In [1]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, Input
import matplotlib.pyplot as plt
```

In [2]:
```
# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.loac
train_images, test_images = train_images / 255.0, test_images / 255.0
```

In [3]:
```
# Add channel dimension to the images
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))
```

In [4]:
```
# Split the dataset into training and validation sets
train_images, val_images, train_labels, val_labels = train_test_split( train_i
```

In [5]:
```
# Data augmentation for training images
datagen = ImageDataGenerator(rotation_range=10, zoom_range=0.1, width_shift_ra
datagen.fit(train_images)
```

In [6]:
```
# Create a CNN model with hyperparameter tuning and regularization
model = models.Sequential()

# Input layer
model.add(Input(shape=(28, 28, 1)))

model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

In [7]:
```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [8]:

```
# Train the model with data augmentation
history = model.fit(
    datagen.flow(train_images, train_labels, batch_size=64),
    epochs=20,
    validation_data=(val_images, val_labels)
)
```

```
C:\Python310\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_a
dapter.py:121: UserWarning: Your `PyDataset` class should call `super().__in
it__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_m
ultiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, a
s they will be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 59s 72ms/step - accuracy: 0.7399 - loss: 0.7824
- val_accuracy: 0.9766 - val_loss: 0.0700
Epoch 2/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 53s 70ms/step - accuracy: 0.9529 - loss: 0.1516
- val_accuracy: 0.9867 - val_loss: 0.0407
Epoch 3/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 53s 71ms/step - accuracy: 0.9668 - loss: 0.1071
- val_accuracy: 0.9872 - val_loss: 0.0394
Epoch 4/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 70s 93ms/step - accuracy: 0.9736 - loss: 0.0868
- val_accuracy: 0.9888 - val_loss: 0.0391
Epoch 5/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 56s 75ms/step - accuracy: 0.9776 - loss: 0.0727
- val_accuracy: 0.9898 - val_loss: 0.0367
Epoch 6/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 74ms/step - accuracy: 0.9799 - loss: 0.0653
- val_accuracy: 0.9903 - val_loss: 0.0346
Epoch 7/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 54s 72ms/step - accuracy: 0.9798 - loss: 0.0658
- val_accuracy: 0.9911 - val_loss: 0.0304
Epoch 8/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 73ms/step - accuracy: 0.9834 - loss: 0.0517
- val_accuracy: 0.9906 - val_loss: 0.0267
Epoch 9/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 57s 75ms/step - accuracy: 0.9843 - loss: 0.0552
- val_accuracy: 0.9924 - val_loss: 0.0259
Epoch 10/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 56s 75ms/step - accuracy: 0.9830 - loss: 0.0539
- val_accuracy: 0.9920 - val_loss: 0.0286
Epoch 11/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 56s 75ms/step - accuracy: 0.9858 - loss: 0.0468
- val_accuracy: 0.9933 - val_loss: 0.0252
Epoch 12/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 74ms/step - accuracy: 0.9858 - loss: 0.0459
- val_accuracy: 0.9948 - val_loss: 0.0197
Epoch 13/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 61s 81ms/step - accuracy: 0.9864 - loss: 0.0447
- val_accuracy: 0.9920 - val_loss: 0.0264
Epoch 14/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 56s 74ms/step - accuracy: 0.9880 - loss: 0.0413
- val_accuracy: 0.9921 - val_loss: 0.0292
Epoch 15/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 74ms/step - accuracy: 0.9865 - loss: 0.0423
- val_accuracy: 0.9908 - val_loss: 0.0322
Epoch 16/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 57s 76ms/step - accuracy: 0.9875 - loss: 0.0400
- val_accuracy: 0.9923 - val_loss: 0.0261
Epoch 17/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 73ms/step - accuracy: 0.9894 - loss: 0.0360
- val_accuracy: 0.9926 - val_loss: 0.0284
Epoch 18/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 57s 75ms/step - accuracy: 0.9886 - loss: 0.0334
- val_accuracy: 0.9934 - val_loss: 0.0240
Epoch 19/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 55s 73ms/step - accuracy: 0.9899 - loss: 0.0334
- val_accuracy: 0.9935 - val_loss: 0.0202
Epoch 20/20
750/750 ━━━━━━━━━━━━━━━━━━━━ 56s 74ms/step - accuracy: 0.9886 - loss: 0.0353
- val_accuracy: 0.9933 - val_loss: 0.0226
```

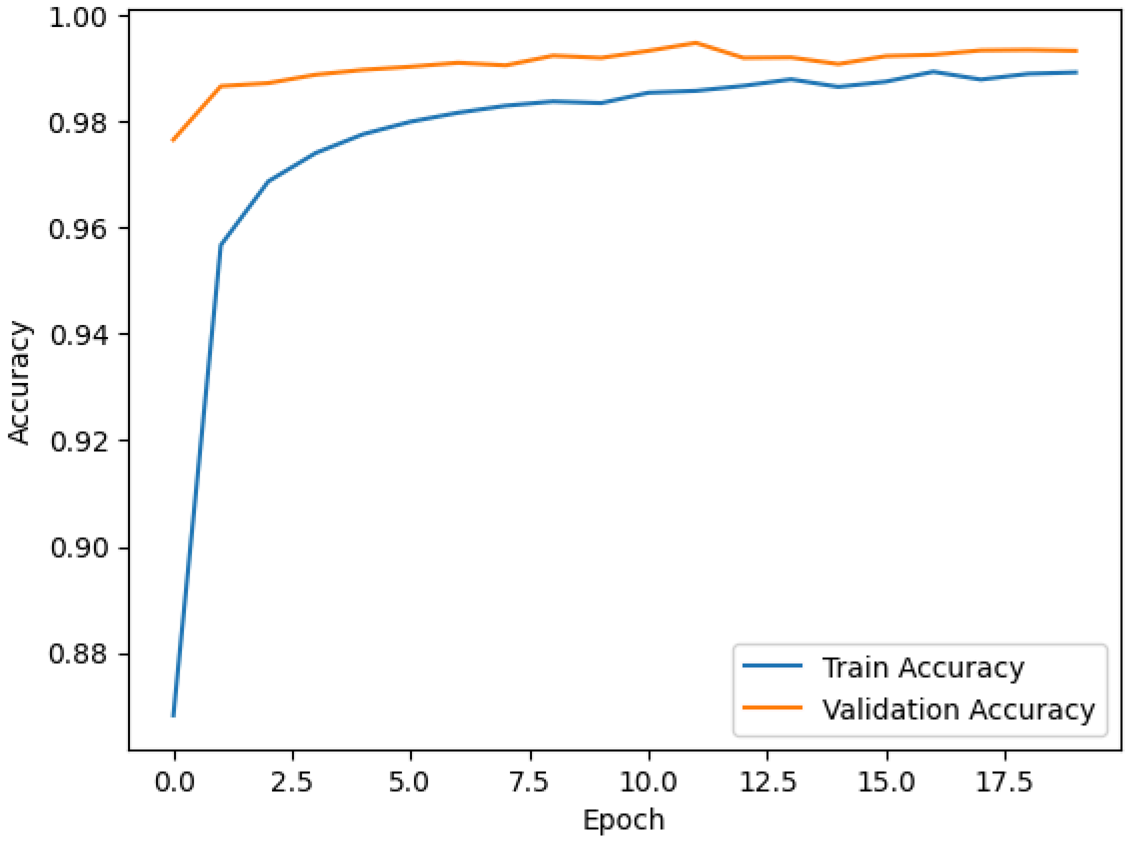
In [9]:

```
# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test Accuracy: {test_acc}")
print(f"Test Loss: {test_loss}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - accuracy: 0.9928 - loss: 0.0196
Test Accuracy: 0.9937999844551086
Test Loss: 0.01752481609582901
```

In [11]:

```
# Plot training history
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [ ]: