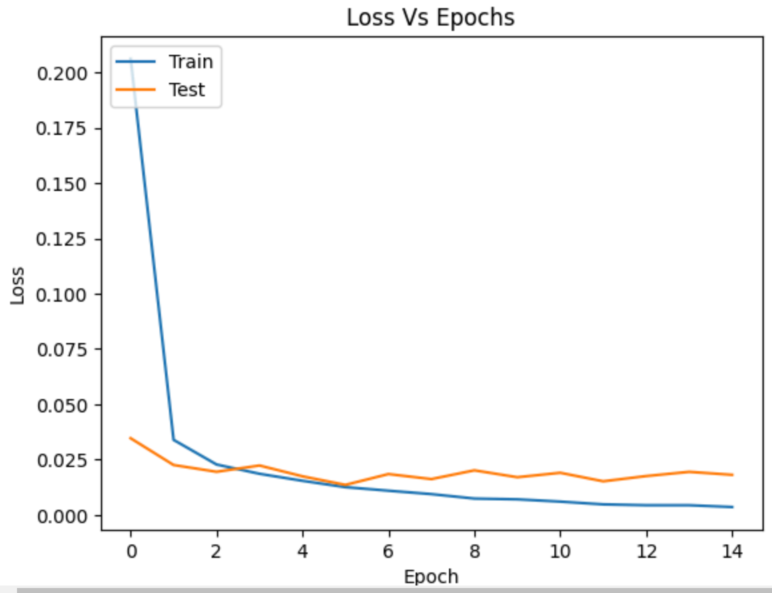
```
Epoch 1/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.8261 - loss: 0.5475
Epoch 1: accuracy improved from -inf to 0.93587, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 57s 96ms/step - accuracy: 0.8263 - loss: 0.5468 - val_accuracy: 0.9897 - val_loss: 0.0346
Epoch 2/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 0.9895 - loss: 0.0360
Epoch 2: accuracy improved from 0.93587 to 0.98962, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 36s 77ms/step - accuracy: 0.9895 - loss: 0.0360 - val_accuracy: 0.9928 - val_loss: 0.0226
Epoch 3/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9931 - loss: 0.0219
Epoch 3: accuracy improved from 0.98962 to 0.99292, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 35s 74ms/step - accuracy: 0.9931 - loss: 0.0219 - val_accuracy: 0.9937 - val_loss: 0.0195
Epoch 4/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9942 - loss: 0.0199
Epoch 4: accuracy improved from 0.99292 to 0.99465, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 35s 74ms/step - accuracy: 0.9942 - loss: 0.0199 - val_accuracy: 0.9930 - val_loss: 0.0223
Epoch 5/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9950 - loss: 0.0160
Epoch 5: accuracy improved from 0.99465 to 0.99522, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 74ms/step - accuracy: 0.9950 - loss: 0.0160 - val_accuracy: 0.9942 - val_loss: 0.0174
Epoch 6/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9964 - loss: 0.0128
Epoch 6: accuracy improved from 0.99522 to 0.99642, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 74ms/step - accuracy: 0.9964 - loss: 0.0128 - val_accuracy: 0.9955 - val_loss: 0.0136
Epoch 7/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9967 - loss: 0.0106
Epoch 7: accuracy improved from 0.99642 to 0.99667, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 74ms/step - accuracy: 0.9967 - loss: 0.0106 - val_accuracy: 0.9935 - val_loss: 0.0185
Epoch 8/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 0.9979 - loss: 0.0074
Epoch 8: accuracy improved from 0.99667 to 0.99728, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 43s 78ms/step - accuracy: 0.9979 - loss: 0.0074 - val_accuracy: 0.9949 - val_loss: 0.0162
Epoch 9/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9978 - loss: 0.0072
Epoch 9: accuracy improved from 0.99728 to 0.99778, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 74ms/step - accuracy: 0.9978 - loss: 0.0072 - val_accuracy: 0.9932 - val_loss: 0.0201
Epoch 10/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9980 - loss: 0.0067
Epoch 10: accuracy did not improve from 0.99778
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 73ms/step - accuracy: 0.9980 - loss: 0.0067 - val_accuracy: 0.9946 - val_loss: 0.0170
Epoch 11/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9986 - loss: 0.0049
Epoch 11: accuracy improved from 0.99778 to 0.99818, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 74ms/step - accuracy: 0.9986 - loss: 0.0049 - val_accuracy: 0.9935 - val_loss: 0.0190
Epoch 12/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9985 - loss: 0.0047
Epoch 12: accuracy improved from 0.99818 to 0.99858, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 42s 77ms/step - accuracy: 0.9985 - loss: 0.0047 - val_accuracy: 0.9948 - val_loss: 0.0152
Epoch 13/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step - accuracy: 0.9990 - loss: 0.0035
Epoch 13: accuracy improved from 0.99858 to 0.99877, saving model to mnist-vgg19.keras
469/469 ━━━━━━━━━━━━━━━━━━━━ 41s 77ms/step - accuracy: 0.9990 - loss: 0.0035 - val_accuracy: 0.9947 - val_loss: 0.0175
Epoch 14/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9990 - loss: 0.0037
Epoch 14: accuracy did not improve from 0.99877
469/469 ━━━━━━━━━━━━━━━━━━━━ 39s 73ms/step - accuracy: 0.9990 - loss: 0.0037 - val_accuracy: 0.9940 - val_loss: 0.0195
Epoch 15/15
469/469 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step - accuracy: 0.9993 - loss: 0.0032
```

```
total_time = end_time - start_time
print("Time taken for training: ", total_time, " seconds")
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
Time taken for training:  624.2155239582062  seconds
```



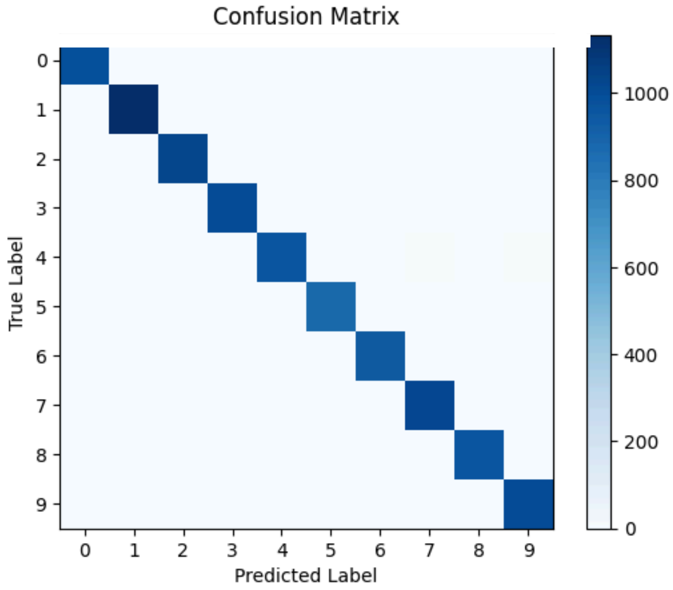
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss Vs Epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



```
if not os.path.exists(model_directory):
  os.makedirs(model_directory)
model_path = os.path.join(model_directory,"model_mnist_vgg.h5"
model.save(model_path)
model.summary()
true_labels = np.argmax(y_test, axis=-1)
predicted_labels = np.argmax(model.predict(x_test), axis=-1)
cm = confusion_matrix(true_labels, predicted_labels)
plt.imshow(cm, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(np.arange(10))
plt.yticks(np.arange(10))
plt.show()
```

```
report = classification_report(true_labels, predicted_labels)
```

```
WARNING:absl:You are saving your model as an HDF5 file vi  `model.save()` or `keras.saving.save_model(model)`. This file format is co
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━  ━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                  ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━  ━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ vgg19 (Functional)                    │ (None, 1, 1, 512              │     20,024,384 │
├───────────────────────────────────────┼─────────────────  ────────────┼────────────────┤
│ flatten (Flatten)                     │ (None, 512)                   │              0 │
├───────────────────────────────────────┼─────────────────  ────────────┼────────────────┤
│ dense (Dense)                         │ (None, 10)                    │          5,130 │
└───────────────────────────────────────┴─────────────────  ────────────┴────────────────┘
 Total params: 20,029,516 (76.41 MB)
 Trainable params: 20,029,514 (76.41 MB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2 (12.00 B)
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step
```



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       980
           1       1.00      1.00      1.00      1135
           2       0.99      1.00      0.99      1032
           3       0.99      1.00      0.99      1010
           4       1.00      0.99      0.99       982
           5       1.00      0.99      0.99       892
           6       1.00      0.99      1.00       958
           7       0.99      0.99      0.99      1028
           8       1.00      0.99      1.00       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```