Classification with CIFAR-10 Dataset

In [ ]:

```
# Recommending to change runtime type to GPU for performance
```

In [ ]:

```
from tensorflow import keras
```

In [ ]:

```
#import data
(train_images,train_labels),(test_images,test_labels) = keras.datasets.cifar10.load_data
()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 6s 0us/step
170508288/170498071 [==============================] - 6s 0us/step
```

In [ ]:

```
#scaling
train_images, test_images = train_images / 255, test_images / 255
```

In [ ]:

```
from keras import layers, models, losses
```

In [ ]:

```
#Create model
model = models.Sequential()
```

In [ ]:

```
#Create CNN Layers
model.add(layers.Conv2D(32,(3,3),activation="relu",input_shape=(32,32,3)))
model.add(layers.MaxPooling2D())
model.add(layers.Conv2D(64,(3,3),activation="relu"))
model.add(layers.MaxPooling2D())
model.add(layers.Conv2D(64,(3,3),activation="relu"))
```

In [ ]:

```
#Create Dense Layers
model.add(layers.Flatten())
model.add(layers.Dense(64,activation="relu"))
model.add(layers.Dense(10))
```

In [ ]:

```
#Compile Model
model.compile(optimizer="adam",
              loss=losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=["accuracy"])
```

In [ ]:

```
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```
# To increase our input data we use Image Augmentation
datagen = ImageDataGenerator(
```

```
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

In [ ]:

```
# Fit the train_images to datagen
datagen.fit(train_images)
```

In [ ]:

```
#Fit the model with datagen
model.fit(datagen.flow(train_images, train_labels),
          batch_size= 32, steps_per_epoch=len(train_images)/32,
          epochs=25, verbose=1)
```

```
Epoch 1/25
1562/1562 [==============================] - 70s 27ms/step - loss: 1.8407 - accuracy: 0.1
069
Epoch 2/25
1562/1562 [==============================] - 42s 27ms/step - loss: 1.6130 - accuracy: 0.0
972
Epoch 3/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.5113 - accuracy: 0.0
974
Epoch 4/25
1562/1562 [==============================] - 41s 27ms/step - loss: 1.4457 - accuracy: 0.0
967
Epoch 5/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.3913 - accuracy: 0.0
972
Epoch 6/25
1562/1562 [==============================] - 41s 27ms/step - loss: 1.3565 - accuracy: 0.0
966
Epoch 7/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.3294 - accuracy: 0.0
971
Epoch 8/25
1562/1562 [==============================] - 41s 27ms/step - loss: 1.3000 - accuracy: 0.0
993
Epoch 9/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.2718 - accuracy: 0.0
980
Epoch 10/25
1562/1562 [==============================] - 41s 27ms/step - loss: 1.2596 - accuracy: 0.0
993
Epoch 11/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.2386 - accuracy: 0.1
006
Epoch 12/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.2184 - accuracy: 0.0
991
Epoch 13/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.2102 - accuracy: 0.0
997
Epoch 14/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1923 - accuracy: 0.1
003
Epoch 15/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1866 - accuracy: 0.0
993
Epoch 16/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1733 - accuracy: 0.1
002
Epoch 17/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1568 - accuracy: 0.0
993
```

```
Epoch 18/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1553 - accuracy: 0.1
011
Epoch 19/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1399 - accuracy: 0.1
008
Epoch 20/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1348 - accuracy: 0.1
004
Epoch 21/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1270 - accuracy: 0.1
013
Epoch 22/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1175 - accuracy: 0.1
005
Epoch 23/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1134 - accuracy: 0.1
013
Epoch 24/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.1055 - accuracy: 0.1
011
Epoch 25/25
1562/1562 [==============================] - 41s 26ms/step - loss: 1.0948 - accuracy: 0.1
003
```

Out[ ]:

```
<keras.callbacks.History at 0x7f9e100d7e10>
```

In [ ]:

```
#evaluate the model and get loss with accuracy metric
loss = model.evaluate(datagen.flow(test_images, test_labels),batch_size=32)
```

```
313/313 [==============================] - 8s 25ms/step - loss: 1.0997 - accuracy: 0.1077
```

In [ ]:

```
prediction = model.predict(test_images)
```

In [ ]:

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

In [ ]:

```
class_names=["airplane","automobile","bird","cat",
             "deer","dog","frog","horse","ship","truck"]
```

In [ ]:

```
INDEX=""
while INDEX.isdigit() == False:
  INDEX = input("Lütfen tahmin gerçekleştirmek istediğiniz indeksi girin: ")
  if int(INDEX)>=len(test_images):
    INDEX=""

INDEX = int(INDEX)

predicted_value=class_names[np.argmax(prediction[INDEX])]
actual_value= class_names[test_labels[INDEX][0]]

print(f"Real Value: {actual_value} - Predicted Value: {predicted_value}")
plt.figure()
plt.imshow(test_images[INDEX])
```

```
Lütfen tahmin gerçekleştirmek istediğiniz indeksi girin: 5
Real Value: frog - Predicted Value: frog
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7f9d2327d10>
```