NAME: Kriyanshi Ghodasara

PRN:23070243017

TOPIC: DL Assignment on image classification

```
import requests
train_url = 'http://ufldl.stanford.edu/housenumbers/train_32x32.mat'
test_url = 'http://ufldl.stanford.edu/housenumbers/test_32x32.mat'

def download_file(url, filename):
    print(f"Downloading {filename}...")
    response = requests.get(url, stream=True)
    with open(filename, 'wb') as f:
        for chunk in response.iter_content(chunk_size=1024):
            if chunk:
                f.write(chunk)
    print(f"{filename} downloaded.")

download_file(train_url, 'train_32x32.mat')
download_file(test_url, 'test_32x32.mat')
```

```
Downloading train_32x32.mat...
train_32x32.mat downloaded.
Downloading test_32x32.mat...
test_32x32.mat downloaded.
```

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

train_data = sio.loadmat('train_32x32.mat')
test_data = sio.loadmat('test_32x32.mat')

X_train = np.array(train_data['X'], dtype=np.float32)
y_train = np.array(train_data['y'], dtype=np.int32).flatten()
X_test = np.array(test_data['X'], dtype=np.float32)
y_test = np.array(test_data['y'], dtype=np.int32).flatten()

X_train = np.transpose(X_train, (3, 0, 1, 2))
X_test = np.transpose(X_test, (3, 0, 1, 2))

y_train[y_train == 10] = 0
y_test[y_test == 10] = 0

X_train /= 255.0
X_test /= 255.0



model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(X_train, y_train, epochs=10, batch_size=64,
                    validation_data=(X_test, y_test))
```

```
Epoch 1/10
1145/1145 [==============================] - 121s 105ms/step - loss: 0.9156 - accuracy: 0.7079 - val_loss: 0.5349 - val_accuracy: 0.8509
Epoch 2/10
1145/1145 [==============================] - 118s 103ms/step - loss: 0.4374 - accuracy: 0.8734 - val_loss: 0.4521 - val_accuracy: 0.8741
Epoch 3/10
1145/1145 [==============================] - 124s 108ms/step - loss: 0.3630 - accuracy: 0.8947 - val_loss: 0.3977 - val_accuracy: 0.8874
Epoch 4/10
1145/1145 [==============================] - 117s 102ms/step - loss: 0.3205 - accuracy: 0.9067 - val_loss: 0.3847 - val_accuracy: 0.8923
Epoch 5/10
1145/1145 [==============================] - 117s 102ms/step - loss: 0.2898 - accuracy: 0.9163 - val_loss: 0.3689 - val_accuracy: 0.8966
Epoch 6/10
1145/1145 [==============================] - 116s 102ms/step - loss: 0.2658 - accuracy: 0.9223 - val_loss: 0.3734 - val_accuracy: 0.8980
Epoch 7/10
1145/1145 [==============================] - 117s 102ms/step - loss: 0.2464 - accuracy: 0.9281 - val_loss: 0.3558 - val_accuracy: 0.9008
Epoch 8/10
1145/1145 [==============================] - 120s 105ms/step - loss: 0.2269 - accuracy: 0.9332 - val_loss: 0.3584 - val_accuracy: 0.9044
Epoch 9/10
1145/1145 [==============================] - 118s 103ms/step - loss: 0.2104 - accuracy: 0.9375 - val_loss: 0.3961 - val_accuracy: 0.8937
```

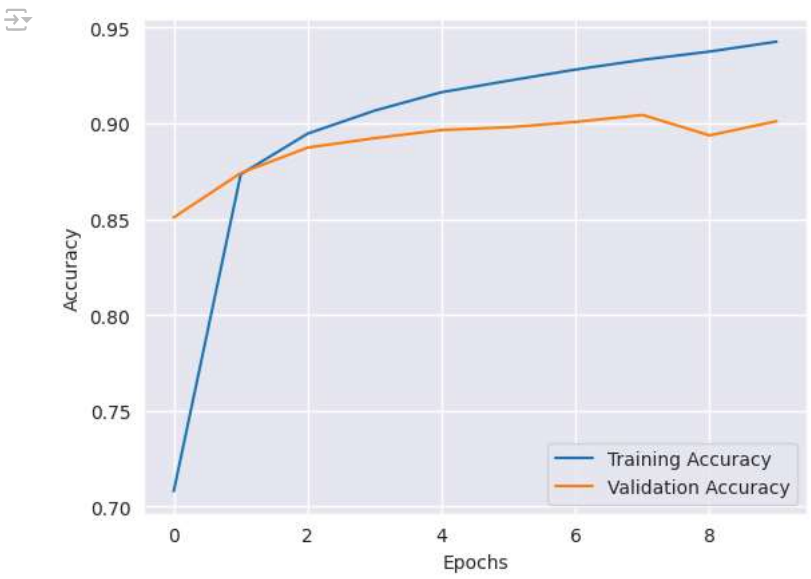
```
Epoch 10/10
1145/1145 [==============================] - 119s 104ms/step - loss: 0.1928 - accuracy: 0.9427 - val_loss: 0.3968 - val_accuracy: 0.9012
```

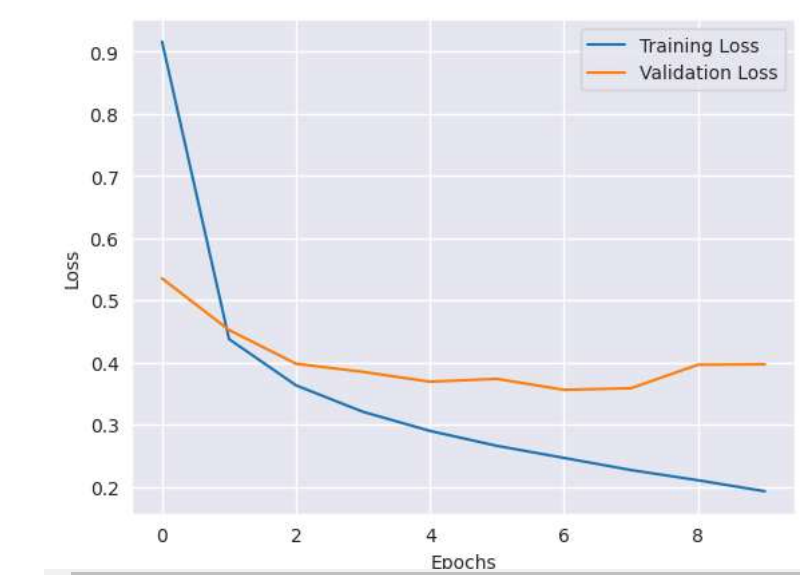
```
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_acc:.4f}")
```

```
814/814 [==============================] - 16s 20ms/step - loss: 0.3968 - accuracy: 0.9012
Test accuracy: 0.9012
```

```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```

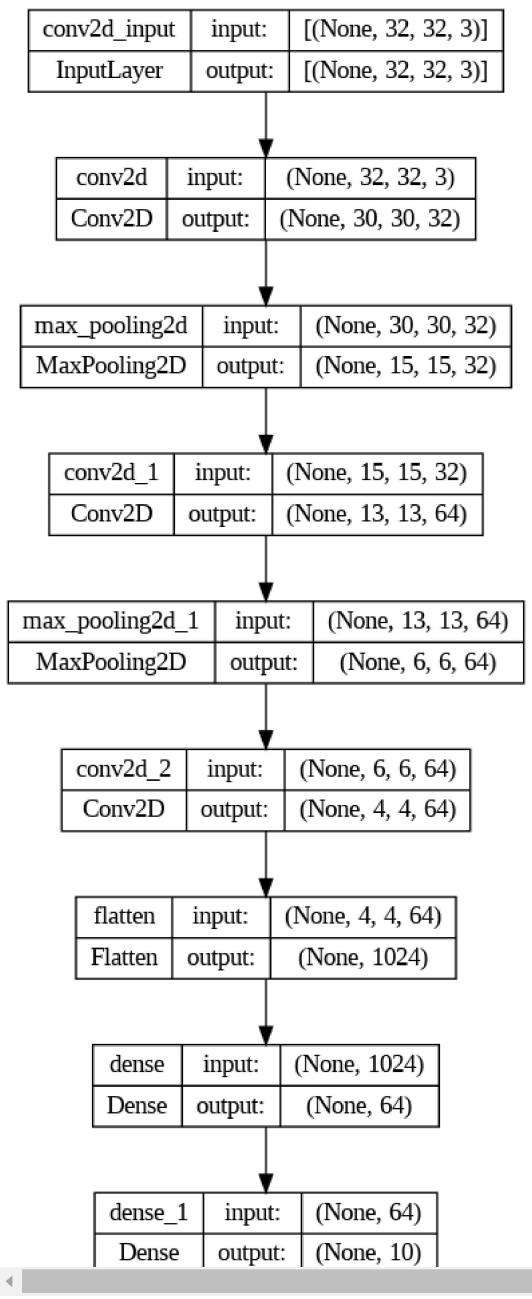





```
from tensorflow.keras.utils import plot_model

plot_model(model, to_file='model_architecture.png', show_shapes=True, show_layer_names=True)

from IPython.display import Image
Image('model_architecture.png')
```



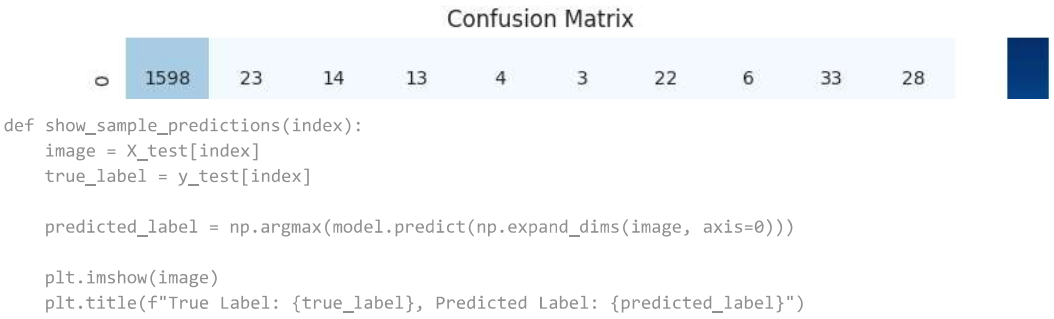

```
from sklearn.metrics import confusion_matrix, classification_report

y_pred = np.argmax(model.predict(X_test), axis=1)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=np.arange(10), yticklabels=np.arange(10))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

print("Classification Report:\n")
print(classification_report(y_test, y_pred, digits=4))
```

```
814/814 [==============================] - 13s 16ms/step
```



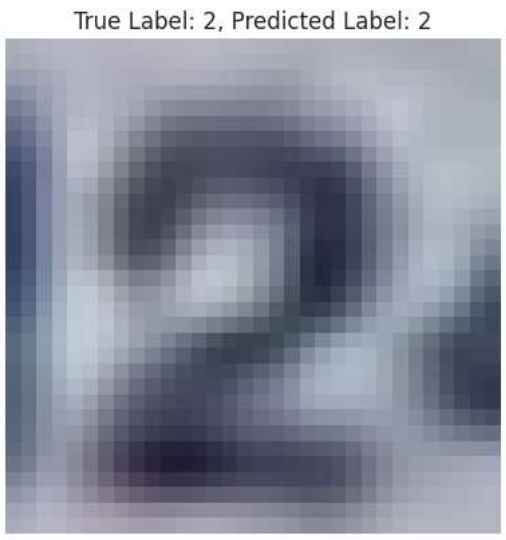

```
def show_sample_predictions(index):
    image = X_test[index]
    true_label = y_test[index]

    predicted_label = np.argmax(model.predict(np.expand_dims(image, axis=0)))

    plt.imshow(image)
    plt.title(f"True Label: {true_label}, Predicted Label: {predicted_label}")
    plt.axis('off')
    plt.show()

sample_index = np.random.randint(len(X_test))
show_sample_predictions(sample_index)
```

```
1/1 [==============================] - 0s 77ms/step
```




```
              2     0.9352    0.9294    0.9323      4149
```

SAVE THE MODEL AND EVALUATE BY USER INPUT

```
              5     0.9380    0.8721    0.9041      2384

model.save('svhn_cnn_model.h5')
              8     0.7691    0.8910    0.8256      1660

loaded_model = tf.keras.models.load_model('svhn_cnn_model.h5')

sample_image = X_test[0]
predicted_label = np.argmax(loaded_model.predict(np.expand_dims(sample_image, axis=0)))
print(f'Predicted Label: {predicted_label}, True Label: {y_test[0]}')
```

```
1/1 [==============================] - 0s 237ms/step
Predicted Label: 5, True Label: 5
```

```
import numpy as np
import matplotlib.pyplot as plt

def get_user_input(max_index):
    while True:
        try:
            user_input = int(input(f"Enter an image index (0 to {max_index - 1}): "))
            if 0 <= user_input < max_index:
                return user_input
            else:
                print(f"Please enter a number between 0 and {max_index - 1}.")
        except ValueError:
            print("Invalid input. Please enter a valid integer.")

max_index = len(X_test)
sample_index = get_user_input(max_index)

sample_image = X_test[sample_index]
true_label = y_test[sample_index]

predicted_label = np.argmax(model.predict(np.expand_dims(sample_image, axis=0)))

plt.imshow(sample_image)
plt.title(f"True Label: {true_label}, Predicted Label: {predicted_label}")
plt.axis('off')
plt.show()

print(f"True Label: {true_label}")
print(f"Predicted Label: {predicted_label}")
```

```
Enter an image index (0 to 26031): 4655
1/1 [==============================] - 0s 33ms/step
```