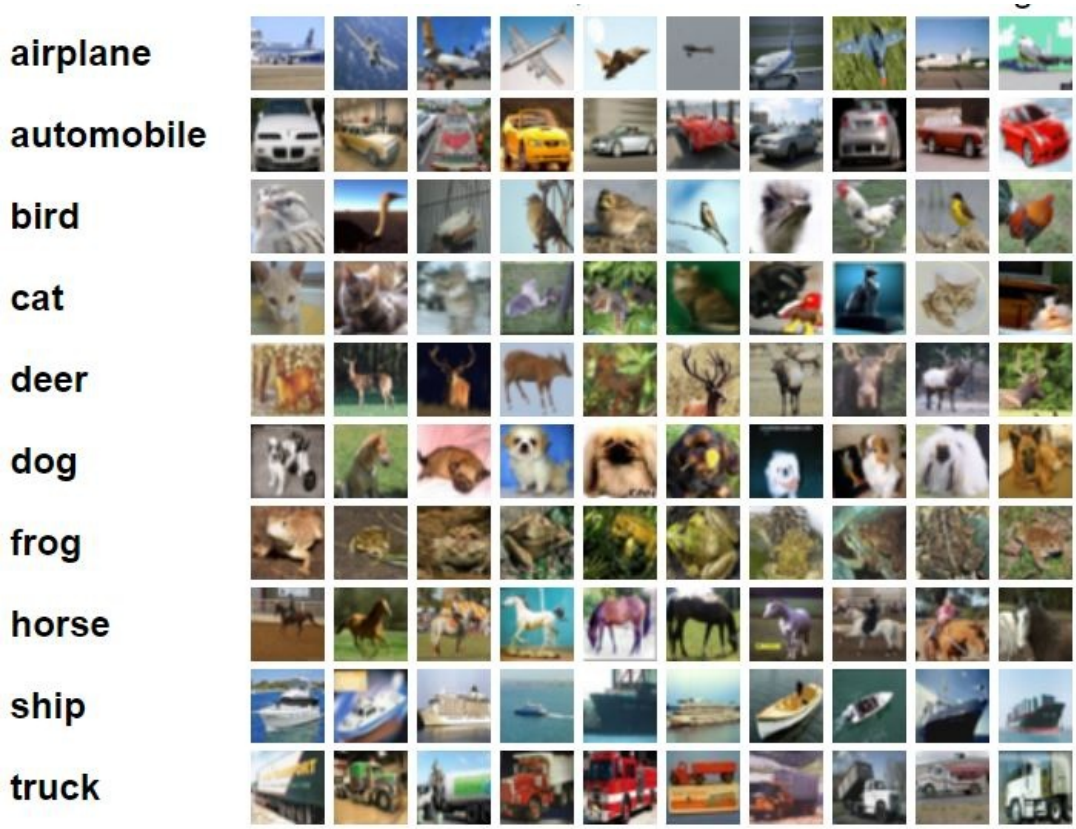
Small Image Classification Using Convolutional Neural Network (CNN)

In this notebook, we will classify small images cifar10 dataset from tensorflow keras datasets. There are total 10 classes as shown below. We will use CNN for classification



In [ ]:

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

In [ ]:

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

In [ ]:

```
X_test.shape
```

In [ ]:

```
y_train[:5]
```

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
plot_sample(X_train, y_train, 0)
```

In [ ]:

```
plot_sample(X_train, y_train, 1)
```

In [ ]:

```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

In [ ]:

```
nn = models.Sequential([
        layers.Flatten(input_shape=(32,32,3)),
        layers.Dense(3000, activation='relu'),
        layers.Dense(1000, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])

ann.compile(optimizer='SGD',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.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))
```

In [ ]:

```
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')
])
```

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
y_test[:5]
```

In [ ]:

```
plot_sample(X_test, y_test,3)
```

In [ ]:

```
classes[y_classes[3]]
```

In [ ]:

```
classes[y_classes[3]]
```