Batch: HO-DL-1

Roll Number: 16010421073 **Experiment Number: 6**

Name: Keyur Patel

Title of the Experiment: Deep neural network for computer vision

Program and Output:

importing requisite libraries

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import classification_report

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import regularizers
```

Load and preprocess CIFAR-10 dataset

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train = X_train.astype('float32') / 255.0

X_test = X_test.astype('float32') / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
```

Define CNN model architecture with regularization

```
model = Sequential([

    Conv2D(32, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001),
```

```
            input_shape=(32, 32, 3)),

    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu', kernel_regularizer=regularizers.l2(0.001)),

    MaxPooling2D((2, 2)),

    Flatten(),

    Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.001)),

    Dropout(0.5),

    Dense(10, activation='softmax')

])

model.compile(optimizer=Adam(learning_rate=0.001),loss='sparse_categorical_crossentropy'
, metrics=['accuracy'])
```

Train the model

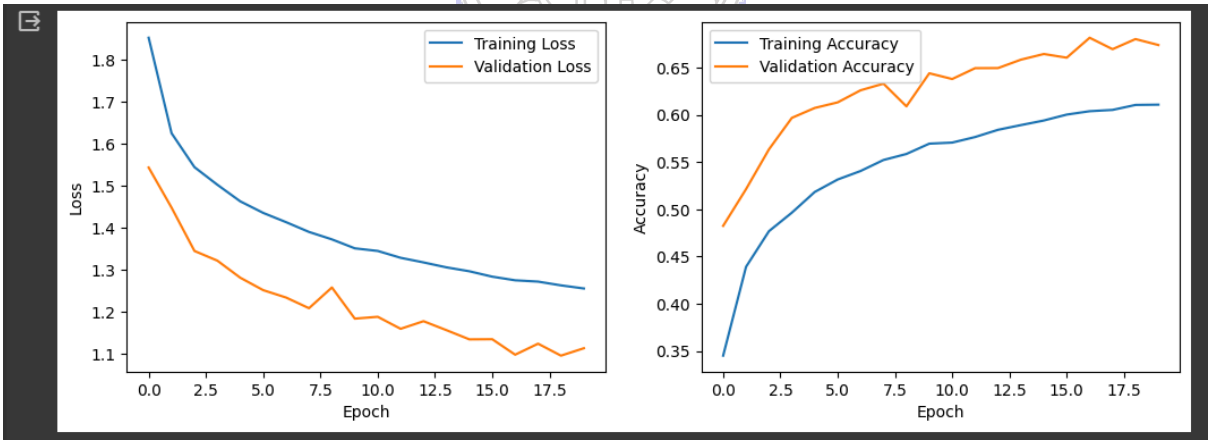
```
history = model.fit(X_train, y_train, epochs=20, batch_size=64, validation_data=(X_test,

y_test), verbose=1)
```

```
21m
Epoch 1/20
782/782 [==============================] - 63s 79ms/step - loss: 1.8519 - accuracy: 0.3451 - val_loss: 1.5430 - val_accuracy: 0.4825
Epoch 2/20
782/782 [==============================] - 63s 81ms/step - loss: 1.6247 - accuracy: 0.4392 - val_loss: 1.4475 - val_accuracy: 0.5214
Epoch 3/20
782/782 [==============================] - 62s 79ms/step - loss: 1.5438 - accuracy: 0.4769 - val_loss: 1.3443 - val_accuracy: 0.5635
Epoch 4/20
782/782 [==============================] - 62s 79ms/step - loss: 1.5019 - accuracy: 0.4965 - val_loss: 1.3212 - val_accuracy: 0.5968
Epoch 5/20
782/782 [==============================] - 61s 78ms/step - loss: 1.4625 - accuracy: 0.5184 - val_loss: 1.2806 - val_accuracy: 0.6073
Epoch 6/20
782/782 [==============================] - 61s 78ms/step - loss: 1.4349 - accuracy: 0.5316 - val_loss: 1.2511 - val_accuracy: 0.6132
Epoch 7/20
782/782 [==============================] - 63s 81ms/step - loss: 1.4129 - accuracy: 0.5406 - val_loss: 1.2334 - val_accuracy: 0.6260
Epoch 8/20
782/782 [==============================] - 64s 82ms/step - loss: 1.3896 - accuracy: 0.5522 - val_loss: 1.2080 - val_accuracy: 0.6330
Epoch 9/20
782/782 [==============================] - 63s 81ms/step - loss: 1.3719 - accuracy: 0.5586 - val_loss: 1.2574 - val_accuracy: 0.6090
Epoch 10/20
782/782 [==============================] - 64s 82ms/step - loss: 1.3506 - accuracy: 0.5695 - val_loss: 1.1832 - val_accuracy: 0.6440
Epoch 11/20
782/782 [==============================] - 63s 81ms/step - loss: 1.3444 - accuracy: 0.5706 - val_loss: 1.1876 - val_accuracy: 0.6379
Epoch 12/20
782/782 [==============================] - 63s 80ms/step - loss: 1.3281 - accuracy: 0.5765 - val_loss: 1.1590 - val_accuracy: 0.6494
Epoch 13/20
782/782 [==============================] - 61s 78ms/step - loss: 1.3171 - accuracy: 0.5842 - val_loss: 1.1771 - val_accuracy: 0.6495
Epoch 14/20
782/782 [==============================] - 61s 78ms/step - loss: 1.3053 - accuracy: 0.5892 - val_loss: 1.1558 - val_accuracy: 0.6584
Epoch 15/20
782/782 [==============================] - 64s 82ms/step - loss: 1.2960 - accuracy: 0.5940 - val_loss: 1.1339 - val_accuracy: 0.6644
Epoch 16/20
782/782 [==============================] - 62s 80ms/step - loss: 1.2831 - accuracy: 0.6003 - val_loss: 1.1342 - val_accuracy: 0.6605
Epoch 17/20
782/782 [==============================] - 61s 78ms/step - loss: 1.2745 - accuracy: 0.6038 - val_loss: 1.0974 - val_accuracy: 0.6816
Epoch 18/20
782/782 [==============================] - 64s 82ms/step - loss: 1.2714 - accuracy: 0.6052 - val_loss: 1.1236 - val_accuracy: 0.6695
Epoch 19/20
782/782 [==============================] - 63s 81ms/step - loss: 1.2624 - accuracy: 0.6105 - val_loss: 1.0951 - val_accuracy: 0.6802
Epoch 20/20
782/782 [==============================] - 63s 81ms/step - loss: 1.2550 - accuracy: 0.6108 - val_loss: 1.1128 - val_accuracy: 0.6739
```

Plot loss and accuracy

```
plt.figure(figsize=(12, 4))
```

```
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.legend()

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.legend()

plt.show()
```



Display classification report

```
y_pred = np.argmax(model.predict(X_test), axis=-1)

print(classification_report(y_test, y_pred))
```

```
313/313 [==============================] - 4s 12ms/step
```

	precision	recall	f1-score	support
0	0.72	0.74	0.73	1000
1	0.78	0.82	0.80	1000
2	0.60	0.47	0.53	1000
3	0.46	0.43	0.45	1000
4	0.67	0.57	0.61	1000
5	0.48	0.72	0.58	1000
6	0.72	0.80	0.76	1000
7	0.76	0.70	0.73	1000
8	0.88	0.68	0.77	1000
9	0.75	0.80	0.78	1000
accuracy			0.67	10000
macro avg	0.68	0.67	0.67	10000
weighted avg	0.68	0.67	0.67	10000

Comment:

The Neural Network model outperforms the Ridge Regression model, as demonstrated by its lower Mean Squared Error (MSE) and superior R-squared metrics. This suggests that the Neural Network is better at identifying complex patterns within the data, resulting in more accurate predictions. The choice of which model to use depends on factors such as the complexity of the dataset and the computational resources available. To gain a deeper understanding, it is essential to further explore the architecture of the model and examine the dataset for potential biases.

CO3: Assimilate fundamentals of Convolutional Neural Network

Conclusion: In this experiment, we developed a deep neural network (DNN) for computer vision has demonstrated the profound capabilities of DNNs in analyzing and interpreting complex visual data with remarkable accuracy.

K. J. SOMAIYA COLLEGE OF ENGG.