Load and Preprocess Data

In [60]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load data from CSV files
data_df = pd.read_csv('train.csv')

# Split data into features (X) and labels (y)
X = data_df.drop('label', axis=1).values
y = data_df['label'].values

# Split data into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
```

Create a Custom Neural Network Model

In [71]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Create a simple neural network model
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    Dense(128, activation='relu'),
    Dense(105, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Train the Model

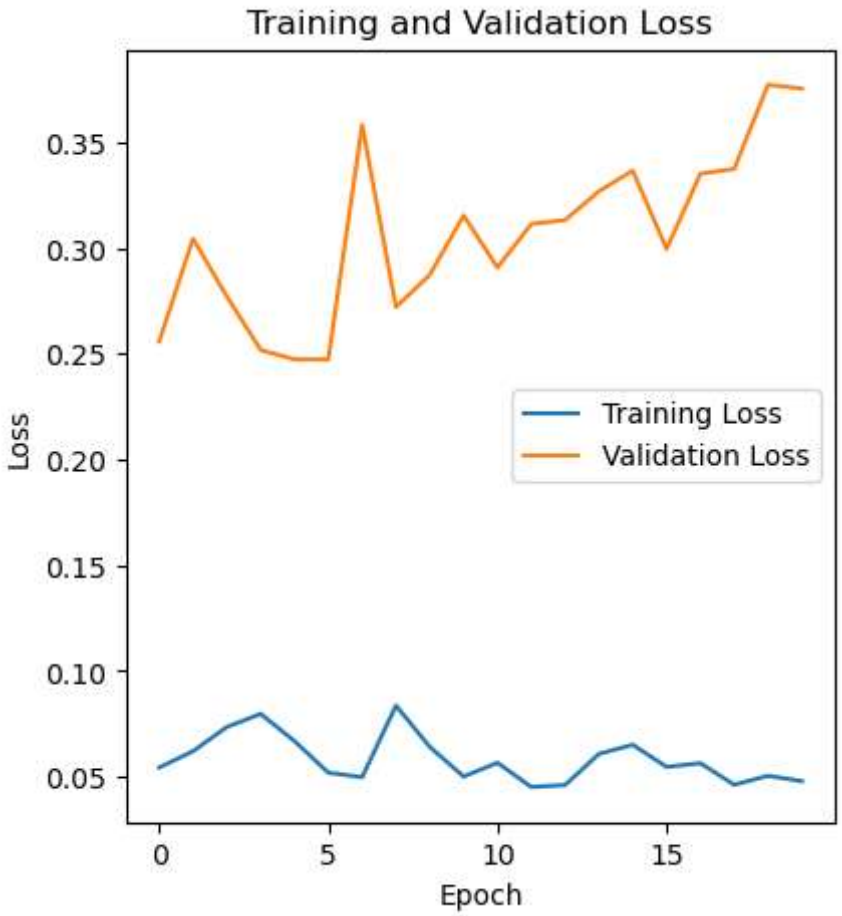
In [80]:
```
batch_size = 64
num_epochs = 20

# Train the model
history = model.fit(X_train, y_train, batch_size=batch_size, epochs=num_epochs, ve
```

```
Epoch 1/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0544 - accuracy:
0.9849 - val_loss: 0.2558 - val_accuracy: 0.9588
Epoch 2/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0621 - accuracy:
0.9849 - val_loss: 0.3043 - val_accuracy: 0.9537
Epoch 3/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0736 - accuracy:
0.9812 - val_loss: 0.2771 - val_accuracy: 0.9571
Epoch 4/20
460/460 [==============================] - 4s 9ms/step - loss: 0.0798 - accuracy:
0.9806 - val_loss: 0.2518 - val_accuracy: 0.9609
Epoch 5/20
460/460 [==============================] - 3s 7ms/step - loss: 0.0669 - accuracy:
0.9830 - val_loss: 0.2473 - val_accuracy: 0.9583
Epoch 6/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0520 - accuracy:
0.9860 - val_loss: 0.2473 - val_accuracy: 0.9652
Epoch 7/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0499 - accuracy:
0.9878 - val_loss: 0.3582 - val_accuracy: 0.9551
Epoch 8/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0837 - accuracy:
0.9805 - val_loss: 0.2721 - val_accuracy: 0.9614
Epoch 9/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0642 - accuracy:
0.9852 - val_loss: 0.2873 - val_accuracy: 0.9576
Epoch 10/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0501 - accuracy:
0.9865 - val_loss: 0.3154 - val_accuracy: 0.9614
Epoch 11/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0567 - accuracy:
0.9876 - val_loss: 0.2907 - val_accuracy: 0.9640
Epoch 12/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0452 - accuracy:
0.9896 - val_loss: 0.3113 - val_accuracy: 0.9619
Epoch 13/20
460/460 [==============================] - 3s 5ms/step - loss: 0.0461 - accuracy:
0.9886 - val_loss: 0.3132 - val_accuracy: 0.9581
Epoch 14/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0609 - accuracy:
0.9861 - val_loss: 0.3268 - val_accuracy: 0.9574
Epoch 15/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0652 - accuracy:
0.9856 - val_loss: 0.3365 - val_accuracy: 0.9614
Epoch 16/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0547 - accuracy:
0.9885 - val_loss: 0.2995 - val_accuracy: 0.9640
Epoch 17/20
460/460 [==============================] - 3s 6ms/step - loss: 0.0565 - accuracy:
0.9880 - val_loss: 0.3351 - val_accuracy: 0.9628
Epoch 18/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0463 - accuracy:
0.9891 - val_loss: 0.3374 - val_accuracy: 0.9582
Epoch 19/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0505 - accuracy:
0.9885 - val_loss: 0.3771 - val_accuracy: 0.9645
Epoch 20/20
460/460 [==============================] - 2s 5ms/step - loss: 0.0480 - accuracy:
0.9889 - val_loss: 0.3753 - val_accuracy: 0.9648
```

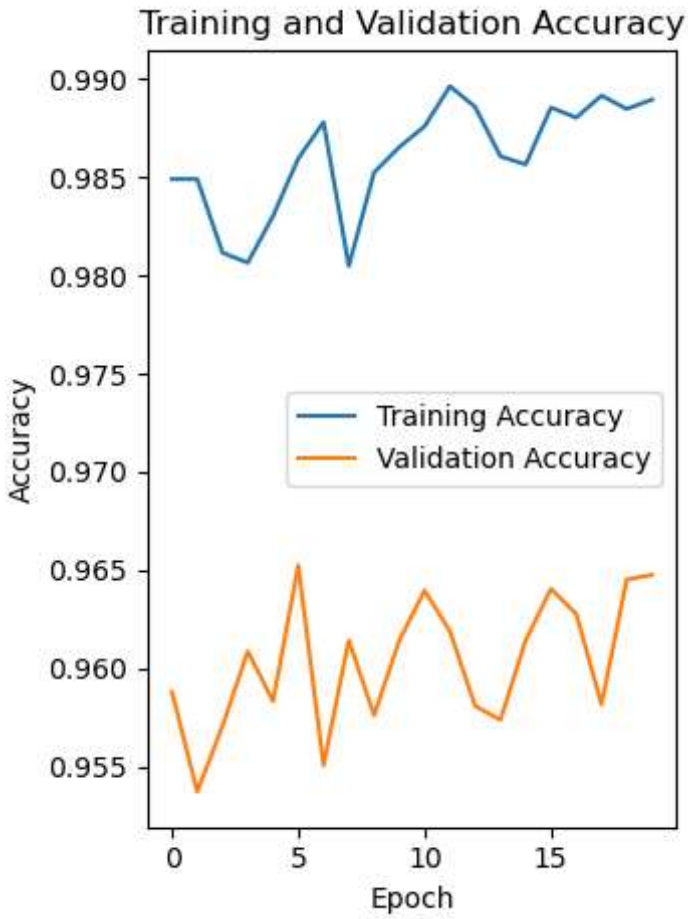
Plot Training Progress

In [81]:
```
import matplotlib.pyplot as plt

# Plot the training loss and accuracy
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
```

Out[81]:
```
<matplotlib.legend.Legend at 0x29ad6352220>
```



In [82]:
```
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

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



Evaluate the Model

In [84]:
```
# Evaluate the model
accuracy = model.evaluate(X_test, y_test, verbose=0)[1]
print(f"Accuracy on test data: {accuracy * 100:.2f}%")
```

```
Accuracy on test data: 96.48%
```