```
!pip install tensorflow keras
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.1)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (3.5.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.68.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras) (0.13.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (202
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=2.
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.utils import to_categorical
import numpy as np
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.utils import to_categorical
```

```
 # Load the MNIST dataset
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
 # Preprocess the data
X_train = np.expand_dims(X_train, axis=-1)  # Add channel dimension (28, 28, 1)
X_test = np.expand_dims(X_test, axis=-1)
X_train = tf.image.resize(X_train, (128, 128))
X_test = tf.image.resize(X_test, (128, 128))

# Normalize pixel values
X_train = X_train / 255.0
X_test = X_test / 255.0
# Replicate the grayscale channel to create 3 channels
X_train = tf.repeat(X_train, 3, axis=-1) # Repeat grayscale channel 3 times
X_test = tf.repeat(X_test, 3, axis=-1)
# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
# Print shape of the dataset
print(f"x_train_resized shape: {X_train.shape}")
```

```
print(f"x_test_resized shape: {X_test.shape}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

Start coding or generate with AI.

```
# Load the VGG16 model pre-trained on ImageNet, excluding the top layer
vgg_base = VGG16(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
# Freeze the base layers (optional)
for layer in vgg_base.layers:
   layer.trainable = False
# Build a new model on top of VGG16 for MNIST classification
model = Sequential([
   vgg_base,
   Flatten(),
   Dense(256, activation='relu'),
   Dropout(0.5),
   Dense(10, activation='softmax')  # 10 output classes for MNIST
])
# Compile the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.summary()
```

```
history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 505s 269ms/step - loss: 0.1973 - accuracy: 0.9493 - val_loss: 0.0444 - val_accuracy: 0.9879
Epoch 2/5
1875/1875 [==============================] - 517s 276ms/step - loss: 0.0596 - accuracy: 0.9827 - val_loss: 0.0338 - val_accuracy: 0.9898
Epoch 3/5
1875/1875 [==============================] - 517s 276ms/step - loss: 0.0462 - accuracy: 0.9859 - val_loss: 0.0288 - val_accuracy: 0.9908
Epoch 4/5
1875/1875 [==============================] - 507s 270ms/step - loss: 0.0376 - accuracy: 0.9887 - val_loss: 0.0247 - val_accuracy: 0.9915
Epoch 5/5
1875/1875 [==============================] - 507s 271ms/step - loss: 0.0321 - accuracy: 0.9903 - val_loss: 0.0257 - val_accuracy: 0.9914
```

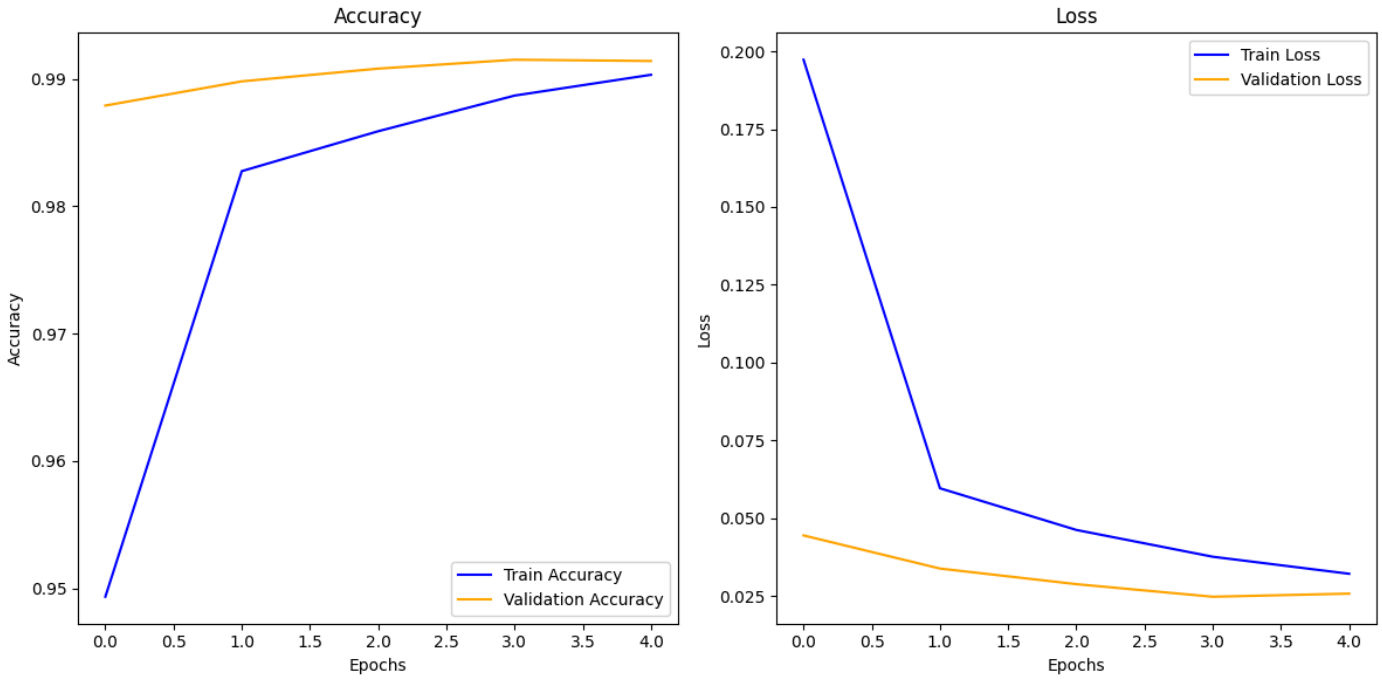
```
import matplotlib.pyplot as plt

# Plot the training and validation accuracy and loss
plt.figure(figsize=(12, 6))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy', color='blue')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='orange')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss', color='blue')
plt.plot(history.history['val_loss'], label='Validation Loss', color='orange')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```




```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test,y_test)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')
```

```
313/313 [==============================] - 73s 234ms/step - loss: 0.0257 - accuracy: 0.9914
Test Loss: 0.025747347623109818
Test Accuracy: 0.9914000034332275
```

Fine Tunning

```
 # Unfreeze some layers for fine-tuning
for layer in model.layers[0].layers[-4:]:
  layer.trainable = True
```

```
 # Recompile the model with a lower learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
loss='categorical_crossentropy',
metrics=['accuracy'])
 # Retrain the model
history_fine_tune = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 743s 395ms/step - loss: 0.0259 - accuracy: 0.9920 - val_loss: 0.0209 - val_accuracy: 0.9931
Epoch 2/5
1875/1875 [==============================] - 745s 397ms/step - loss: 0.0151 - accuracy: 0.9949 - val_loss: 0.0189 - val_accuracy: 0.9933
Epoch 3/5
1875/1875 [==============================] - 731s 390ms/step - loss: 0.0113 - accuracy: 0.9963 - val_loss: 0.0194 - val_accuracy: 0.9942
Epoch 4/5
1875/1875 [==============================] - 732s 391ms/step - loss: 0.0073 - accuracy: 0.9979 - val_loss: 0.0179 - val_accuracy: 0.9942
Epoch 5/5
1875/1875 [==============================] - 742s 396ms/step - loss: 0.0068 - accuracy: 0.9978 - val_loss: 0.0188 - val_accuracy: 0.9943
```

```
 # Evaluate the fine-tuned model
 test_loss, test_accuracy = model.evaluate(X_test, y_test)
 print(f'Test accuracy after fine-tuning: {test_accuracy:.4f}')
```

```
313/313 [==============================] - 72s 229ms/step - loss: 0.0188 - accuracy: 0.9943
Test accuracy after fine-tuning: 0.9943
```

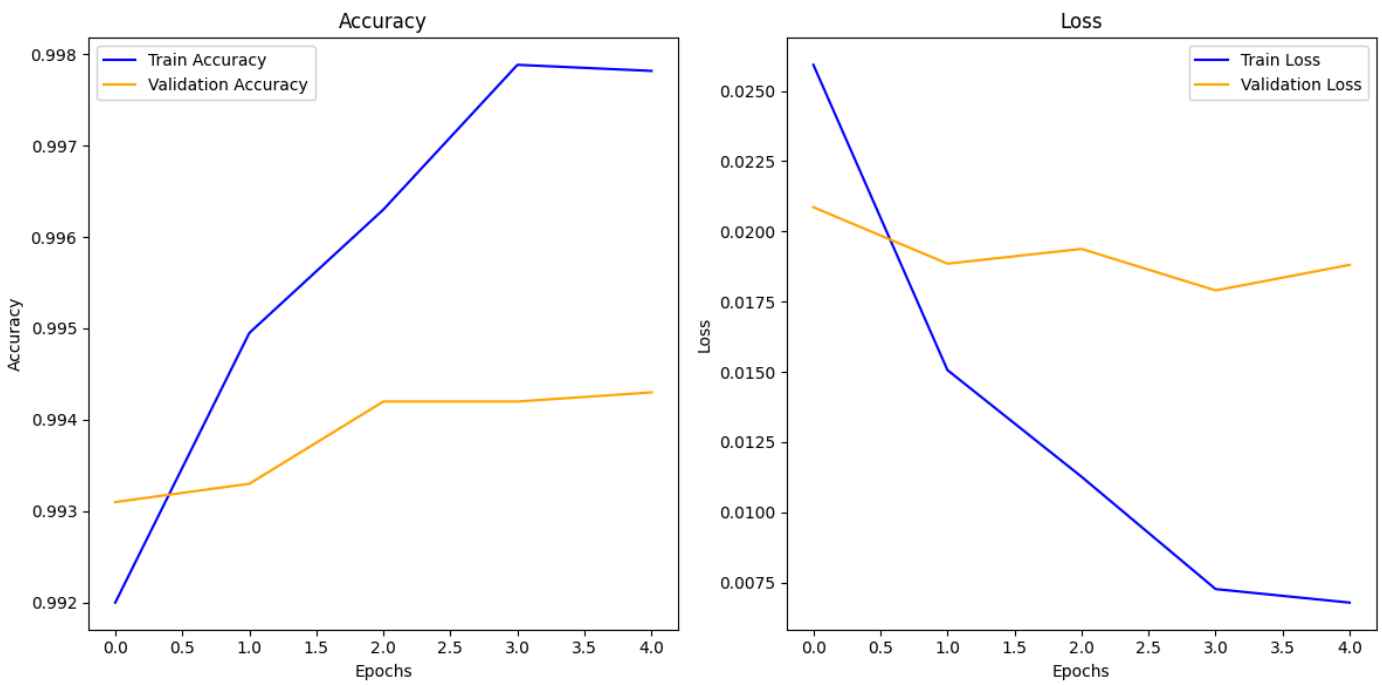
```
import matplotlib.pyplot as plt

# Plot the training and validation accuracy and loss
plt.figure(figsize=(12, 6))

# Accuracy plot
plt.subplot(1, 2, 1)
plt.plot(history_fine_tune.history['accuracy'], label='Train Accuracy', color='blue')
plt.plot(history_fine_tune.history['val_accuracy'], label='Validation Accuracy', color='orange')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 2, 2)
plt.plot(history_fine_tune.history['loss'], label='Train Loss', color='blue')
plt.plot(history_fine_tune.history['val_loss'], label='Validation Loss', color='orange')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



Data Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
 # Set up data augmentation
datagen = ImageDataGenerator(rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, zoom_range=0.1)
train_generator = datagen.flow(X_train, y_train, batch_size=32)
 # Train the model using augmented data
history_augmented = model.fit(train_generator, epochs=5, validation_data=(X_test, y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 747s 398ms/step - loss: 0.0223 - accuracy: 0.9928 - val_loss: 0.0180 - val_accuracy: 0.9941
Epoch 2/5
1875/1875 [==============================] - ETA: 0s - loss: 0.0134 - accuracy: 0.9957
```

Start coding or generate with AI.