```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
```

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()  # Replace with cifar10 for CIFAR-10
# x_train, x_test: pixel values; y_train, y_test: labels
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),  # Flatten for MNIST, (32,32,3) for CIFAR-10
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(**kwargs)
```

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

```
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))
```

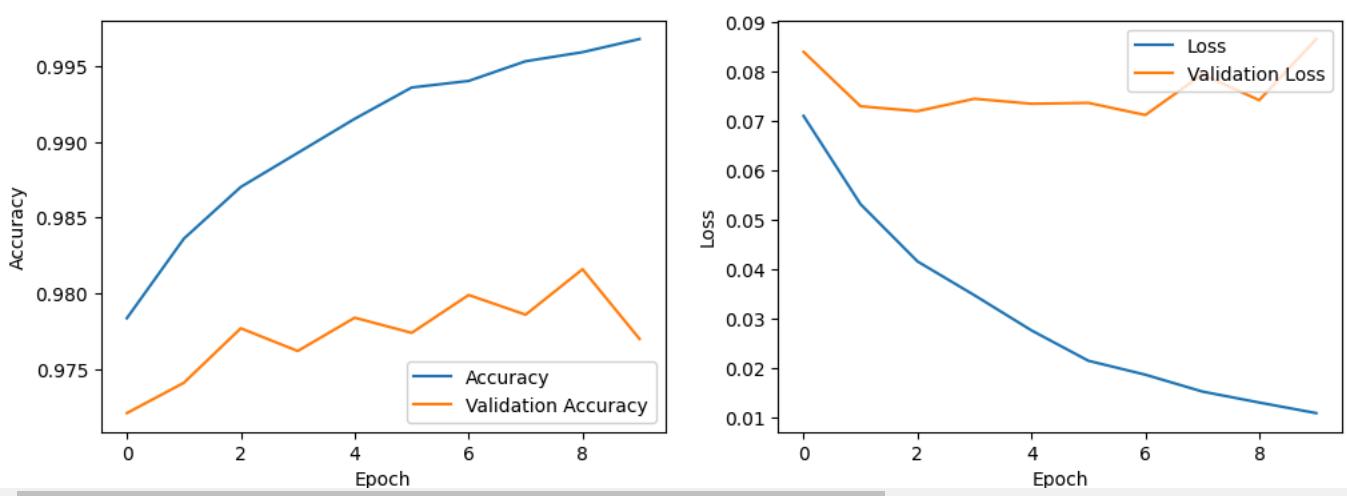
```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9781 - loss: 0.0716 - val_accuracy: 0.9721 - val_loss: 0.0840
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9851 - loss: 0.0504 - val_accuracy: 0.9741 - val_loss: 0.0730
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9881 - loss: 0.0388 - val_accuracy: 0.9777 - val_loss: 0.0720
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9899 - loss: 0.0325 - val_accuracy: 0.9762 - val_loss: 0.0745
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9927 - loss: 0.0238 - val_accuracy: 0.9784 - val_loss: 0.0735
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 4ms/step - accuracy: 0.9944 - loss: 0.0198 - val_accuracy: 0.9774 - val_loss: 0.0737
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 5ms/step - accuracy: 0.9949 - loss: 0.0166 - val_accuracy: 0.9799 - val_loss: 0.0712
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9961 - loss: 0.0139 - val_accuracy: 0.9786 - val_loss: 0.0794
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9963 - loss: 0.0117 - val_accuracy: 0.9816 - val_loss: 0.0742
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9979 - loss: 0.0083 - val_accuracy: 0.9770 - val_loss: 0.0866
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc}")
```

```
313/313 - 0s - 1ms/step - accuracy: 0.9770 - loss: 0.0866
Test accuracy: 0.9769999980926514
```

```
def plot_history(history):
    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Accuracy')
    plt.plot(history.history['val_accuracy'], label = 'Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(loc='lower right')

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Loss')
    plt.plot(history.history['val_loss'], label = 'Validation Loss')
    plt.xlabel('Epoch')
```

```
    plt.ylabel('Loss')
    plt.legend(loc='upper right')

    plt.show()

plot_history(history)
```




```
print("It is done by chetan bamble")
```

```
It is done by chetan bamble
```

Generate | 10 random numbers using numpy | Close

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt


# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 11s 0us/step
```

Generate | a slider using jupyter widgets | Close

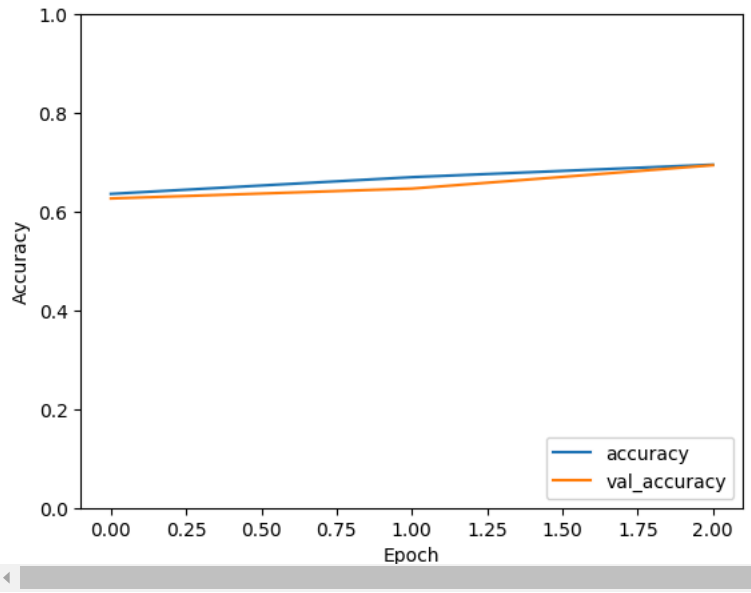
```
# Normalize the pixel values to be between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0


# Define the CNN model
model = models.Sequential()



# Add convolutional layers, followed by pooling layers
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
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(64, (3, 3), activation='relu'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Generate | create a dataframe with 2 columns and 10 rows | Close

```
 #Flatten the layer and add dense layers (fully connected layers)
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))  # CIFAR-10 has 10 classes


# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, epochs=3,
                    validation_data=(x_test, y_test))

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test accuracy: {test_acc}')

# Plot accuracy and loss curves
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

```
Epoch 1/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 86s 54ms/step - accuracy: 0.6297 - loss: 1.0490 - val_accuracy: 0.6267 - val_loss: 1.0498
Epoch 2/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 134s 49ms/step - accuracy: 0.6700 - loss: 0.9437 - val_accuracy: 0.6467 - val_loss: 1.0287
Epoch 3/3
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 74s 47ms/step - accuracy: 0.6937 - loss: 0.8726 - val_accuracy: 0.6941 - val_loss: 0.8856
313/313 - 4s - 12ms/step - accuracy: 0.6941 - loss: 0.8856
Test accuracy: 0.694100022315979
```




```
print("BY CHETAN GANESH BAMBLE")
```

```
BY CHETAN GANESH BAMBLE
```

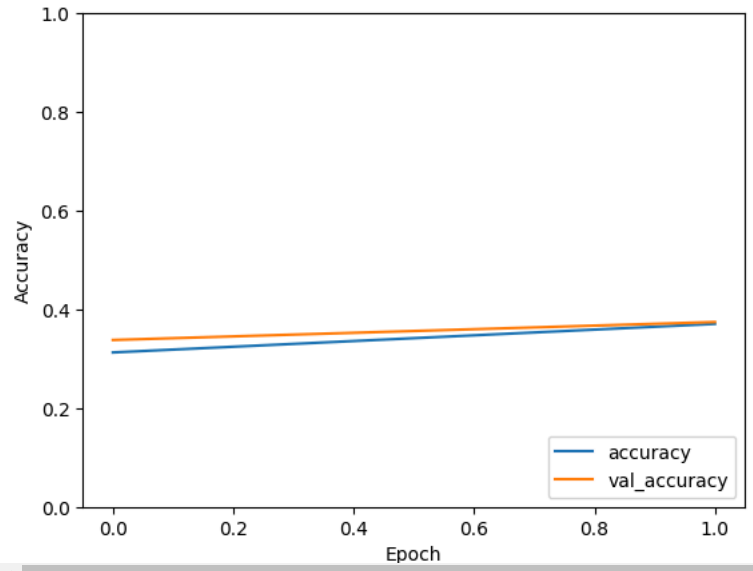
Start coding or generate with AI.

Start coding or generate with AI.

```
# Define a simple ANN model
ann_model = models.Sequential()

# Flatten the input (32x32x3) to a 1D vector
ann_model.add(layers.Flatten(input_shape=(32, 32, 3)))

# Add fully connected (dense) layers
```

```
ann_model.add(layers.Dense(128, activation='relu'))
ann_model.add(layers.Dense(10))  # CIFAR-10 has 10 output classes

# Compile the model
ann_model.compile(optimizer='adam',
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

# Train the model
ann_history = ann_model.fit(x_train, y_train, epochs=2,
                            validation_data=(x_test, y_test))

# Evaluate the model
ann_test_loss, ann_test_acc = ann_model.evaluate(x_test, y_test, verbose=2)
print(f'ANN Test accuracy: {ann_test_acc}')

# Plot accuracy and loss curves for ANN
plt.plot(ann_history.history['accuracy'], label='accuracy')
plt.plot(ann_history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```

```
Epoch 1/2
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.2733 - loss: 2.0356 - val_accuracy: 0.3380 - val_loss: 1.8502
Epoch 2/2
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.3618 - loss: 1.7854 - val_accuracy: 0.3744 - val_loss: 1.7303
313/313 - 1s - 3ms/step - accuracy: 0.3744 - loss: 1.7303
ANN Test accuracy: 0.37439998984336853
```



Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.