

```
[info] ROOT=/home/kim1484/assignment1_colab/assignment1
[info] DATA_DIR=/home/kim1484/assignment1_colab/assignment1/cs231n/datasets
[ready] 환경 준비 완료.
```

Image features exercise

Complete and hand in this completed worksheet (including its outputs and any supporting code outside of the worksheet) with your assignment submission. For more details see the [assignments page](#) on the course website.

We have seen that we can achieve reasonable performance on an image classification task by training a linear classifier on the pixels of the input image. In this exercise we will show that we can improve our classification performance by training linear classifiers not on raw pixels but on features that are computed from the raw pixels.

All of your work for this exercise will be done in this notebook.

Load data

Similar to previous exercises, we will load CIFAR-10 data from disk.

Extract Features

For each image we will compute a Histogram of Oriented Gradients (HOG) as well as a color histogram using the hue channel in HSV color space. We form our final feature vector for each image by concatenating the HOG and color histogram feature vectors.

Roughly speaking, HOG should capture the texture of the image while ignoring color information, and the color histogram represents the color of the input image while ignoring texture. As a result, we expect that using both together ought to work better than using either alone. Verifying this assumption would be a good thing to try for your own interest.

The `hog_feature` and `color_histogram_hsv` functions both operate on a single image and return a feature vector for that image. The `extract_features` function takes a set of images and a list of feature functions and evaluates each feature function on each image, storing the results in a matrix where each column is the concatenation of all feature vectors for a single image.

Done extracting features for 1000 / 49000 images
Done extracting features for 2000 / 49000 images
Done extracting features for 3000 / 49000 images
Done extracting features for 4000 / 49000 images
Done extracting features for 5000 / 49000 images
Done extracting features for 6000 / 49000 images
Done extracting features for 7000 / 49000 images
Done extracting features for 8000 / 49000 images
Done extracting features for 9000 / 49000 images
Done extracting features for 10000 / 49000 images
Done extracting features for 11000 / 49000 images
Done extracting features for 12000 / 49000 images
Done extracting features for 13000 / 49000 images
Done extracting features for 14000 / 49000 images
Done extracting features for 15000 / 49000 images
Done extracting features for 16000 / 49000 images
Done extracting features for 17000 / 49000 images
Done extracting features for 18000 / 49000 images
Done extracting features for 19000 / 49000 images
Done extracting features for 20000 / 49000 images
Done extracting features for 21000 / 49000 images
Done extracting features for 22000 / 49000 images
Done extracting features for 23000 / 49000 images
Done extracting features for 24000 / 49000 images
Done extracting features for 25000 / 49000 images
Done extracting features for 26000 / 49000 images
Done extracting features for 27000 / 49000 images
Done extracting features for 28000 / 49000 images
Done extracting features for 29000 / 49000 images
Done extracting features for 30000 / 49000 images
Done extracting features for 31000 / 49000 images
Done extracting features for 32000 / 49000 images
Done extracting features for 33000 / 49000 images
Done extracting features for 34000 / 49000 images
Done extracting features for 35000 / 49000 images
Done extracting features for 36000 / 49000 images
Done extracting features for 37000 / 49000 images
Done extracting features for 38000 / 49000 images
Done extracting features for 39000 / 49000 images
Done extracting features for 40000 / 49000 images
Done extracting features for 41000 / 49000 images
Done extracting features for 42000 / 49000 images
Done extracting features for 43000 / 49000 images
Done extracting features for 44000 / 49000 images
Done extracting features for 45000 / 49000 images
Done extracting features for 46000 / 49000 images
Done extracting features for 47000 / 49000 images
Done extracting features for 48000 / 49000 images
Done extracting features for 49000 / 49000 images

Train Softmax classifier on features

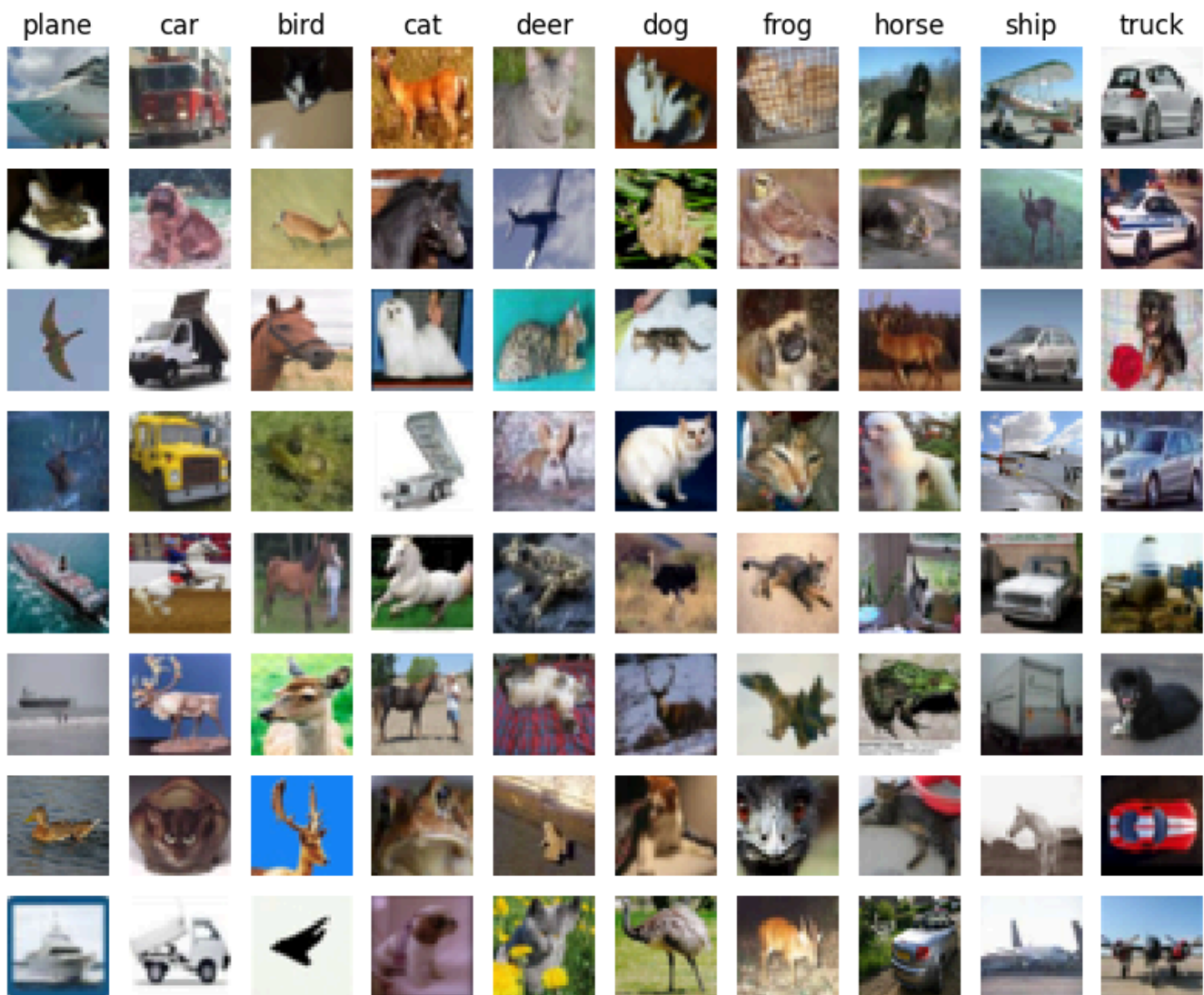
Using the Softmax code developed earlier in the assignment, train Softmax classifiers on top of the features extracted above; this should achieve better results than training them

directly on top of raw pixels.

```
/home/kim1484/assignment1_colab/assignment1/cs231n/classifiers/softmax.py:85: Runtime
Warning: divide by zero encountered in log
    loss -= np.sum(np.log(result))
/home/kim1484/assignment1_colab/assignment1/cs231n/classifiers/softmax.py:86: Runtime
Warning: overflow encountered in scalar multiply
    loss = loss / num_train + reg * np.sum(W * W)
/home/kim1484/anaconda3/envs/cs231n/lib/python3.8/site-packages/numpy/core/fromnumerici.py:86: RuntimeWarning: overflow encountered in reduce
    return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
/home/kim1484/assignment1_colab/assignment1/cs231n/classifiers/softmax.py:86: Runtime
Warning: overflow encountered in multiply
    loss = loss / num_train + reg * np.sum(W * W)
lr 1.000000e-07 reg 5.000000e+05 train accuracy: 0.420612 val accuracy: 0.432000
lr 1.000000e-07 reg 5.000000e+06 train accuracy: 0.348000 val accuracy: 0.364000
lr 1.000000e-06 reg 5.000000e+05 train accuracy: 0.323306 val accuracy: 0.335000
lr 1.000000e-06 reg 5.000000e+06 train accuracy: 0.100265 val accuracy: 0.087000
best validation accuracy achieved: 0.432000
/home/kim1484/assignment1_colab/assignment1/cs231n/classifiers/softmax.py:102: RuntimeWarning: overflow encountered in multiply
    dW = dW / num_train + reg * 2 * W
```

0.438

best_softmax_features.npy saved.



Inline question 1:

Describe the misclassification results that you see. Do they make sense?

Your Answer : Most of the misclassifications do not appear to make much sense, as there is no consistent pattern across most classes. However, for the plane class, the model tends to predict images with sky or sea backgrounds as plane. In contrast, the other classes show no consistent relationship with background features, indicating that the model's predictions are largely inconsistent and not based on stable visual cues.

Neural Network on image features

Earlier in this assignment we saw that training a two-layer neural network on raw pixels achieved better classification performance than linear classifiers on raw pixels. In this notebook we have seen that linear classifiers on image features outperform linear classifiers on raw pixels.

For completeness, we should also try training a neural network on image features. This approach should outperform all previous approaches: you should easily be able to achieve over 55% classification accuracy on the test set; our best model achieves about 60% classification accuracy.

```
(49000, 170)
```

```
(49000, 169)
```

```
learning rate: 0.001 regularization strength: 0.0 val accuracy: 0.612
```

```
learning rate: 0.001 regularization strength: 0.0001 val accuracy: 0.608
```

```
learning rate: 0.001 regularization strength: 0.0003 val accuracy: 0.613
```

```
learning rate: 0.001 regularization strength: 0.001 val accuracy: 0.62
```

```
learning rate: 0.0003 regularization strength: 0.0 val accuracy: 0.553
```

```
learning rate: 0.0003 regularization strength: 0.0001 val accuracy: 0.556
```

```
learning rate: 0.0003 regularization strength: 0.0003 val accuracy: 0.553
```

```
learning rate: 0.0003 regularization strength: 0.001 val accuracy: 0.559
```

```
learning rate: 0.1 regularization strength: 0.0 val accuracy: 0.124
```

```
learning rate: 0.1 regularization strength: 0.0001 val accuracy: 0.112
```

```
learning rate: 0.1 regularization strength: 0.0003 val accuracy: 0.112
```

```
learning rate: 0.1 regularization strength: 0.001 val accuracy: 0.124
```

```
learning rate: 0.0005 regularization strength: 0.0 val accuracy: 0.591
```

```
learning rate: 0.0005 regularization strength: 0.0001 val accuracy: 0.578
```

```
learning rate: 0.0005 regularization strength: 0.0003 val accuracy: 0.59
```

```
learning rate: 0.0005 regularization strength: 0.001 val accuracy: 0.579
```

```
0.589
```

```
best_two_layer_net_features.npy saved.
```