exam 개인학습

Lec 1.5

Type something...

01) Higher Level Representations: Image Features

- 지금까지 우리는 raw pixel로 이미지를 다뤄왔다.
- 하지만 이미지 부분 부분을 모아서 feature를 만들고 이를 통해서 이미지를 파악하는 것이 linear classifier에서 더 낫다.
- 2가지 방식을 혼합해서 사용할 것이다.
- 첫 번째는 Histogram of Oriented Gradients(HOG)는 주변에 있는 색깔정보를 무시하고 이미지의 texture만 뽑아내는 것이다.
- 두 번째는 Color histogram으로 texture를 무시하고 색깔정보로 표현하는 것이다. 주어진 hog_feature, color_histogram으로 이미지의 feature를 추출하여 모델에 넣어줄 것이다.

```
from cs231n.features import *

num_color_bins = 10 # Number of bins in the color histogram
feature_fns = [hog_feature, lambda img: color_histogram_hsv(img, nbin=num_color_bins)]
X_train_feats = extract_features(X_train, feature_fns, verbose=True)
X_val_feats = extract_features(X_val, feature_fns)
X_test_feats = extract_features(X_test, feature_fns)

# Preprocessing: Subtract the mean feature
mean_feat = np.mean(X_train_feats, axis=0, keepdims=True)
X_train_feats -= mean_feat
X_val_feats -= mean_feat
X_test_feats -= mean_feat

# Preprocessing: Divide by standard deviation. This ensures that each feature
# has roughly the same scale.
std_feat = np.std(X_train_feats, axis=0, keepdims=True)
X_train_feats /= std_feat
X_val_feats /= std_feat
X_test_feats /= std_feat

# Preprocessing: Add a bias dimension
X_train_feats = np.hstack([X_train_feats, np.ones((X_train_feats.shape[0], 1))])
X_val_feats = np.hstack([X_val_feats, np.ones((X_val_feats.shape[0], 1))])
X_test_feats = np.hstack([X_test_feats, np.ones((X_test_feats.shape[0], 1))])
```

- `extract_features` 을 통해서 이미지를 추출해주었다.

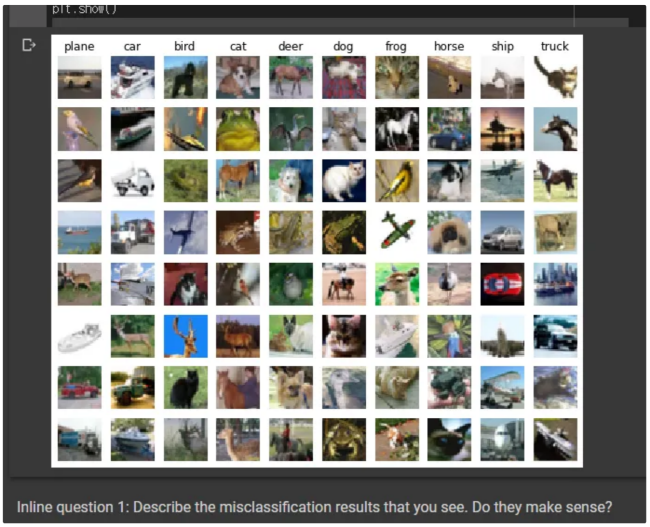
01_SVM

```python
for lr in learning_rates:
  for reg in regularization_strengths:
      svm = LinearSVM()
      result = (lr, reg)
      svm.train(X_train_feats, y_train, learning_rate = lr, reg = reg, num_iters = 1500,
                verbose = False)
      y_train_pred = svm.predict(X_train_feats)
      training_accuracy = np.mean(y_train == y_train_pred)

      y_val_pred = svm.predict(X_val_feats)
      validation_accuracy = np.mean(y_val == y_val_pred)

      if validation_accuracy > best_val:
        best_val = validation_accuracy
        best_svm = svm

      results[result] = (training_accuracy, validation_accuracy)

# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

# Print out results.
for lr, reg in sorted(results):
    train_accuracy, val_accuracy = results[(lr, reg)]
    print('lr %e reg %e train accuracy: %f val accuracy: %f' % (
                lr, reg, train_accuracy, val_accuracy))

print('best validation accuracy achieved: %f' % best_val)
```

```
lr 1.000000e-09 reg 5.000000e+04 train accuracy: 0.084980 val accuracy: 0.094000
lr 1.000000e-09 reg 5.000000e+05 train accuracy: 0.117184 val accuracy: 0.109000
lr 1.000000e-09 reg 5.000000e+06 train accuracy: 0.411633 val accuracy: 0.404000
lr 1.000000e-08 reg 5.000000e+04 train accuracy: 0.096163 val accuracy: 0.101000
lr 1.000000e-08 reg 5.000000e+05 train accuracy: 0.413265 val accuracy: 0.418000
lr 1.000000e-08 reg 5.000000e+06 train accuracy: 0.398612 val accuracy: 0.395000
lr 1.000000e-07 reg 5.000000e+04 train accuracy: 0.415612 val accuracy: 0.416000
lr 1.000000e-07 reg 5.000000e+05 train accuracy: 0.406612 val accuracy: 0.415000
lr 1.000000e-07 reg 5.000000e+06 train accuracy: 0.316020 val accuracy: 0.322000
best validation accuracy achieved: 0.418000
```

- `svm` 을 통해서 정확도 40%를 끌어낼 수 있었다.

1. Inline Question 1



A: 구분하기 힘든 이미지도 있지만 그렇지 않은 이미지들도 존재했다.

- 일반적으로 one layer보다 two layer일 때 성능이 더 좋다.
- 이번 과제에서는 55%이상의 정확도를 보이는 것을 목표로 하고 있다.
- 코드는 앞에서 했던 two layer코드와 똑같다.
- 대신 lr과 reg값은 달라진다. lr값만 조정시켜서 정확도를 55%넘기는 것을 목표로 삼았다.
- 1근처에서 높게나오는 것을 확인하고 그 근처에서 추적을 계속해주었다.

```
net = TwoLayerNet(input_dim, hidden_dim, num_classes)
best_net = None
best_val = -1

################################################################################
# TODO: Train a two-layer neural network on image features. You may want to    #
# cross-validate various parameters as in previous sections. Store your best   #
# model in the best_net variable.                                              #
################################################################################
# *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
lr_arr = np.random.uniform(1e-1, 1e0, 20)
for lr in lr_arr:
  print('lr:', lr)
  net = TwoLayerNet(input_dim, hidden_dim, num_classes)
  solver = Solver(net, data, optim_config = {'learning_rate': lr}, lr_decay = 0.95,
                  num_epochs = 5, batch_size = 200, print_every = 100)
  solver.train()
  val_acc = solver.check_accuracy(data['X_val'], data['y_val'])

  if val_acc > best_val:
    best_val = val_acc
    best_net = net
# *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
```

- hyperparameter tuning은 쉽지 않은 과정이다.
- 그래서 random으로 하기 보다는 정확도가 잘 나오는 hyperparameter 값 주변에서 탐색을 진행하여 값을 정제해 나가면 된다.