```
pip install keras tensorflow
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (3.8.0)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras) (3.13.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras) (0.14.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras) (24.2)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from tensor
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
```

```
import tensorflow as tf
print(tf.keras.__version__)
```

```
3.8.0
```

```
import tensorflow as tf

x = tf.Variable(5.0)  # Trainable variable
with tf.GradientTape() as tape:
    fy = x ** 2  # y = x^2

grad = tape.gradient(fy, x)  # Use fy instead of y
print(grad.numpy())  # Output: 10.0
```

```
10.0
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
!unzip "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/Copy of devnagari digit.zip" -d "/content/drive/MyDrive/Artificial Intelligence and Machine
```

```
Streaming output truncated to the last 5000 lines.
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10299.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10300.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10301.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10302.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10303.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10304.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10306.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/10307.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104017.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104018.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104019.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104021.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104023.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104024.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104025.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104026.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104027.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104028.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104029.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104030.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104031.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104032.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104033.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104034.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104057.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104058.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104059.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104060.png
  inflating: /content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_7/104061.png
```

```
!ls "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Test"
```

```
digit_0  digit_1  digit_2  digit_3  digit_4  digit_5  digit_6  digit_7  digit_8  digit_9
```

```
!ls "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train/digit_0"
```

```
103265.png  12335.png  28867.png  39442.png  4575.png   49273.png  6783.png   76339.png  9506.png
103266.png  12336.png  28868.png  39443.png  4576.png   49274.png  6784.png   76340.png  9507.png
103267.png  12337.png  28870.png  39444.png  4577.png   49275.png  6785.png   76341.png  9508.png
103268.png  12338.png  28873.png  39446.png  4578.png   49277.png  6786.png   76342.png  9509.png
103269.png  12339.png  28874.png  39447.png  4579.png   49278.png  6787.png   76343.png  9510.png
103270.png  12340.png  28876.png  39448.png  4580.png   49279.png  6788.png   76344.png  9511.png
103271.png  12341.png  28877.png  39449.png  4581.png   49280.png  6789.png   76345.png  9512.png
103272.png  12342.png  28878.png  39450.png  4582.png   49281.png  6790.png   76346.png  9513.png
103273.png  12343.png  28879.png  39451.png  4583.png   49282.png  6791.png   76347.png  9514.png
103274.png  12344.png  28880.png  39452.png  4585.png   49283.png  6792.png   76348.png  9515.png
103275.png  12345.png  28882.png  39453.png  4586.png   49284.png  6793.png   76349.png  9516.png
103276.png  12347.png  28883.png  39454.png  4587.png   49285.png  6794.png   76350.png  9517.png
103279.png  12349.png  28885.png  39455.png  4588.png   49286.png  6795.png   76351.png  9518.png
103280.png  12351.png  28887.png  39456.png  4589.png   49287.png  6796.png   76352.png  9519.png
103282.png  12353.png  28888.png  39458.png  4590.png   49289.png  6797.png   76353.png  9520.png
103283.png  12354.png  28890.png  39459.png  4591.png   49290.png  6798.png   76354.png  9521.png
103284.png  12355.png  28891.png  39460.png  4592.png   49291.png  6799.png   76355.png  9522.png
103305.png  12356.png  28892.png  39462.png  4595.png   49292.png  6800.png   76356.png  9523.png
103306.png  12357.png  28893.png  39463.png  4596.png   49294.png  6801.png   76357.png  9524.png
103307.png  12359.png  28894.png  39464.png  4597.png   49295.png  6802.png   76359.png  9525.png
103308.png  12360.png  28895.png  39465.png  4598.png   49296.png  6803.png   76360.png  9526.png
103310.png  12361.png  28896.png  39466.png  4599.png   49297.png  6804.png   76361.png  9527.png
103312.png  12362.png  28897.png  39468.png  4601.png   49298.png  6806.png   76362.png  9528.png
103313.png  12363.png  28898.png  39469.png  4602.png   49299.png  6807.png   76363.png  9530.png
103314.png  12364.png  28899.png  39470.png  4603.png   49300.png  6808.png   76364.png  9532.png
103315.png  12365.png  28900.png  39472.png  4604.png   49301.png  6809.png   85645.png  9533.png
```

```python
from PIL import Image
import os
import numpy as np

# Path to the dataset folder
dataset_path = "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train"

# Placeholder for images and labels
images = []
labels = []

# Iterate through each folder and file
for label_folder in os.listdir(dataset_path):
    folder_path = os.path.join(dataset_path, label_folder)
    if os.path.isdir(folder_path):  # Ensure it's a directory
        for file_name in os.listdir(folder_path):
            file_path = os.path.join(folder_path, file_name)
            if file_name.endswith('.png'):  # Assuming images are PNG files
                # Open and process image
                image = Image.open(file_path).convert('L')  # Convert to grayscale
                image_array = np.array(image)  # Convert to numpy array
                images.append(image_array)
                labels.append(label_folder)  # Use folder name as label

# Convert lists to numpy arrays
images = np.array(images)
labels = np.array(labels)

print(f"Loaded {len(images)} images with labels.")
```

```
print(f"Loaded {len(images)} images with labels.")
```

```
Loaded 17000 images with labels.
```

```
from PIL import Image
import os
import numpy as np
from tensorflow.keras.utils import to_categorical
from sklearn.utils import shuffle

# Function to load, process, and normalize images
def load_images_and_labels(folder_path, target_size=(28, 28)):
    images = []
    labels = []
    label_mapping = {}  # To map folder names to numerical labels
    current_label = 0

    # Iterate through folders (each folder represents a class)
    for label_folder in sorted(os.listdir(folder_path)):  # Ensure consistent label order
        label_path = os.path.join(folder_path, label_folder)
        if os.path.isdir(label_path):
            # Map folder name to a numerical label
            if label_folder not in label_mapping:
                label_mapping[label_folder] = current_label
                current_label += 1

            for file_name in os.listdir(label_path):
                file_path = os.path.join(label_path, file_name)
                if file_name.endswith('.png'):  # Check for PNG files
                    # Load image, resize to target size, and convert to grayscale
                    image = Image.open(file_path).convert('L').resize(target_size)
                    image_array = np.array(image, dtype=np.float32)  # Convert to NumPy array
                    image_array /= 255.0  # Normalize to range 0-1
                    images.append(image_array)
                    labels.append(label_mapping[label_folder])  # Assign numerical label

    # Convert to NumPy arrays
    images = np.array(images)
    labels = np.array(labels)

    return images, labels, label_mapping

# Paths to the train and test folders
train_folder = "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Train"
test_folder = "/content/drive/MyDrive/Artificial Intelligence and Machine Learning/week 4/DevanagariHandwrittenDigitDataset/Test"

# Load and process training and testing data
train_images, train_labels, label_mapping = load_images_and_labels(train_folder)
test_images, test_labels, _ = load_images_and_labels(test_folder)

# ✅ Shuffle after loading!
train_images, train_labels = shuffle(train_images, train_labels, random_state=42)

# One-hot encode the labels for multi-class classification
train_labels = to_categorical(train_labels, num_classes=len(label_mapping))
test_labels = to_categorical(test_labels, num_classes=len(label_mapping))
```

```
# Confirm everything looks good
print(f"Training set: {train_images.shape[0]} images, {train_labels.shape[0]} labels.")
print(f"Testing set: {test_images.shape[0]} images, {test_labels.shape[0]} labels.")
print(f"Label Mapping: {label_mapping}")
print(os.listdir(train_folder))
```

```
Training set: 17000 images, 17000 labels.
Testing set: 3000 images, 3000 labels.
Label Mapping: {'digit_0': 0, 'digit_1': 1, 'digit_2': 2, 'digit_3': 3, 'digit_4': 4, 'digit_5': 5, 'digit_6': 6, 'digit_7': 7, 'digit_8': 8, 'digit_9': 9}
['digit_0', 'digit_1', 'digit_2', 'digit_3', 'digit_4', 'digit_5', 'digit_6', 'digit_7', 'digit_8', 'digit_9']
```

```
from tensorflow.keras.models import Sequential

# Initialize the Sequential model
model = Sequential()

print("Sequential model initialized.")
```

```
Sequential model initialized.
```

```
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(64, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
model.summary()
```

```
model.summary()
```

Model: "sequential_8"

Layer (type)	Output Shape	Param #
flatten_10 (Flatten)	(None, 784)	0
dense_44 (Dense)	(None, 64)	50,240
dense_45 (Dense)	(None, 128)	8,320
dense_46 (Dense)	(None, 256)	33,024
dense_47 (Dense)	(None, 10)	2,570

Total params: 94,154 (367.79 KB)
Trainable params: 94,154 (367.79 KB)
Non-trainable params: 0 (0.00 B)

```
from tensorflow.keras.optimizers import Adam

# Compile the previously built model
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])


print("Model compiled successfully.")
```

```
Model compiled successfully.
```

```
print(f"Shape of train_images: {train_images.shape}")
print(f"Shape of train_labels: {train_labels.shape}")
```

```
Shape of train_images: (17000, 28, 28)
Shape of train_labels: (17000, 10)
```

```python
# Set up callbacks
checkpoint = ModelCheckpoint("best_model.h5", monitor="val_loss", save_best_only=True)
early_stopping = EarlyStopping(monitor="val_loss", patience=3, restore_best_weights=True)
callbacks = [checkpoint, early_stopping]

# Train the model
history = model.fit(
    train_images, train_labels,
    batch_size=128,
    epochs=20,
    validation_split=0.2,
    callbacks=[checkpoint, early_stopping]
)

# Print summary
print(model.summary())
```

```
Epoch 1/20
105/107 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6910 - loss: 1.0358WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 0.6952 - loss: 1.0219 - val_accuracy: 0.9426 - val_loss: 0.1887
Epoch 2/20
 99/107 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9462 - loss: 0.1733WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9467 - loss: 0.1717 - val_accuracy: 0.9579 - val_loss: 0.1337
Epoch 3/20
103/107 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9722 - loss: 0.1015WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9722 - loss: 0.1013 - val_accuracy: 0.9694 - val_loss: 0.0937
Epoch 4/20
 96/107 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9802 - loss: 0.0673WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9802 - loss: 0.0674 - val_accuracy: 0.9738 - val_loss: 0.0836
Epoch 5/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9886 - loss: 0.0455 - val_accuracy: 0.9726 - val_loss: 0.0929
Epoch 6/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9863 - loss: 0.0448 - val_accuracy: 0.9726 - val_loss: 0.0886
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.9948 - loss: 0.0193 - val_accuracy: 0.9765 - val_loss: 0.074
Epoch 10/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9954 - loss: 0.0166 - val_accuracy: 0.9829 - val_loss: 0.067
Model: "sequential_8"
```

Layer (type)	Output Shape	Param #
flatten_10 (Flatten)	(None, 784)	0
dense_44 (Dense)	(None, 64)	50,240
dense_45 (Dense)	(None, 128)	8,320
dense_46 (Dense)	(None, 256)	33,024
dense_47 (Dense)	(None, 10)	2,570

```
Total params: 282,464 (1.08 MB)
Trainable params: 94,154 (367.79 KB)
Non-trainable params: 0 (0.00 B)
```

```python
test_loss, test_acc = model.evaluate(train_images, train_labels, verbose=2)
print(f"Test accuracy: {test_acc:.4f}")
```

```
532/532 - 1s - 2ms/step - accuracy: 0.9929 - loss: 0.0282
Test accuracy: 0.9929
```

```python
# Save model as .h5 file
model.save("devanagari_model.h5")
print("Model saved as devanagari_model.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
Model saved as devanagari_model.h5
```

```python
from tensorflow.keras.models import load_model

# Load the .h5 model
model = load_model("devanagari_model.h5")
print("Model loaded from devanagari_model.h5")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Model loaded from devanagari_model.h5
```

```python
# Use the model to make predictions on the test set
predictions = model.predict(train_images)

# Convert predictions from probabilities to digit labels
predicted_labels = np.argmax(predictions, axis=1)

# Check the first prediction
print(f"Predicted label for first image: {predicted_labels[0]}")
print(f"True label for first image: {np.argmax(train_labels[0])}")
```

```
532/532 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Predicted label for first image: 6
True label for first image: 6
```

```python
import matplotlib.pyplot as plt

train_loss = history.history['loss']
val_loss = history.history['val_loss']

train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(range(1, len(train_loss) + 1), train_loss, label='Training Loss', color='blue')
plt.plot(range(1, len(val_loss) + 1), val_loss, label='Validation Loss', color='orange')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(range(1, len(train_acc) + 1), train_acc, label='Training Accuracy', color='blue')
plt.plot(range(1, len(val_acc) + 1), val_acc, label='Validation Accuracy', color='orange')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.tight_layout()
plt.show()
```