Homework 4 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

In the beginning...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. See Equation **??**.

$$a = b + c \tag{1}$$

Interesting Implementation Detail

My code snippet highlights an interesting point.

```
NBNN = true;

if NBNN
    D = zeros(C,M);
    C_train = cell(C, 1);
    for i=1:C
        idx = strcmp(train_labels, categories{i});
        C_train{i} = train_image_feats(idx, :);
    end
    num_f = 8;
    num_iter = length(train_image_feats(1,:)) / num_f;
    for i=1:M
        for j=1:C
```

```
                d = C_train{j} - test_image_feats(i,:);
                d = d.^2;
                dd = zeros(size(d,1), num_iter);
                for k=1:num_iter
                    for l=1:num_f
                        dd(:,k) = dd(:,k) + d( : , (k - 1) *
                            num_f + l);
                    end
                end
                dd = min(dd);
                D(j, i) = D(j, i) + sum(dd);
            end
            [~,idx] = min(D(:,i));
            predicted_categories{i} = categories{idx};
        end

else
```

This is about improving the nearest neighbor classifier using NBNN Algorithm part. When NBNN variable is ture, execute that algorithm.

I refer to the following formular.

The NBNN Algorithm:
1. Compute descriptors $d_1, \ldots, d_n$ of the query image Q.
2. $\forall d_i\ \forall C$ compute the NN of d_i in C: $\mathrm{NN}_C(d_i)$.
3. $\hat{C} = arg \min_C \sum_{i=1}^{n} \| d_i - \mathrm{NN}_C(d_i) \|^2$.

Figure 1: NBNN algorithm process

```
Getting paths and labels for all train and test data
Using bag of words representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.532
Getting paths and labels for all train and test data
Using bag of words representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.609
```

Figure 2: *Up:* accuracy without nbnn. *Down:* accuracy with nbnn.

```
if soft_assignment
    D = -10 * D.^2;
```

```
    eD = exp(D);
    image_feats(k,:) = image_feats(k,:) + (eD / sum(eD))
       .';
else
    [~, idx] = min(D(:,1));
     image_feats(k,idx) = image_feats(k,idx) + 1;
end
```

This is about soft assignment using kernel codebook encoding part. When soft assignment variable is ture, execute that algorithm.

I refer to the following formular.

Kernel codebook encoding [18, 20] is a variant in which descriptors are assigned to visual words in a soft manner. More specifically, each descriptor is encoded as $[\mathbf{f}_{\text{kcb}}(\mathbf{x}_i)]_k = K(\mathbf{x}_i, \mu_k)/\sum_{j=1}^{K} K(\mathbf{x}_i, \mu_j)$ where $K(\mathbf{x}, \mu) = \exp(-\frac{\gamma}{2}\|\mathbf{x} - \mu\|^2)$, and a set of N descriptors extracted from an image as $\mathbf{f}_{\text{kcb}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{f}_{\text{kcb}}(\mathbf{x}_i)$. Both give an encoding of size K.

Figure 3: kernel codebook encoding process

```
Getting paths and labels for all train and test data
Using bag of words representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.609
```

```
Getting paths and labels for all train and test data
Using bag of words representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.613
```

Figure 4: *Up:* nn accuracy without soft assign. *Down:* nn accuracy with soft assign.

```
Getting paths and labels for all train and test data
Using bag of words representation for images
Using support vector machine classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.630
```

```
Getting paths and labels for all train and test data
Using bag of words representation for images
Using support vector machine classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.646
```

Figure 5: *Up:* svm accuracy without soft assign. *Down:* svm accuracy with soft assign.

A Result

Therefore, Best model is bow and svm model with soft assignment.

```
Getting paths and labels for all train and test data
Using tiny image representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.219
Getting paths and labels for all train and test data
Using bag of words representation for images
Using nearest neighbor classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.613
Getting paths and labels for all train and test data
Using bag of words representation for images
Using support vector machine classifier to predict test set categories
Creating results_webpage/index.html, thumbnails, and confusion matrix
Accuracy (mean of diagonal of confusion matrix) is 0.646
```

Figure 6: *Up:* tiny and nn accuracy. *Middle:* bag and nn accuracy. *Down:* bag and svm accuracy.