In [14]:

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

In [15]:

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

Out[15]:

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

50000 - Training Örneği
32x32 - Görsel Boyutları
3 - RGB

In [16]:

```
X_test.shape
```

Out[16]:

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

10000 - Test Örneği
32x32 - Görsel Boyutları
3 - RGB

In [17]:

```
y_train[:5]
```

Out[17]:

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

In [18]:

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

Out[18]:

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

In [19]:

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

In [20]:

```
classes[0]
```

Out[20]:

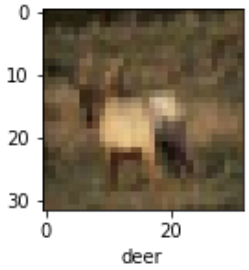
```
'airplane'
```

In [21]:

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

In [22]:

```
gorsel_goster(X_train, y_train, 3)
```



RGB kanalı 3 bölümden oluşur. R(kırmızı), G(yeşil) ve B(mavi).
Bu üç ayrı kanalın her biri 0 ile 255 arası bir değer alabilir ve böylece renkler oluşturulur.
Verisetimizdeki her bir gorselin değerlerini 255'e bölersek, 0 ile 1 arasında normalizasyon yapmış oluruz.

In [23]:

```
X_train = X_train / 255
X_test = X_test / 255
```

In [24]:

```
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.Conv2D(filters=128, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

In [27]:

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

In [28]:

```
history1 = cnn.fit(X_train, y_train, epochs=100, steps_per_epoch = 50, batch_size = 3)
```

```
Epoch 1/100
50/50 [==============================] - 0s 3ms/step - loss: 2.3162 - accuracy: 0.1200
Epoch 2/100
50/50 [==============================] - 0s 3ms/step - loss: 2.2983 - accuracy: 0.0933
Epoch 3/100
50/50 [==============================] - 0s 3ms/step - loss: 2.2778 - accuracy: 0.1600
Epoch 4/100
50/50 [==============================] - 0s 3ms/step - loss: 2.2436 - accuracy: 0.1800
Epoch 5/100
50/50 [==============================] - 0s 3ms/step - loss: 2.2336 - accuracy: 0.1200
Epoch 6/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1498 - accuracy: 0.1600
```

```
Epoch 7/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1547 - accuracy: 0.2467
Epoch 8/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1183 - accuracy: 0.1800
Epoch 9/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1128 - accuracy: 0.1533
Epoch 10/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1221 - accuracy: 0.2200
Epoch 11/100
50/50 [==============================] - 0s 3ms/step - loss: 2.1223 - accuracy: 0.1800
Epoch 12/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0150 - accuracy: 0.1800
Epoch 13/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0401 - accuracy: 0.2067
Epoch 14/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0190 - accuracy: 0.2400
Epoch 15/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0695 - accuracy: 0.2333
Epoch 16/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0202 - accuracy: 0.2267
Epoch 17/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0091 - accuracy: 0.2200
Epoch 18/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9537 - accuracy: 0.2267
Epoch 19/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9552 - accuracy: 0.2933
Epoch 20/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9664 - accuracy: 0.2133
Epoch 21/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9391 - accuracy: 0.2733
Epoch 22/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9306 - accuracy: 0.2333 ET
A: 0s - loss: 2.0696 - accuracy: 0.15
Epoch 23/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8785 - accuracy: 0.3000
Epoch 24/100
50/50 [==============================] - 0s 3ms/step - loss: 2.0361 - accuracy: 0.2067
Epoch 25/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8984 - accuracy: 0.2933
Epoch 26/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8929 - accuracy: 0.2800
Epoch 27/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8943 - accuracy: 0.2600
Epoch 28/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9326 - accuracy: 0.2667
Epoch 29/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8447 - accuracy: 0.2933
Epoch 30/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8549 - accuracy: 0.3000
Epoch 31/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8437 - accuracy: 0.3133
Epoch 32/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7985 - accuracy: 0.2933
Epoch 33/100
50/50 [==============================] - 0s 3ms/step - loss: 1.9394 - accuracy: 0.2667
Epoch 34/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8492 - accuracy: 0.2267
Epoch 35/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8533 - accuracy: 0.3000
Epoch 36/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8341 - accuracy: 0.3000
Epoch 37/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8149 - accuracy: 0.3400
Epoch 38/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8504 - accuracy: 0.2400
Epoch 39/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6448 - accuracy: 0.4133
Epoch 40/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8332 - accuracy: 0.2733
Epoch 41/100
50/50 [==============================] - 0s 4ms/step - loss: 1.7177 - accuracy: 0.3533
Epoch 42/100
```

```
50/50 [==============================] - 0s 3ms/step - loss: 1.9423 - accuracy: 0.2933
Epoch 43/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7962 - accuracy: 0.3467
Epoch 44/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8536 - accuracy: 0.3267
Epoch 45/100
50/50 [==============================] - 0s 4ms/step - loss: 1.8159 - accuracy: 0.3133
Epoch 46/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7122 - accuracy: 0.3333
Epoch 47/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7209 - accuracy: 0.3267
Epoch 48/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6549 - accuracy: 0.3933
Epoch 49/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7377 - accuracy: 0.3000
Epoch 50/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7044 - accuracy: 0.4067
Epoch 51/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7379 - accuracy: 0.3467
Epoch 52/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8232 - accuracy: 0.2667
Epoch 53/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7596 - accuracy: 0.3667
Epoch 54/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7853 - accuracy: 0.3667
Epoch 55/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5833 - accuracy: 0.3667
Epoch 56/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7068 - accuracy: 0.3600
Epoch 57/100
50/50 [==============================] - 0s 4ms/step - loss: 1.6315 - accuracy: 0.3533
Epoch 58/100
50/50 [==============================] - 0s 4ms/step - loss: 1.6087 - accuracy: 0.4067
Epoch 59/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7170 - accuracy: 0.3667
Epoch 60/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6145 - accuracy: 0.4267
Epoch 61/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5843 - accuracy: 0.4133
Epoch 62/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5975 - accuracy: 0.3667
Epoch 63/100
50/50 [==============================] - 0s 4ms/step - loss: 1.5572 - accuracy: 0.4600
Epoch 64/100
50/50 [==============================] - ETA: 0s - loss: 1.7226 - accuracy: 0.39 - 0s 4ms
/step - loss: 1.7011 - accuracy: 0.4000
Epoch 65/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5578 - accuracy: 0.4600
Epoch 66/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7149 - accuracy: 0.3800
Epoch 67/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7183 - accuracy: 0.3867
Epoch 68/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7221 - accuracy: 0.3867
Epoch 69/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5489 - accuracy: 0.4733
Epoch 70/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5455 - accuracy: 0.3600
Epoch 71/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5368 - accuracy: 0.4800
Epoch 72/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5970 - accuracy: 0.3800
Epoch 73/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6864 - accuracy: 0.3733
Epoch 74/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5339 - accuracy: 0.4200
Epoch 75/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6296 - accuracy: 0.4333
Epoch 76/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6160 - accuracy: 0.4133
Epoch 77/100
50/50 [==============================] - 0s 3ms/step - loss: 1.4729 - accuracy: 0.5133
```

```
Epoch 78/100
50/50 [==============================] - 0s 3ms/step - loss: 1.8488 - accuracy: 0.3200
Epoch 79/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5787 - accuracy: 0.3800
Epoch 80/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5111 - accuracy: 0.4600
Epoch 81/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5473 - accuracy: 0.4467
Epoch 82/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6659 - accuracy: 0.4067
Epoch 83/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7146 - accuracy: 0.3533
Epoch 84/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5938 - accuracy: 0.4133
Epoch 85/100
50/50 [==============================] - 0s 3ms/step - loss: 1.4985 - accuracy: 0.4533
Epoch 86/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5828 - accuracy: 0.4000
Epoch 87/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7098 - accuracy: 0.4000
Epoch 88/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6099 - accuracy: 0.4267
Epoch 89/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6367 - accuracy: 0.3667
Epoch 90/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6991 - accuracy: 0.3533
Epoch 91/100
50/50 [==============================] - 0s 3ms/step - loss: 1.3340 - accuracy: 0.5067
Epoch 92/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5555 - accuracy: 0.4467
Epoch 93/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6806 - accuracy: 0.4067
Epoch 94/100
50/50 [==============================] - 0s 3ms/step - loss: 1.4706 - accuracy: 0.4667
Epoch 95/100
50/50 [==============================] - 0s 3ms/step - loss: 1.6432 - accuracy: 0.3800
Epoch 96/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7020 - accuracy: 0.3667
Epoch 97/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5104 - accuracy: 0.3867
Epoch 98/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5184 - accuracy: 0.4667
Epoch 99/100
50/50 [==============================] - 0s 3ms/step - loss: 1.7628 - accuracy: 0.3067
Epoch 100/100
50/50 [==============================] - 0s 3ms/step - loss: 1.5893 - accuracy: 0.3933
```

In [29]:

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

```
313/313 [==============================] - 2s 5ms/step - loss: 1.5071 - accuracy: 0.4439:
0s - loss: 1.5052 - accuracy:
```

Out[29]:

```
[1.5070774555206299, 0.4438999891281128]
```

In [30]:

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

Out[30]:

```
array([[1.42093757e-02, 2.05363352e-02, 1.02000117e-01, 4.52514589e-01,
        4.48653661e-02, 1.48441106e-01, 3.79784070e-02, 4.40853834e-02,
        1.19218983e-01, 1.61502641e-02],
       [9.50297806e-03, 6.87898636e-01, 2.46850977e-04, 3.33523203e-04,
        5.47318719e-04, 2.03422333e-05, 2.34327672e-04, 1.64016819e-04,
        1.58853158e-01, 1.42198965e-01],
       [8.84266198e-02, 3.48226577e-01, 1.70814674e-02, 2.04881318e-02,
        1.71994641e-02, 4.95366985e-03, 1.15356445e-02, 1.09788040e-02,
```

```
        2.80276626e-01, 2.00833023e-01],
       [6.83930516e-02, 9.42807347e-02, 3.54099227e-03, 1.43081986e-03,
        2.77194357e-03, 1.08096683e-04, 6.13887620e-04, 4.37786570e-04,
        7.78652191e-01, 4.97704260e-02],
       [6.76082214e-03, 2.70181592e-03, 1.30349353e-01, 3.49817425e-01,
        1.27694264e-01, 2.22330198e-01, 8.30499455e-02, 6.85855523e-02,
        6.21005055e-03, 2.50055688e-03]], dtype=float32)
```

In [31]:

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

Out[31]:

```
[3, 1, 1, 8, 3]
```

In [32]:

```
y_test[:5]
```

Out[32]:

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

In [33]:

```
from sklearn.metrics import confusion_matrix , classification_report
print("Sınıflandırma Sonucu : \n" , classification_report(y_test , y_classes))
```

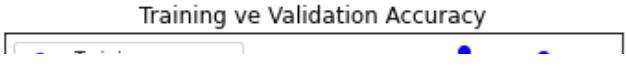
```
Sınıflandırma Sonucu :
              precision    recall  f1-score   support

           0       0.58      0.25      0.35      1000
           1       0.52      0.76      0.62      1000
           2       0.42      0.21      0.28      1000
           3       0.26      0.46      0.33      1000
           4       0.43      0.28      0.34      1000
           5       0.39      0.31      0.35      1000
           6       0.58      0.40      0.47      1000
           7       0.45      0.63      0.52      1000
           8       0.46      0.72      0.56      1000
           9       0.57      0.41      0.48      1000

    accuracy                           0.44     10000
   macro avg       0.47      0.44      0.43     10000
weighted avg       0.47      0.44      0.43     10000
```

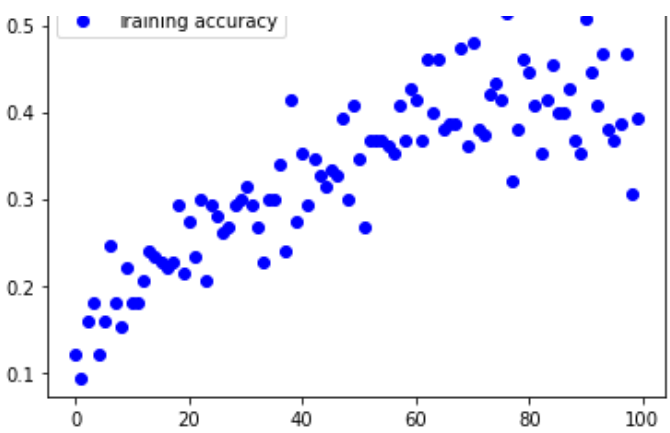
In [34]:

```
import matplotlib.pyplot as plt
%matplotlib inline

accuracy = history1.history['accuracy']
loss = history1.history['loss']
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.title('Training ve Validation Accuracy')
plt.legend()
plt.figure()
```

Out[34]:

```
<Figure size 432x288 with 0 Axes>
```






```
<Figure size 432x288 with 0 Axes>
```

In [ ]: