Comparison of Different Novel Methods

on Scene Classification Problem

Group 16: 0856622 余家宏、309551067 吳子涵、309551122 曹芳驊

Video link: link

Motivation

Because our performance of Homework 5 was not as good as expected, we would like to further configure whether we could improve the classification performance by other novel methods. The original approach in Homework 5, BoW (Bag of Words), simply counts the number of descriptors associated with each cluster in a codebook, and creates a histogram for each set of descriptors from an image. Therefore, BoW doesn't consider much information, such as the first order statistics and the relevance among local descriptors in an image. We would like to try different approaches in order to improve this part and in turn perform the classification with SVM.

First, we try to apply VLAD (Vector of Locally Aggregated Descriptors), which accumulates the residual of each descriptor with respect to its assigned cluster. The primary advantage of VLAD over BoW is that we add more discriminative property in our feature vector by adding the difference of each descriptor from the mean in its voronoi cell.

Second, we try a deep learning method, named NetVLAD, which is built on top of the VLAD algorithm with NN architecture. It can not only retain the advantages of the classical VLAD algorithm, but also use neural networks to learn better parameters. Thus, NetVLAD is expected to generate better descriptors than the VLAD.

Third, we also further train two end-to-end networks, VGG16 and ResNet34 to perform classification.

In this project, we use different approaches in order to improve the performance of the BoW, and further make the comparison among these methods.

Method & Implementation

Method

1. VLAD

Vector of locally aggregated descriptors, known as VLAD, is an image feature encoding and pooling methods proposed in《Aggregating local descriptors into a compact image representation》[1].

As for bag of words (BoW), we first learn a codebook $C = \{c_{1,} \dots c_k\}$ of k visual words with k-means, and each local descriptor x is associated to its nearest visual word $c_i = NN(x)$. The idea of the VLAD descriptor is to accumulate the differences $x - c_i$ of the vectors x assigned to c_i for each visual wordc_i. This characterizes the distribution of the vectors with respect to the center.

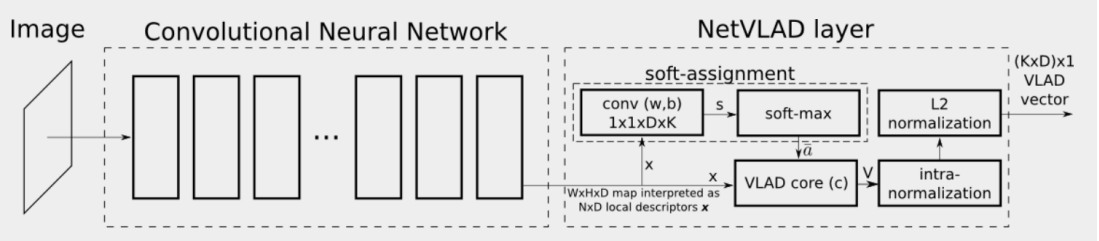
Assuming the local descriptor to be d-dimensional, the dimension D of our representation is $D = k \times d$. In the following, we represent the descriptor by $v_{i,j}$, where the indices $i = 1 \dots k$ and $j = 1 \dots d$ respectively index the visual word and the local descriptor component. Hence, a component of v is obtained as a sum over all the image descriptors:

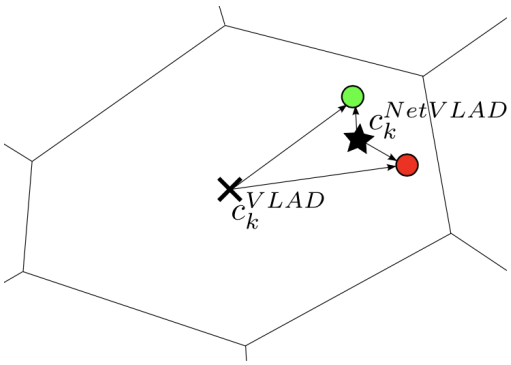
$$v_{i,j} = \sum_{x \text{ such that } \mathrm{NN}(x) = c_i} x_j - c_{i,j}$$

2. NetVLAD

In《NetVLAD: CNN architecture for weakly supervised place recognition》[2], they present a new generalized VLAD convolutional neural network architecture layer, called NetVLAD. The VLAD formula can be represented as $V(j,k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)\,(x_i(j) - c_k(j))$, where $a_k(x_i)$ denotes the membership of the descriptor x_i to k-th visual word (i.e., its value will be 1 or 0, depends on x_i belongs to

cluster c_k or not) . To make NetVLAD an end-to-end trainable network layer, they replace a_k with a differentiable softmax operation, $\bar{a}_k(\mathbf{x}_i) = \frac{e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_k\|^2}}{\sum_{k'} e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_{k'}\|^2}}$, which assigns the weight of descriptor x_i to cluster c_k proportional to their proximity, but relative to proximities to other cluster centers.

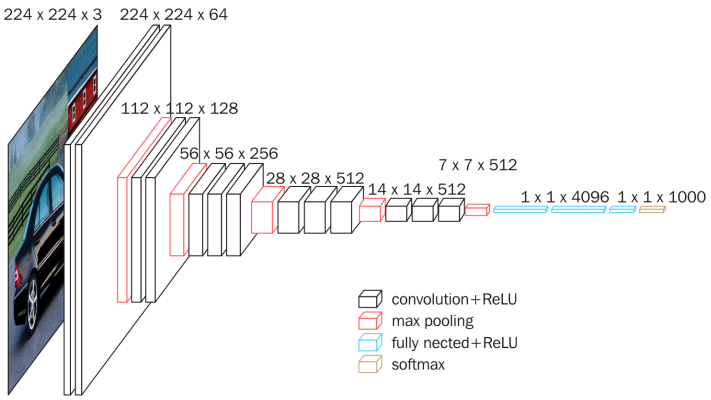




Different from VLAD, NetVLAD can learn better cluster center c_k because in a weakly-supervised setting, the descriptors are known to belong to images which should match or not.

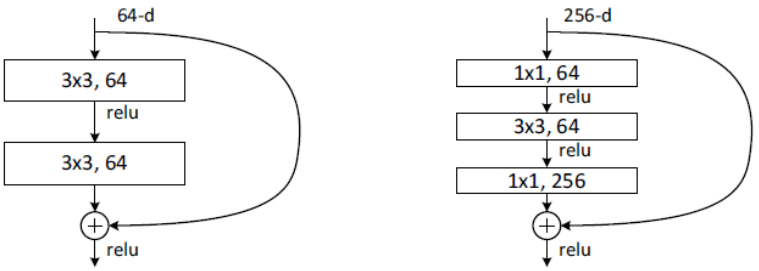
3. VGG16

VGG is a famous CNN proposed by Oxford Visual Geometry Group in 2014. This architecture is the 1st runner up of ILSVR2014 in the classification task. The main feature of VGG is using lots of 3×3 convolutional layers to enlarge the receptive field. Additionally, using more small convolution filters can enhance non-linearity, and have less parameters than large filters, which leads to better performance.



4. ResNet

In《Deep Residual Learning for Image Recognition》, the authors noticed that deeper neural networks can make the performance better, but also harder to train due to degradation problems. Hence, they proposed two new architectures called "Residual Block" and "Bottleneck Block", which will directly forward the value without applying convolution. With this architecture, ResNet can easily grow deeper but still easy to train.



left: Residual Block
right: Bottleneck Block

Implementation

1. VLAD

We follow the step in [1] to hand-craft the VLAD algorithm. First, extract the dense sift feature which yields a set of 128-dim local descriptors and put all the local descriptors into k-means clustering. In our experimebetternt, we have tried K=32, 64.

For each image, accumulating all of the residual vectors between local descriptors and the corresponding centroids which yields a $k \times d$ vector V. In turn, flatten V and do the SSR-normalization ($Sign(xi)\sqrt{|xi|}$) and L2-normalization ($x/\sqrt{\sum_i xi^2}$) inorder.

The output of the above VLAD descriptor is a $k \times d$ dimension vector. In our implementation, it is 32(64) x 128, which is too large and might cause the lack of efficiency and the performance defect in the subsequent classification stage. Hence, we further perform PCA on the output data to reduce the dimension. We have tried D = 128, 256, 512, 1024 for the final output dimensions.

In the final classification part, we still use SVM as the classifier in order to compare the performance with the method of Homework 5 (BoW).

Below is the implementation code :

```
def VLAD(img_fs, kmeans, k):
    centers = kmeans.cluster_centers_
    ret = []
    for fs in img_fs:
        V = np.zeros_like(centers)
        # labels = kmeans.predict(fs)
        labels, _ = pairwise_distances_argmin_min(fs, centers)
        diffs = fs - centers[labels]
        for l, diff in zip(labels, diffs):
            V[l] += diff
        V = V.flatten()

        # SSR normalization
        V = np.sign(V)*np.sqrt(np.abs(V))
        # L2 normalization
        V = V/np.sqrt(np.dot(V,V))

        ret.append(V)
    return ret
```

```
for i in range(7,11):
    D = 2**i
    print(f'export D :{D}')
    pca = PCA(n_components=D)
    train_vlad_pca = pca.fit_transform(train_vlad)
    test_vlad_pca = pca.transform(test_vlad)
```

2. NetVLAD

For the NetVLAD part, we use it as a global descriptor extractor, just as the VLAD part. We do not re-train the network by ourselves but use the origin network pre-trained on the Pitt250k [3] (source), which uses the VGG16 as its backbone feature extractor, and outputs the 512-dim vector. Then, feed it into the NetVLAD pooling layer which cluster number = 64. Finally with the PCA, its output will be a 4096-D vector which we use as the image's global descriptors.

In order to compare with the previous result, we don't directly train the network end-to-end. Instead, we feed the NetVLAD output into the SVM model. We compare that using the network as a feature extractor or not to see the improvement of NetVLAD over VLAD.

```
with tf.name_scope('pred'):
    with tf.device('/gpu:0'):
        pred_out = self._model(data, Mode.PRED, **self.config)
self.pred_out = {n: tf.identity(p, name=n) for n, p in pred_out.items()}
```

```
class NetvladOriginal(BaseModel):
    input_spec = {
            'image': {'shape': [None, None, None, 3], 'type': tf.float32},
    }
    required_config_keys = []
    default_config = {
            'num_clusters': 64,
            'pca_dimension': 4096,
    }

    def _model(self, inputs, mode, **config):
        image_batch = inputs['image']

        with tf.variable_scope('vgg16_netvlad_pca'):
```

3. VGG16

We adopt VGG16 architecture by only changing the fully connected layer to fit our class number. Train the fully connected layer first, and then fine tune the entire network.

```
class VGG16(nn.Module):
    def __init__(self,num_class):
        super(VGG16,self).__init__()
        self.model = models.vgg16(pretrained=True)

        for param in self.model.features.parameters():
            param.require_grad = False

        num_features = self.model.classifier[6].in_features
        features = list(self.model.classifier.children())[:-1]
        features.extend([nn.Linear(num_features, 15)])
        self.model.classifier = nn.Sequential(*features)

    def forward(self,X):
        out=self.model(X)
        return out
```

4. ResNet34

We adopt ResNet34 architecture by only changing the fully connected layer to fit our class number. Train the fully connected layer first, and then fine tune the entire network.

```
class ResNet34(nn.Module):
    def __init__(self,num_class):
        super(ResNet34,self).__init__()
        self.model = models.resnet34(pretrained=True)

        for param in self.model.parameters():
            param.requires_grad=False

        num_ftrs = self.model.fc.in_features
        self.model.fc = nn.Linear(num_ftrs, 15)

    def forward(self,X):
        out=self.model(X)
        return out
```

Results

1. VLAD

Because the dataset of Homework 5 is a small dataset, in order to avoid overfitting, we train our SVM model with the 3-fold cross-validation to find the best SVM parameter. We have tried k = 32, 64, PCA-Dim = 128, 256, 512, 1024 and origin (without PCA) parameter sets for our SVM input. The best test accuracy we've got is **76.66**% in (K=64, PCA-D = 1024)

K\PCA-D	128	256	512	1024
32	73.33	68.66	72.66	74
64	74.66	74.66	76	**76.66**

We can see that when K=64, the accuracy is better in general, and for the PCA dimension, the higher value also retrieves better accuracy.

2. NetVLAD

The best test accuracy we've got with NetVLAD descriptor is **86.66**%. This might be a little tricky. Compared to the VLAD approach, it has about 10% improvement. But compared to our end-to-end classification approach with VGG-16 architecture, it's 4% behind. The backbone of NetVLAD is also a part of VGG-16, so the result of these two approaches we expected to be on the same level. However, because the NetVLAD is not trained on the purpose of image classification and we also use the SVM as a classifier, which might not be as powerful as the end-to-end approach, we think the 4% accuracy gap is reasonable.

3. VGG16

Because the dataset provided by Homework 5 is too small, it is hard to train a neural network. Thus, we normalize and apply random flip to images in order to avoid overfitting.

```
Training:
epoch 1, Loss: 2.2955, Acc: 29.53
epoch 2, Loss: 0.9027, Acc: 69.93
epoch 3, Loss: 0.6104, Acc: 78.60
epoch 4, Loss: 0.5059, Acc: 82.47
epoch 5, Loss: 0.5300, Acc: 81.93
Fine Tuning:
epoch 1, Loss: 0.4366, Acc: 84.33
epoch 2, Loss: 0.4354, Acc: 84.33
epoch 3, Loss: 0.3717, Acc: 88.07
epoch 4, Loss: 0.3676, Acc: 87.33
epoch 5, Loss: 0.3683, Acc: 86.33
epoch 6, Loss: 0.3280, Acc: 87.73
epoch 7, Loss: 0.3521, Acc: 88.53
epoch 8, Loss: 0.3416, Acc: 88.60
epoch 9, Loss: 0.2855, Acc: 89.13
epoch 10, Loss: 0.2760, Acc: 91.07
epoch 11, Loss: 0.2927, Acc: 89.67
epoch 12, Loss: 0.2724, Acc: 89.87
epoch 13, Loss: 0.2914, Acc: 89.53
epoch 14, Loss: 0.2499, Acc: 92.13
epoch 15, Loss: 0.2550, Acc: 91.87
Test Acc: 90.67
```

4. ResNet34

We applied the same data augmentation process as we used with VGG and got 92.00% accuracy after training 15 epochs.

```
Training:
epoch 1, Loss: 2.7064, Acc: 10.73
epoch 2, Loss: 2.1113, Acc: 41.20
epoch 3, Loss: 1.6892, Acc: 56.93
epoch 4, Loss: 1.4072, Acc: 68.13
epoch 5, Loss: 1.2278, Acc: 70.40
Fine Tuning:
epoch 1, Loss: 1.0211, Acc: 75.87
epoch 2, Loss: 0.7134, Acc: 81.87
epoch 3, Loss: 0.5574, Acc: 85.80
epoch 4, Loss: 0.4701, Acc: 86.93
epoch 5, Loss: 0.4440, Acc: 87.13
epoch 6, Loss: 0.3850, Acc: 89.27
epoch 7, Loss: 0.3389, Acc: 90.73
epoch 8, Loss: 0.3739, Acc: 88.40
epoch 9, Loss: 0.3106, Acc: 91.53
epoch 10, Loss: 0.2900, Acc: 91.13
epoch 11, Loss: 0.2760, Acc: 93.00
epoch 12, Loss: 0.2658, Acc: 92.33
epoch 13, Loss: 0.2406, Acc: 92.80
epoch 14, Loss: 0.2472, Acc: 92.80
epoch 15, Loss: 0.2189, Acc: 93.80
Test Acc: 92.00
```

Discussion & Conclusion

The following table shows the accuracy of each algorithm, including BoW+SVM in Homework 5, VLAD+SVM, NetVLAD+SVM, VGG16, and ResNet34.

Method	BoW	VLAD	NetVLAD	VGG16	ResNet34
Accuracy	54%	76.66%	86.66%	90.67%	92.00%

BoW only counts the number of descriptors associated with each cluster in a codebook, and VLAD is an extension of this concept. VLAD stores the sum of the differences of the descriptors assigned to the cluster and the centroid of the cluster, which provides more information for generating better global descriptors. From our result, using VLAD instead of BoW to perform classification indeed improves the performance a lot.

NetVLAD could generate better descriptors than VLAD by training neural networks to learn better parameters because in a weakly-supervised setting, the descriptors are known to belong to images which should match or not. With good descriptors, the performance of classification could be improved. The NetVLAD result of our experiment shows much improvement.

We also compare with the end-to-end network architectures, such as VGG16 and ResNet34. It can improve the performance by directly training the entire network from images to labels, rather than generating the descriptors first.

For the NetVLAD part, we haven't further tried using the traditional sift local descriptor as the NetVLAD pooling layer input and re-train the network because in the paper it mentioned that one of the advantages of NetVLAD is that it can take advantage of the end-to-end training for the image representation. However, this experiment might help us further investigate the capability of the pooling layer of NetVLAD.

In this project, we implement and compare many algorithms in order to get better performance of scene classification. We improve the performance step by step through BoW, VLAD, NetVLAD, VGG16, to ResNet34, finding that end-to-end model performs best and deep learning is such a powerful tool indeed.

Work assignment plan between team members

- Code: 吳子涵、曹芳驊
- Experiment: 余家宏、吳子涵、曹芳驊
- Report: 余家宏、吳子涵、曹芳驊
- Oral: 余家宏、曹芳驊

Reference

1. Hervé Jégou, Matthijs Douze, Cordelia Schmid, Patrick Pérez. Aggregating local descriptors into a compact image representation. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*

2. Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

3. Akihiko Torii, Josef Sivic, Masatoshi Okutomi, Tomas PajdlaVisual Place Recognition with Repetitive Structures *EEE PAMI*, 2015.