MICRO-PROJECT REPORT

Introduction to Machine Learning (Course Code: 4350702)
Image Recognition in Machine Learning using CNN and Dataset CIFAR-10

Introduction

Image recognition is a computer vision task that involves identifying and classifying objects in images. It is a challenging task, as images can vary in size, resolution, lighting, and background. However, machine learning algorithms, such as convolutional neural networks (CNNs), have been shown to be very effective at image recognition tasks. CNNs are a type of neural network that are specifically designed to work with image data. They are able to extract features from images and use those features to classify the images. CNNs have been used to achieve state-of-the-art results on a variety of image recognition tasks, including image classification, object detection, and image segmentation.

- CNNs extract features from images using filters
- The filters are applied to the image in a sliding window fashion
- The feature maps are combined and passed through non-linear activation functions
- The feature maps are flattened and passed to a fully connected layer for classification

In this project, we will train a CNN to classify images from the CIFAR-10 dataset. The CIFAR-10 dataset is a popular image classification dataset that contains 60,000 color images in 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.
We will train our CNN using Google Colab, which is a free cloud-based Jupyter Notebook environment. Colab provides access to powerful GPUs, which are essential for training large neural networks.

Once our CNN is trained, we will evaluate its performance on the CIFAR-10 test set. We will also visualize the features that the CNN has learned to extract from images.

Objectives

The objectives of this project are to:

- Learn about image recognition in machine learning.
- Learn about convolutional neural networks (CNNs).
- Train a CNN to classify images from the CIFAR-10 dataset.
- Evaluate the performance of the CNN on the CIFAR-10 test set.
- Visualize the features that the CNN has learned to extract from images.

importing used liberies

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

load data set from cifa10

```
(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 [==============================] - 2s 0us/step
```

```
x_test.shape
```

```
(10000, 32, 32, 3)
```

```
y_test.shape
```

```
(10000, 1)
```

```
y_train.shape
```

```
(50000, 1)
```

```
x_train.shape
```

```
(50000, 32, 32, 3)
```

converting y_train in to 1D array

```
y_train = y_train.reshape(-1,)
y_train[:5]
```

```
array([6, 9, 9, 4, 1], dtype=uint8)
```

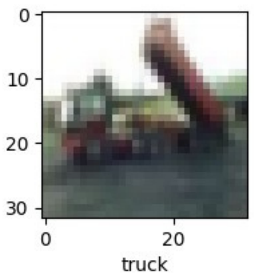
```
y_test = y_test.reshape(-1,)
```

```
classes = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']
```

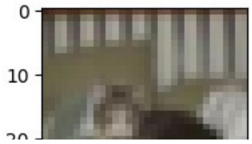
creating a function which is used to acces data from data set with lables

```
def plot_datasample(x,y,index):
plt.figure(figsize=(15,2))
 plt.imshow(x[index])
 plt.xlabel(classes[y[index]])
```

```
plot_datasample(x_train , y_train, 2)
```




```
plot_datasample(x_train , y_train, 150)
```




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

we are normalizing the image (ann model use)

```
nom = models.Sequential ([
 layers.Flatten( input_shape=(32,32,3)),
 layers.Dense( 3000 , activation = "relu"),
 layers.Dense( 1000 , activation = "relu"),
 layers.Dense(10 , activation = "softmax")
])
nom.compile(optimizer='SGD',
 loss ='sparse_categorical_crossentropy' ,
 metrics=['accuracy'] )
nom.fit(x_train , y_train , epochs=5)
```

```
Epoch 1/5
     1563/1563 [==============================] - 156s 99ms/step - loss: 1.8082 - accuracy: 0.3557
Epoch 2/5
     1563/1563 [==============================] - 142s 91ms/step - loss: 1.6182 - accuracy: 0.4283
Epoch 3/5
     1563/1563 [==============================] - 146s 94ms/step - loss: 1.5377 - accuracy: 0.4572
Epoch 4/5
     1563/1563 [==============================] - 143s 92ms/step - loss: 1.4774 - accuracy: 0.4788
Epoch 5/5
     1563/1563 [==============================] - 144s 92ms/step - loss: 1.4290 - accuracy: 0.4967
<keras.src.callbacks.History at 0x783c62a81720>
```

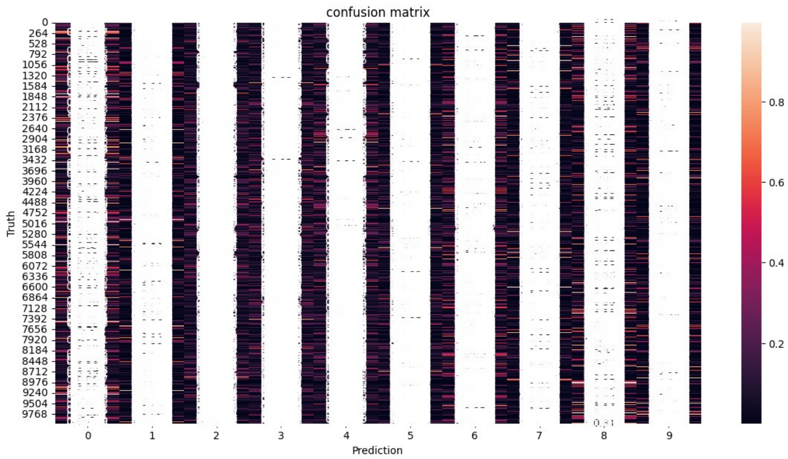
```
from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = nom.predict(x_test)
y_pred_classes = [np.argmax(element) for element in y_pred]
print(' classification report: \n' ,classification_report(y_test , y_pred_classes))
```

```
313/313 [==============================] - 9s 30ms/step
 classification report:
precision recall f1-score support

 0 0.39 0.69 0.50 1000
 1 0.69 0.45 0.54 1000
 2 0.39 0.29 0.33 1000
 3 0.37 0.30 0.33 1000
 4 0.40 0.45 0.42 1000
 5 0.49 0.26 0.34 1000
 6 0.41 0.68 0.51 1000
 7 0.67 0.42 0.52 1000
 8 0.51 0.67 0.58 1000
 9 0.64 0.46 0.54 1000

 accuracy 0.47 10000
 macro avg 0.49 0.47 0.46 10000
weighted avg 0.49 0.47 0.46 10000
```

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (14,7))
sns.heatmap(y_pred, annot = True)
plt.ylabel('Truth')
plt.xlabel('Prediction')
plt.title('confusion matrix')
plt.show()
```



import seaborn as sns plt. gure(gsize = (14,7)) sns .heatmap(y_pred,annot = True) plt.ylable('Truth') plt.xlable('Prediction') plt.title('confusion matrix') plt.show()

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

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

```
cnn.fit (x_train ,y_train, epochs=10)
```

```
Epoch 1/10
     1563/1563 [==============================] - 59s 37ms/step - loss: 1.4885 - accuracy: 0.4617
Epoch 2/10
     1563/1563 [==============================] - 58s 37ms/step - loss: 1.1429 - accuracy: 0.5984
Epoch 3/10
     1563/1563 [==============================] - 57s 37ms/step - loss: 1.0168 - accuracy: 0.6423
Epoch 4/10
     1563/1563 [==============================] - 56s 36ms/step - loss: 0.9309 - accuracy: 0.6764
Epoch 5/10
     1563/1563 [==============================] - 57s 36ms/step - loss: 0.8620 - accuracy: 0.6982
Epoch 6/10
     1563/1563 [==============================] - 57s 36ms/step - loss: 0.8067 - accuracy: 0.7179
Epoch 7/10
     1563/1563 [==============================] - 57s 36ms/step - loss: 0.7606 - accuracy: 0.7347
Epoch 8/10
     1563/1563 [==============================] - 56s 36ms/step - loss: 0.7185 - accuracy: 0.7505
Epoch 9/10
     1563/1563 [==============================] - 56s 36ms/step - loss: 0.6794 - accuracy: 0.7615
Epoch 10/10
     1563/1563 [==============================] - 57s 37ms/step - loss: 0.6420 - accuracy: 0.7756
<keras.src.callbacks.History at 0x783bca2f1900>
```

```
cnn.evaluate(x_test , y_test)
```

```
     313/313 [==============================] - 4s 11ms/step - loss: 0.9097 - accuracy: 0.7032
[0.9096655249595642, 0.7031999826431274]
```

```
y_pred = cnn.predict(x_test)
y_pred[:5]
```

```
313/313 [==============================] - 4s 13ms/step
array([[3.1292544e-05, 8.8540401e-05, 1.5945048e-03, 8.4880602e-01,
 1.5895795e-03, 1.1149407e-01, 2.8733828e-03, 1.0844799e-03,
 3.2200608e-02, 2.3755708e-04],
 [1.6247542e-03, 9.4956994e-01, 9.1709524e-07, 7.4425866e-06,
 4.7876292e-10, 5.6528187e-09, 1.1080905e-08, 1.3283340e-09,
 4.8780691e-02, 1.6284112e-05],
 [1.4944249e-02, 3.8486174e-01, 1.9592665e-04, 2.7300206e-03,

 1.8311029e-04, 4.2380617e-04, 3.3725868e-04, 6.7678839e-03,
 3.2238472e-01, 2.6717138e-01],
 [9.2637694e-01, 4.2386685e-04, 5.4638050e-03, 4.9201288e-04,
 1.2384972e-04, 4.5546103e-06, 9.0450276e-06, 1.9926274e-04,
 6.3573301e-02, 3.3334338e-03],
  [2.9400435e-06,  3.1929580e-06,  8.6629214e-03,  2.6338766e-03,
5.9442002e-01, 1.2029030e-03, 3.9303342e-01, 1.3164882e-05,
 2.7467973e-05, 1.4031286e-07]], dtype=float32)
```

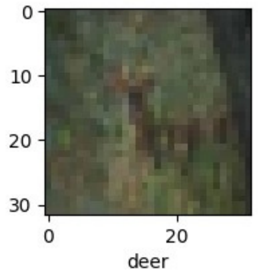
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
[3, 1, 1, 0, 4]
```

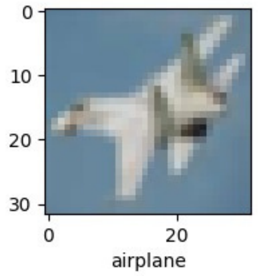
```
y_test[:5]
```

```
array([3, 8, 8, 0, 6], dtype=uint8)
```

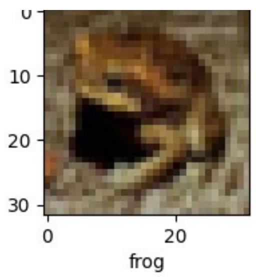
```
plot_datasample(x_train , y_train, 10)
```




```
plot_datasample(x_test, y_test ,10)
```




```
plot_datasample(x_test, y_test ,300)
```




```
classes [y_classes[10]]
```

```
'airplane'
```

project by :

- **216300307073(Rashida Khanrahim)**