Final Project: Image Classification

Dhruba Pujary **Tarun Krishna**
11576200 11593040
University of Amsterdam
{dhruba.pujary, tarun.krishna}@student.uva.nl

1 Introduction

The goal of the assignment is to implement a system for image classification; in other words, this system should tell if there is an object of a given class in an image. We will perform four-class (airplanes, motorbikes, faces, and cars) image classification based on bag-of-words approach. Next part of the task is to make use of Convolutional Neural Networks(CNN) for the same.Former approach makes use of hand-crafted features like SIFT to represent images, then trains a classifier on top of them. In this way, learning is a two-step procedure with image representation and learning. The method used in latter approach instead learns the features jointly with the classification.

2 Bag of Words

Object recognition in images has been a well known problem in computer vision. One way to solve this problem is to represent the object in some form of feature vectors, like pixel intensity, its position in the image, raw color value of the image etc. Another approach which is inspired by natural language processing field is using a bag of words. The idea is to represent the object by a collection of small low-level details or subparts which can be used for identification or classification. For example, if the visual "words" in the image are eyes, nose, ears, mouth etc. then most likely we are looking at something that resembles a face.

2.1 Dataset

The dataset used for this task is from Caltech which consists of images of 4 class type; Airplanes, Cars, Faces, and Motorbikes. Each class consists of 500 training images and 50 test images.

2.2 Feature extraction

Feature extraction is done to collect the relevant information from an image and to reduce the amount of resource required to describe the image. Edges, corner, blobs are some of the features that are used. SIFT descriptors are one of the feature descriptors that describes image patches and also have a very nice scale-invariance properties which is also widely used along with its many color variants.

SIFT Descriptors: The SIFT descriptor proposed by [13] is a 3-D spatial histogram of the image gradients describing the local space of a region. The feature vectors are invariant to image translation, scaling and rotation, partially invariant to illumination changes and robust to local geometric distortion. The descriptor is a 128-dimensional vector which is 4×4 window around an interest point and at each window the histogram of the orientation gradients in 8 bins (i.e. $4 \times 4 \times 8$).

SIFT descriptors can be extracted at different interest points in an image, i.e. key points sampling, or at every location known as dense sampling. SIFT descriptors can also be computed on an RGB color image by extracting the descriptors for every color channel in the image. Similarly, for rgb and opponent color, the SIFT descriptors are also known as rgb-SIFT or opponent-SIFT respectively.

2.3 Visual Vocabulary

The feature descriptors obtained after feature extraction are then used to build visual words by clustering, where each cluster center is a visual word. The number of clusters in knows as the vocabulary size and the collection of such visual words is called visual vocabulary.

2.4 Quantization

An image can be represented as a collection of visual words. This is done by extracting the feature descriptors of the image and assigning it to the closest visual word in the vocabulary. The count of these visual words in the form of the histogram is then used to represent the image.

2.5 Classification and training

The task of a bag of words representation is given any image we are able to classify the object to its correct class. With the quantized representation of the image, i.e. the histogram of visual words, we train a Support Vector Machine classifier per object. From the given dataset, for each classifier, we evaluate the histogram of visual words of the images. The training set is then prepared which consist of images of the correct class as positive examples and images from other class as negative examples. These histogram of visual words along with their correct labels is then used to train a binary classifier, which is then used for prediction.

2.6 Testing

Testing is done on the testing dataset by extracting the feature descriptors of the image, assigning them to their closest visual word followed by evaluating the histogram of visual words. This histogram of visual words is then used to obtain the prediction using a binary classifier of a class, that whether or not the image contains the object of the class. This is performed for every image in the training set and for each class. A ranking of these images is obtained per class which is then used to calculate the average precision per class and the mean average precision over all class.

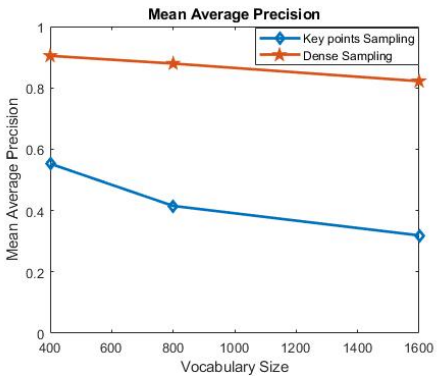
3 Results and analysis of BoW

Experiments were conducted by varying the sampling type namely key points and dense sampling with different step size, the type of SIFT descriptor, the vocabulary size, number of example per class used for generating the vocabulary, and number of positive and negative examples used for training the SVM classifier with a linear kernel. From figure 2, dense sampling performs better than key points sampling in general. The number of shift descriptors generated by dense sampling is more than key points which allow capturing more information from an image than key points sampling.

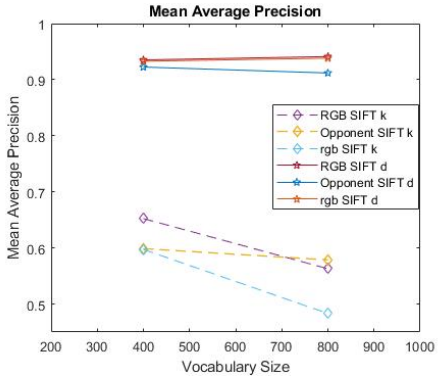
In key points sampling, increasing the vocabulary size(figure 1(a)) does not guarantee better MAP and however, increasing the number of example for generating vocabulary words and training the SVM improves the MAP. In dense sampling, increasing the vocabulary size does not give better results in most of the cases(figure 1(a), figure 1(b)). However, decreasing the step size does improve the result for vocabulary size of 800 but need not be true always as in case of vocabulary size of 400. Increasing the number of training examples for SVM classifier improves results of both key points and dense sampling. Thus, for comparison of other types of SIFT descriptors, we used the best settings, i.e vocabulary size of 400 and 800, 400 training examples of which 50 training examples of each class is used for generating vocabulary words and rest as positive and negative examples.[1]

In case of color SIFT descriptors, the model with dense sampling of step size 10px, vocabulary size of 800 words and 400 training examples perform better than rest. Normalized RGB also with similar settings but with step size of 15 pixels gave nearly similar MAP. Changing the step size of RGB to 8 pixels showed a dip in the MAP of approx 0.45% which is similar to RGB with vocabulary size of 400 and step size of 10px. Normalized rgb with vocabulary size of 400 and step size 15px gave good results of 93.288% MAP which is better than opponent SIFT with 92.253% and 91.194% MAP for

[1]We couldn't perform many experiments because of limitation of computational power; with multiple system crash. Thus few results cannot be compared directly unless they are under similar setting.



(a) Key points and Dense sampling



(b) Color SIFT (k:Key points and d:Dense sampling)

Figure 1: Mean Average Precision with varying vocabulary size

400 and 800 vocabulary words respectively. The color SIFT's performs nearly as good as gray SIFT because the SIFT descriptors of the individual color channel are adding similar information as the gray SIFT does. The color SIFT descriptors with key points sampling perform similar to gray SIFT with key points sampling(figure1(b)).

Sampling	SIFT	Vocabulary Size	Example per class	Vocabulary Fraction	Positive Examples	Negative Examples	Step Size	MAP
Key points	-	400	150	1/3	100	300	-	0.55276
Key points	-	400	300	1/3	200	600	-	0.56328
Key points	-	800	150	1/3	100	300	-	0.41509
Key points	-	800	300	1/3	200	600	-	0.48231
Key points	-	1600	150	1/3	100	300	-	0.31891
Key points	-	2000	150	1/3	100	300	-	0.29656
Key points	RGB	400	400	1/8	350	1050	-	0.65236
Key points	RGB	800	400	1/8	350	1050	-	0.56295
Key points	rgb	400	400	1/8	350	1050	-	0.59726
Key points	rgb	800	400	1/8	350	1050	-	0.48317
Key points	Opponent	400	400	1/8	350	1050	-	0.59877
Key points	Opponent	800	400	1/8	350	1050	-	0.57872
Dense	-	400	150	1/3	100	300	10	0.90444
Dense	-	400	150	1/3	100	300	8	0.89302
Dense	-	800	100	1/2	50	150	7	0.84687
Dense	-	800	150	1/3	100	300	8	0.93593
Dense	-	800	150	1/3	100	300	10	0.87929
Dense	-	800	200	1/4	150	450	10	0.90221
Dense	-	800	400	1/8	350	1050	8	0.95262
Dense	-	1600	100	1/2	50	150	10	0.70538
Dense	-	1600	150	1/3	100	300	10	0.82183
Dense	-	1600	400	1/8	350	1050	10	0.85604
Dense	RGB	400	400	1/8	350	1050	10	0.93008
Dense	RGB	800	400	1/8	350	1050	10	0.9412
Dense	RGB	800	400	1/8	350	1050	8	0.9367
Dense	rgb	400	400	1/8	350	1050	15	0.93288
Dense	rgb	800	400	1/8	350	1050	15	0.93833
Dense	Opponent	400	400	1/8	350	1050	15	0.92253
Dense	Opponent	800	400	1/8	350	1050	15	0.91194

Figure 2: Mean Average Precision of BoW with different settings and SVM with Linear Kernel

Upon testing with different kernels for SVM, the linear kernel performed the best in comparison to other kernels such as polynomial kernels and radial basis function with default settings. Polynomial

Kernel gave very poor MAP value without convergence during training. Whereas rbf kernel gave 0 MAP because the number of support vectors identified was very high, nearly equal to sample size.

3.1 Additional

[2] In this **section** we try to give an overview of different methods in the literature that has been employed to improve the accuracy Bag-of-Visual-word(BoW) model in the past(Deep-learning has replaced them all). The scope of improving the overall performance of the system can be worked out by looking at the different modules of BoW model.

1. **Feature Representation**: Since the annotation accuracy is heavily dependent on feature representation, using different region/point descriptors and/or the BoW feature representation will provide different levels of discriminative power for annotation. For example [15] compare 10 different local descriptors for object recognition. In [11] examines the classification accuracy of the BoW features using different numbers of visual words and different weighting schemes.

   Due to the drawbacks that vector quantization may reduce the discriminative power of images and the BoW methodology ignores geometric relationships among visual words, [22] present a novel scheme where SIFT features are bundled into local groups. These bundled features are repeatable and are much more discriminative than an individual SIFT feature. In other words, a bundled feature provides a flexible representation that allows us to partially match two groups of SIFT features.

   On the other hand, since the image feature generally carries mixed information of the entire image which may contain multiple objects and background, the annotation accuracy can degrade by such noisy (or diluted) feature representations. [5] propose a novel feature representation, pseudo-objects. It is based on a subset of proximate feature points with its own feature vector to represent a local area to approximate candidate objects in images. So, these comparison and new methods for feature representation can give us more idea that could help increasing the accuracy of our system. But exploiting feature representation is not the only option that one should look into for better performance.

2. **Vector Quantization** In order to reduce the quantization noise, [9] construct short codes using quantization. The goal is to estimate distances using vector-to centroid distances, that is, the query vector is not quantized, codes are assigned to the database vectors only. In other words, the feature space is decomposed into a Cartesian product of low-dimensional subspaces, and then each subspace is quantized separately. In particular, a vector is represented by a short code composed of its subspace quantization indices. On the other hand, [21] propose a Semantics Preserving Bag-of-Words (SPBoW) model, which considers the distance between the semantically identical features as a measurement of the semantic gap and tries to learn a codebook by minimizing this semantic gap. That is, the codebook generation task is formulated as a distance metric learning problem. In addition, one visual feature can be assigned to multiple visual words in different object categories. These trick of vector quantization can be useful in boosting systems accuracy as well.

3. **Visual Vocabulary Construction** Since related studies, such as [10, 14, 19], have shown that the commonly generated visual words are still not as expressive as text words, in [23], images are represented as visual documents composed of repeatable and distinctive visual elements, which are comparable to text words. They propose descriptive visual words (DVWs) and descriptive visual phrases (DVPs) as the visual correspondences to text words and phrases, where visual phrases refer to the frequently co-occurring visual word pairs. This robustness in constructing vocabulary may help in boosting some accuracy as well.

Above discussed approaches from literature may help in boosting systems overall performance but with the advent of deep learning most the techniques have become obsolete. In the next section 4 we exploit deep learning for the object classification as well.

[2] **BONUS** Any other thing that you can think of or find from the literature that can boost the results.

4 Understanding the Network Architecture

[3] The architecture given to us does follow some pattern. Firstly the model has been trained for 10 classes with an input of size $32 \times 32 \times 3$ (this is not the pattern). Pattern observed are as follows: Each *conv* layer is being followed by some *activation* layer or some *pooling*. The very first block of *conv* layer is followed by *max* pool and then with *relu* activation layer. Next block of *conv* is followed by *relu* activation and then *average* pooling (observe the difference in first block and second block). As compared to the first block, in second block the order of operation has changed, also *pooling* type has change from *max* to *average* pooling. Also, block4 *conv* is only followed with *relu* activation no pooling operation. In block5 conv is basically a fully connected layer which outputs a vector of size 10 which is then fed into a *softmax* layer to get a distribution over 10 classes.

[4] So, if we work around the mathematics we see that each block of convolutional layer has some weights and biases, so called parameters which are required to be learned. Accordingly, *conv1* has $5 \times 5 \times 3 \times 32$ (weights) $+32$ (biases), *conv2* has $5 \times 5 \times 32 \times 32 + 32$, *conv3* $5 \times 5 \times 32 \times 64 + 64$, *conv4* $4 \times 4 \times 64 \times 64 + 64$. Which means conv4 has highest weights and biases. Regarding biggest size, it depends upon what you consider size i.e output filter dimensions or activation map size. If we go by former conv3 and conv4 have output of 64 dimensions and if we go by latter then conv1 will have highest activation of size 32×32 ($32 \times 32 \times 32$ entire output from conv1).

4.1 Preparing the Input Data

The task of setting up the *imdb* data structure has been implemented in *finetune_cnn.m* with four classes namely *airplane, cars, faces, motorbikes* under function **getCaltechIMDB**.

4.2 Updating the Network Architecture

[5] NEW_INPUT_SIZE is 64 for block5 as it is the output from previous block which is 64 and NEW_OUTPUT_SIZE is the number of classed which is 4 in our case. These new values has been updated in file *update_model.m*

4.3 Setting up the Hyperparameters

Most machine learning algorithms involve hyper-parameters which are variables set before actually optimizing the models parameters. Setting the values of hyper-parameters can be seen as model selection, i.e choosing which model to use from the hypothesized set of possible models. In our case we are considering *batch size* and *number of epochs* as our hyper-parameters.

Epochs	Batch size	CNN Accuracy	Pretrained-SVM Accuracy	Fintune-SVM Accuracy
40	50	98.50	90.00	98.00
40	100	98.00	90.00	97.50
80	50	99.00	90.00	99.00
80	100	98.50	90.00	98.00
120	50	99.00	90.00	98.50
120	100	99.00	90.00	98.50

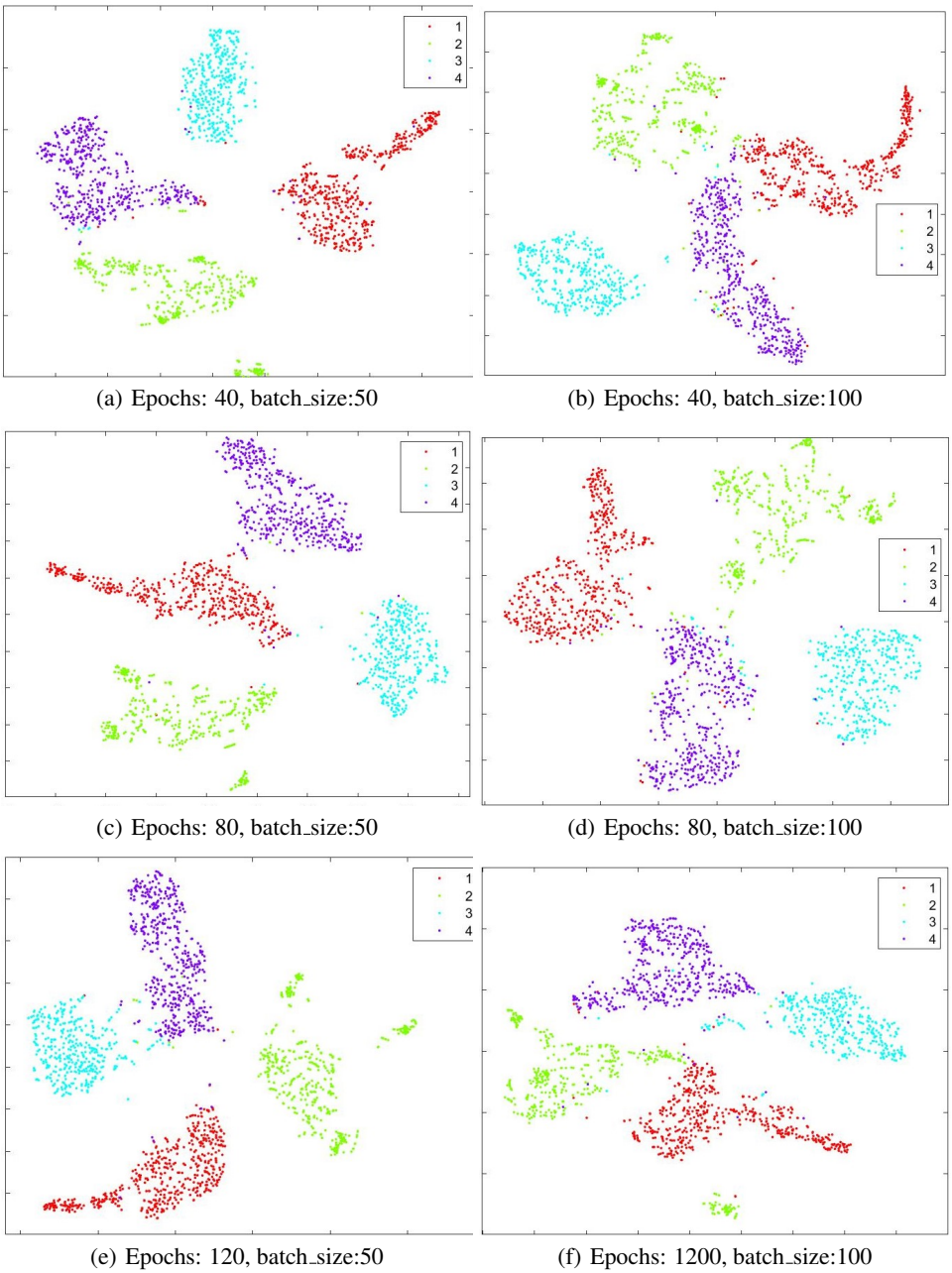
Figure 3: Accuracy for different hyper-parameters setting.

The most common way to set number of epochs is using the principle of early stopping. Early stopping simply stops training once performance on a held-out validation set stops increasing(i.s cost begins increasing steadily instead of decreasing). This can be a powerful way to prevent over-fitting, to the extent that it may make the choice of hyper-parameters less important. Theoretically, the choice of batch size is mostly computational, when batch size is larger, updates can be computed

[3] Do you observe any pattern in the architecture of the network? If so, describe it in your own words.

[4] Which part of the network has the most parameters and the biggest size?

[5] In this part of the assignment, you are expected to define the network structure. The architecture of the pre-trained network is provided in update model.m script. Specifically, it is expected to update the output layer in a way that it is trained for classifying images of 4 different classes (rather than 10). You need to set NEW_INPUT_SIZE and NEW_OUTPUT_SIZE accordingly in Block-5 of the network architecture.



(a) Epochs: 40, batch_size:50

(b) Epochs: 40, batch_size:100

(c) Epochs: 80, batch_size:50

(d) Epochs: 80, batch_size:100

(e) Epochs: 120, batch_size:50

(f) Epochs: 1200, batch_size:100

Figure 4: Feature space visualization for fine-tuned features for different parameters setting, with classes airplanes(red), car(green), faces(blue), motorbikes(purple).

more efficiently due to the use of parallel architectures, when batch size is smaller more updates can be made. Algorithmically speaking, using larger mini-batches in SGD allows you to reduce the variance of your stochastic gradient updates and this in turn allows you to take bigger step-sizes, which means optimizing algorithm will makes progress faster. However, the amount of work done to reach certain accuracy in the objectives will be the same: with a mini-batch size of n, the variance of the update direction will be reduced by a factor n, so the theory allows you to take step-sizes that are n times larger[1], so that a single step will take you roughly to the same accuracy as n steps of SGD with a mini-batch size of 1. We depict the results of varying different batch sizes and training epochs in Figure3.

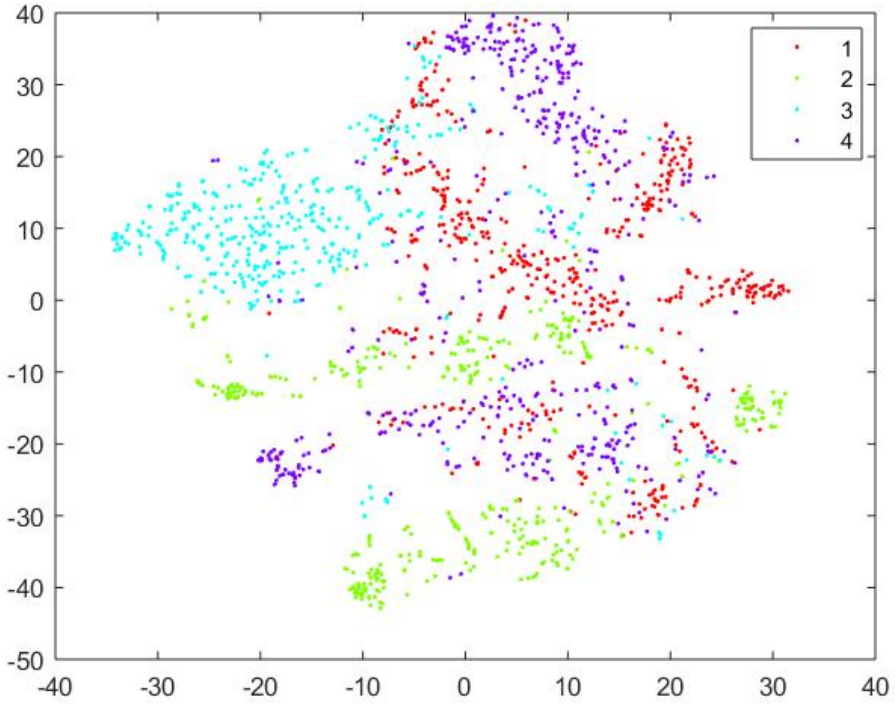
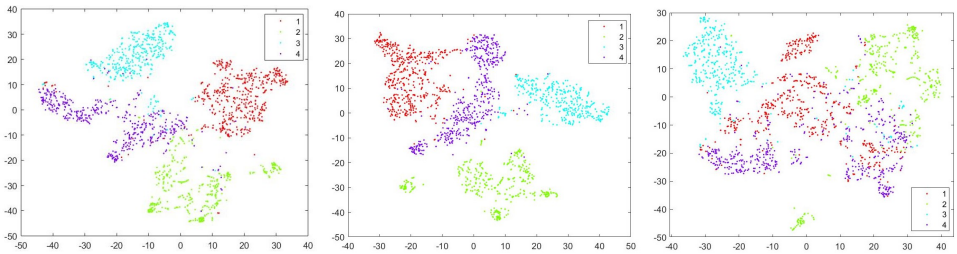


Figure 5: Feature space visualization for Pre-trained network on provided dataset. Classes airplanes(red), car(green), faces(blue), motorbikes(purple).

Analysis: As depicted in the Figure3 for given training epoch varying batch size hardly effects the accuracy. This also validates the fact stated above that, larger mini-batches helps in faster convergence of the algorithm without effecting the accuracy. Increasing the number of epochs we saw a very slight improvement in the accuracy, this is due to fact that the weights have not yet converged so training them further will help to get a better accuracy. Also the slight improvement in the accuracy could also be attributed to the fact that the problem we are dealing with is very simple learning problem only with 4 classes and that too with very distinct features. So, we can't comment more on this until and unless we are dealing with something as huge as ImageNet[6].



(a) mixed pooling (b) max pooling (c) average Pooling

Figure 6: Feature visualization to depict the effects of different pooling layers. Classes airplanes(red), car(green), faces(blue), motorbikes(purple).

Epochs	Batch size	Pooling Type	CNN Accuracy
50	50	Mixed	97.50
50	50	Max	98.00
50	50	Average	93.00

Figure 7: Effect of different pooling layers.

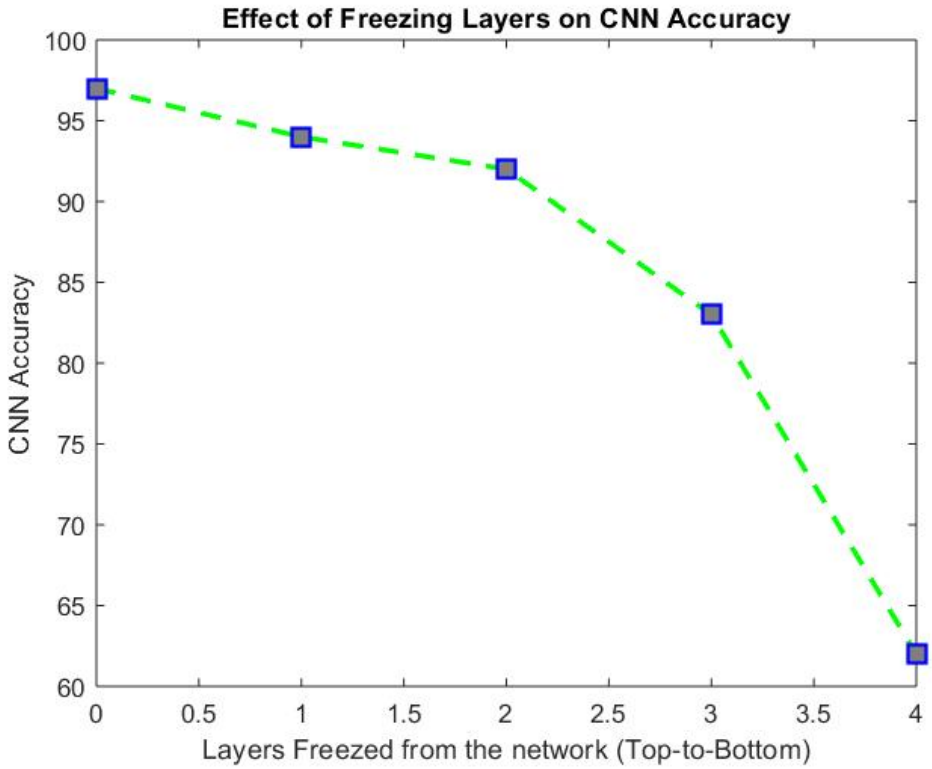


Figure 8: Plot depicting effect of freezing layers in the network.

5 Experiments

5.1 Feature Space Visualization

Feature space visualization for Fine-tuned and Pre-trained features can be observed in Figure4 and Figure5 respectively. Differences can be easily observed between the Figures. In Figure4 clusters of each classes are easily separable for all the different settings but this is not the case with Figure5 classes are cluttered this also very much consistent with accuracy of pre-trained model as mentioned in Figure3. So, using the given pre-trained model for extraction of features for a given data may not achieve as good accuracy as a fine-tuned model. To validate this fact we try to freeze the layers, i.e we don't update weights of pre-trained layers from the top of the network. As a result we observe decrease in overall performance of the system every time we increase the number of freezed layers. Results shown in Figure8. This implies that we need to fine-tune already pre-trained layers also.

5.2 Evaluating Accuracy

Accuracy for different hyperparameters is depicted in Figure3 and has been discussed in section4.3.

[6]Once we have the model of our network, we need to find the best set of parameters to minimize the training/test loss and maximize the accuracy of the model. Model solver brings together training data, the model and the optimization algorithms to train the model. In general there has been a lot of

[6]**BONUS** Any other thing that you can think of or find from the literature that can boost the results.

work in visual recognition using deep CNN models. Considering the kind of data we have 4 distinct classes it will be difficult to comment what other methods could boost up the accuracy as we are already having 99.00% accuracy. But in general for more complicated task of visual recognition using the technique of *Batch Normalization* [8] has helped in achieving better accuracy. Each deep CNNs comes with different approach to improve the accuracy. Few of them are listed below:

AlexNet[12]was the first CNN to win the ImageNet Challenge in 2012. It consists of five CONV layers and three FC layers. Within each CONV layer, there are 96 to 384 filters and the filter size ranges from 3×3 to 11×11, with 3 to 256 channels each. In the first layer, the 3 channels of the filter correspond to the red, green and blue components of the input image. A ReLU non-linearity is used in each layer. Max pooling of 33 is applied to the outputs of layers 1, 2 and 5. To reduce computation, a stride of 4 is used at the first layer of the network. AlexNet introduced the use of LRN in layers 1 and 2 before the max pooling, though LRN is no longer popular in later CNN models. One important factor that differentiates AlexNet from LeNet is that the number of weights is much larger and the shapes vary from layer to layer. To reduce the amount of weights and computation in the second CONV layer, the 96 output channels of the first layer are split into two groups of 48 input channels for the second layer, such that the filters in the second layer only have 48 channels. Similarly, the weights in fourth and fifth layer are also split into two groups. In total, AlexNet requires 61M weights to process one 227×227 input image. This work was the breakthrough in area of visual recognition
Overfeat [17] has a very similar architecture to AlexNet with five CONV layers and three FC layers. The main differences are that the number of filters is increased for layers 3 (384 to 512), 4 (384 to 1024), and 5 (256 to 1024), layer 2 is not split into two groups, the first fully connected layer only has 3072 channels rather than 4096, and the input size is 231×231 rather than 227×227. As a result, the number of weights grows to 146M. Overfeat has two different models: fast (described here) and accurate. The accurate model used in the ImageNet Challenge gives a 0.65% lower top-5 error rate than the fast model. *VGG-16*[18] goes deeper to 16 layers consisting of 13 CONV layers and 3 FC layers. In order to balance out the cost of going deeper, larger filters (e.g., 5×5) are built from multiple smaller filters (e.g., 3×3), which have fewer weights, to achieve the same receptive fields. As a result, all CONV layers have the same filter size of 3×3. In total, VGG-16 requires 138M weights to process one 224×224 input image. VGG has two different models: VGG-16 (described here) and VGG-19. VGG-19 gives a 0.1% lower top-5 error rate than VGG-16 at the cost of 1.27 more MACs. Like-wise *GoogleNet*[20] and *ResNet*[7]. So, the different models discussed above have very subtle difference among each other yet each model captures different visual properties of images. So, depending upon the task there could be various ways of improving accuracy which might require a little bit of tinkering as well.

6 Experiments and Results : CNN

We also, tried to tweak some other parameters apart from batch size and epochs. We tried to study the effect of different pooling layers on the accuracy. There has been quite a few work which focuses on effect of pooling in visual recognition [16, 3, 2]. It has been also shown in [4] that the optimal pooling type for a given classification problem may be neither max nor average pooling, but something in between. This can be viewed as an intermediate position in a parametrization from average pooling to max pooling over a sample of fixed size, where the parameter is the number of feature points over which the max is computed: the expected value of the max computed over one feature is the average, while the max computed over the whole sample is obviously the real max.

We initially trained a model for 50 epochs with 50 batch-size with mixed pooling as initially given to us in the task. And subsequently we also trained the model with only max and average pooling. The results are shown in Figure7 and feature visualization is depicted in 6. Max pooling clearly outperforms average pooling and mixed pooling as shown in Figure7 and feature space visualization Figure6 depicts it even more accurately.

[7]We also show the results of freezing layers on the accuracy as well. As shown in Figure8, we achieve higher accuracy when we fine-tune each layer but this accuracy goes on decreasing as we freeze our pre-trained layers. In our case we freezed maximum upto 4 blocks of conv layer.

7 Comparison

CNN architectures learns representation of an image i.e these are not hand-crafted features . However these layers cannot be used outside a trained CNN model for any other task. This is similar to the visual words which are generated by clustering feature descriptors into fixed size of vocabulary words. These visual words doesn't necessarily resemble any original feature descriptor and cannot be used for any general case. The CNN based model performs better than BoW words model with very higher accuracy rate. With the bag of words model, the best MAP obtained is 95.262% but with the CNN based accuracy is 98-99%. The high accuracy of CNN is because of the models ability to learn features relevant to the class data and optimize it for correct classification. CNN is end-to-end image classification method.

8 Conclusion

In conclusion, we experimented with two different model, Bag of Words and CNN for classifying objects into one of the four classes of the data set. In BoW, we tried different vocabulary size for building the codebook and two types of sampling; dense with different step size and key points.The dense sampling is better than keypoints sampling. For feature extraction, we tried gray SIFT as well as color SIFT such as RGB, rgb and opponent colors. For training the classifier, we used SVM and tested with linear, polynomial and RBF kernels. The MAP of gray SIFT is highest followed by color SIFT. While in CNN we experimented with different hyper-parameters settings. Along with that we also conducted some experiments to study the effects of freezing on systems performance. We did some evaluation on different pooling type on overall accuracy as well.

References

[1] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

[2] Y-Lan Boureau, Francis R. Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *CVPR*, pages 2559–2566. IEEE Computer Society, 2010.

[3] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann Lecun. Ask the locals: multi-way local pooling for image recognition. In *ICCV'11 - The 13th International Conference on Computer Vision*, Barcelone, Spain, November 2011.

[4] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL*, 2010.

[5] Kuan-Ting Chen, Kuan-Hung Lin, Yin-Hsi Kuo, Yi-Lun Wu, and Winston H Hsu. Boosting image object retrieval and indexing by automatically discovered pseudo-objects. *Journal of Visual Communication and Image Representation*, 21(8):815–825, 2010.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7]**BONUS** Freezing early layers. When you look at the default setting and parameters we provide, you observe that learning rate is different for early layers and the layer we fine-tune and randomly initialize. One way to achieve fine-tuning is also freezing completely early layers and only training the last layer. You can achieve this by simply setting the learning rate for early layers to 0. You can report the accuracy you obtained.

[9] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011.

[10] Herve Jegou, Hedi Harzallah, and Cordelia Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[11] Yu-Gang Jiang, Jun Yang, Chong-Wah Ngo, and Alexander G Hauptmann. Representations of keypoint-based semantic concept detection: A comprehensive study. *IEEE Transactions on Multimedia*, 12(1):42–53, 2010.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[14] Marcin Marszaek and Cordelia Schmid. Spatial weighting for bag-of-features. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2118–2125. IEEE, 2006.

[15] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[16] Andrew M. Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y. Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1089–1096, USA, 2011. Omnipress.

[17] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions, 2015.

[21] Lei Wu, Steven CH Hoi, and Nenghai Yu. Semantics-preserving bag-of-words models and applications. *IEEE Transactions on Image Processing*, 19(7):1908–1920, 2010.

[22] Zhong Wu, Qifa Ke, Michael Isard, and Jian Sun. Bundling features for large scale partial-duplicate web image search. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 25–32. IEEE, 2009.

[23] Shiliang Zhang, Qi Tian, Gang Hua, Qingming Huang, and Wen Gao. Generating descriptive visual words and visual phrases for large-scale image applications. *IEEE Transactions on Image Processing*, 20(9):2664–2677, 2011.