CS485 Machine Learning for Computer Vision

Jaehan Jeong
20233933
fortes@kaist.ac.kr

Minhajur Rahman Chowdhury Mahim
20210753
minhaj@kaist.ac.kr

1. K-means Codebook

1.1. Vector Quantization Process

We effectively build the k-means codebook following the given instruction. We used SIFT to extract descriptors. As shown in the example cases in Figure 1, SIFT is applied on the grayscaled version of the input images and descriptors are returned. Using the training set, we randomly select 100k of them, which is later used for k-means clustering for centroids. During this process, the learned descriptors are applied and Bag-of-word histogram is built.

1.2. Vocabulary Size

Figure 2 illustrates our finding that the consumed time on building the vocabulary is proportional to the vocabulary size we choose. It seems to be logarithmicly expensive with respect to time. If the size chosen is too small, it will not generalize properly. On the other hand, larger vocabulary size will be computationally expensive. We present the Bag-of-Word (BoW) representation with the vocabulary size and examples with SIFT descriptors, $k = 100$ in Figure 1.

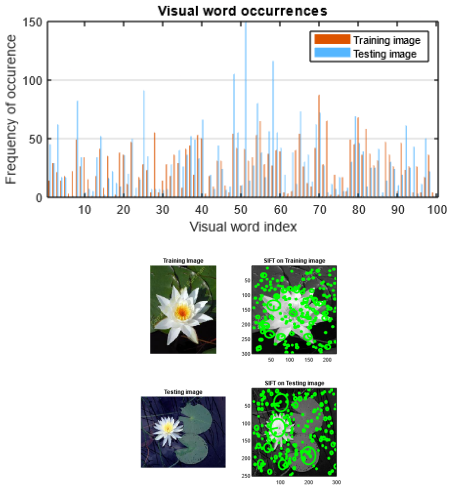

Figure 1. BoW Representation in Histograms using $K = 100$ and Example Cases

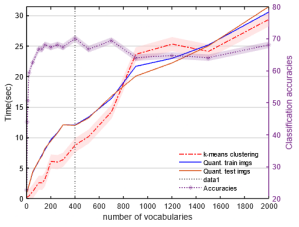

Figure 2. Vocabulary Time Complexity with respect to size

2. Random Forest Classifier

We train the Random Forest using the Bag-of-word representation of the training set. As the random forest has a variety of impactful hyperparameters, we will discuss their considerable effects on the model. The default parameters of Random Forest are stated in the Appendix.

2.1. Number of Trees, Maximum Tree Depth, Degree of Randomness

Figure 3 summarizes our experiment on varying number of trees, maximum tree depth and degree of randomness on Random Forest. As expected, the classification accuracy gradually increases as the number of trees is increased. The time complexity for training grows almost linearly with the respect to the tree count.

Coming to the maximum tree depth, the classification accuracy is compromised at both smaller values and larger values. It is because the random forest underfits the data because of being unable to do data splitting properly, whereas it may overfit if the leafs nodes possess fewer points. Also, the the time complexity for train and test show a exponential trend from depth = 18.

Degree of randomness is, in fact, the the number of splits. Figure 3 shows that setting the number to 4 gives the best classification accuracy.

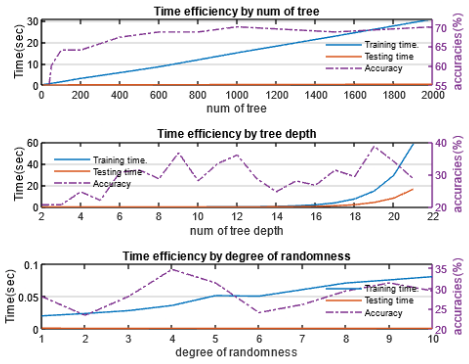

Figure 3. Performance of Random Forest with respect to number of trees, maximum tree depth & degree of randomness

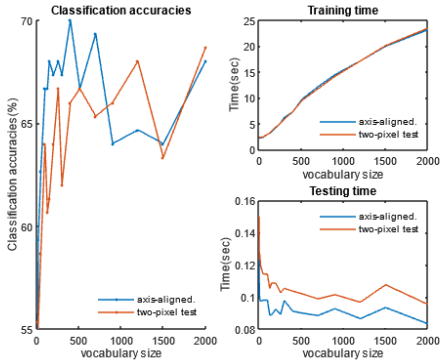

Figure 4. Performance Comparison with Weak Learners & Vocabulary Size

2.2. Weak Learners & Vocabulary Size

We aimed to experiment with varying vocabulary size from the BoW with two different weak-learners: axis-aligned test and two-pixel test. Figure 4 shows our results. Interestingly, although training time for both weak-learners with respect to vocabulary size is analogous, it takes longer time for testing when used the two-pixel test. However, the model gradually overfits when the vocabulary size becomes greater than around 500.

2.3. Optimal Configuration

After all analyses, we chose the number of trees = 300, maximum tree depth = 6, degree of randomness = 8 with the axis-aligned test as the hyperparameters of random forest while keeping the BoW vocabulary size = 400, and achieved 70% recognition accuracy. Figure 5 shows the confusion matrix for this model. We attempted with another version of the model after changing the vocabulary size to 700 and changing the weak learner to two-pixel test and gained 69.33% accuracy in classification, close to the results of the former.

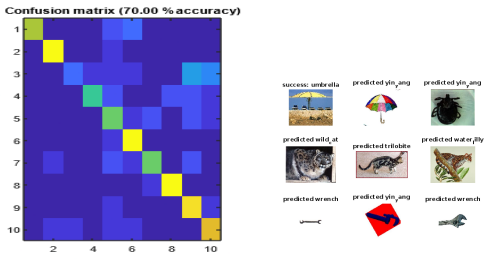

Figure 5. RF Classifier on k-Means Codebook: Confusion Matrix and Example Cases

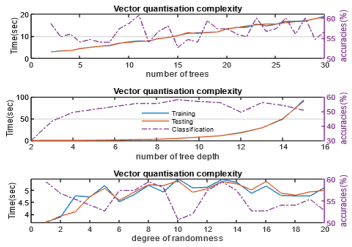

Figure 6. Vector Quantization Complexity for RF Codebook

3. Random Forest Codebook

3.1. Configuration

For the RF codebook implementation, we choose number of trees = 5, maximum tree depth = 9, and the degree of randomness = 4 after exploring several variations.

3.2. Vector Quantization Process

The difference between the k-means codebook and the RF codebook is that the RF codebook extracts discriminative information in the descriptor space. Unlike k-means, Random Forest codebook vocabulary size is not explicitly stated. Rather it has an upper bound of $K \leq num\_trees \times 2^{max\_depth-1}$. Figure 6 illustrates the results. We observe "zigzag" trend in accuracy, train and test time for changing degrees of freedom.

3.3. Evaluation & Comparison

We evaluate with the Random Forest built from RF Codebook and try with axis-aligned and two-pixel test variants. The classification accuracies are 59.33% & 57.33%. Both results are comparitively lower than the results found in 2.3.

4. Convolutional Neural Networks

We begin with 2 convolutional neural networks named SimpleCNN and SimpleCNN2, which are described in Figure 7 and Figure 8 with the parameters. The total number of trainable parameters in these architechture is 2000154 and

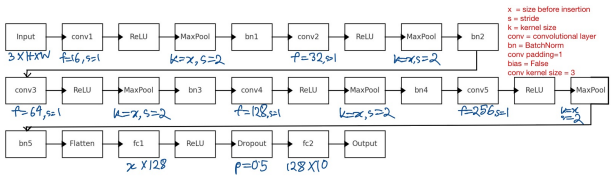

Figure 7. SimpleCNN Architecture

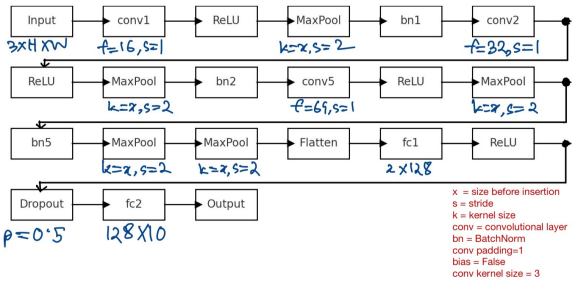

Figure 8. SimpleCNN2 Architecture

426522 respectively. Detailed parameter and hyperparameter considerations are provided in Appendix. We first compare these architectures with an already established network AlexNet [4] (trainable parameters = 46787978) and analyze the performance. Figure 9 shows that our models nearly converge during training within 15 epochs whereas Alexnet hardly converge even in 30 epochs. The reason is probably due to the fact that the our training sample is quite small compared to much bigger non-pretrained AlexNet with too many learnable parameters. Since AlexNet could not converge properly while training, we suspect that it suffered gradient vanishing/exploding. On the other hand, our models show almost similar performance to each other in terms of loss and accuracy in 15 epochs even though there is a significant difference in the number of learning parameters between them. We investigate further with our base models to determine the better one.

4.1. Normalization & Dropout Check

We continue with our SimpleCNN and SimpleCNN2 base models to run experiments with varying normalization methods and dropout regularization. Initially, our base models had the batch normalization in the normalization layers with dropout enabled in the first fully connected layer. We first do our experiment by disabling dropout while keeping batch normalization as the normalization method, and then change the normalization to group normalization and train and test twice with dropout enabled and disabled. In total, we have 4 variations. Figure 10 summarizes our findings. It is visible from the figures that on average, all the models with the dropout the ones without the dropout in testing evaluation, reaffirming the importance of regularization while training deep neural networks. In the next sections, we proceed with the SimpleCNN with the batch normalization and dropout enabled variant as it gives slightly better testing accuracy than the others till now.

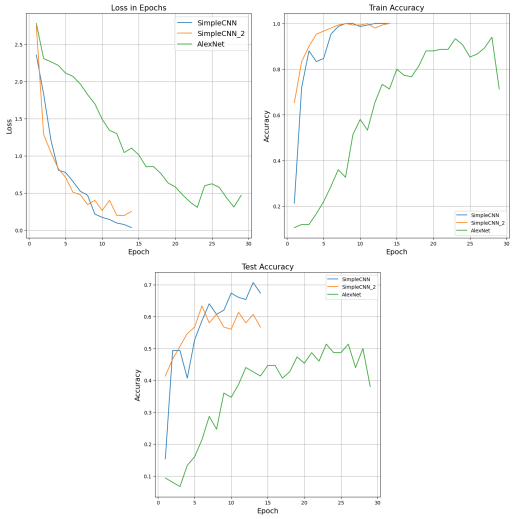

Figure 9. Performance Comparison between SimpleCNN, SimpleCNN2, AlexNet

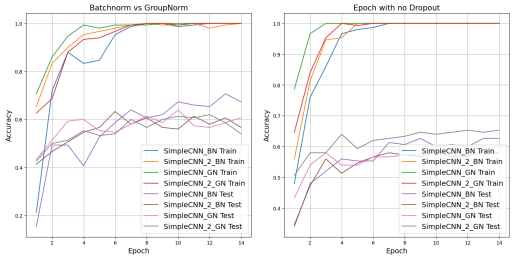

Figure 10. Comparison Between BatchNorm and GroupNorm. Left: With Dropout, Right: Without Dropout

4.2. Loss Functions

We compare between Softmax and Square Hinge Loss functions and analyze the loss and accuracy. For a multi-class classification, the square hinge loss is given by $max(0, 1 - z_y + z_j)^2$ [3] for each incorrect class j where z_y is the score of the true class. It has piecewise linear and quadratic components so, the gradient can change abruptly when the model's prediction crosses the margin threshold. The loss function analysis in Figure 11 confirms the "zigzagness" of square hinge loss as expected while the softmax loss is smooth. However, evaluation-wise we do not find much difference.

4.3. Learning Rate and Batch Size

Our results in Figure 12 suggests that learning rate of 0.0001 and batch size = 5 gives the best performance in-

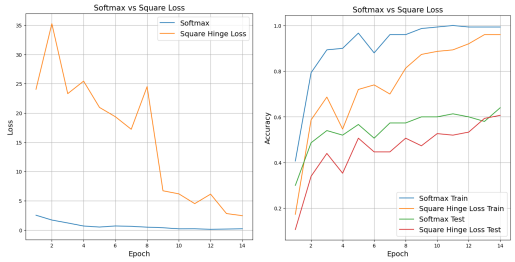

Figure 11. Loss Functions Comparison

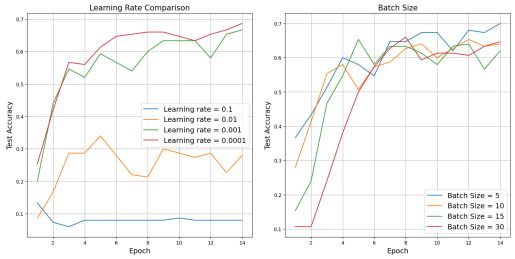

Figure 12. Learning Rate and Batch Size Comparison

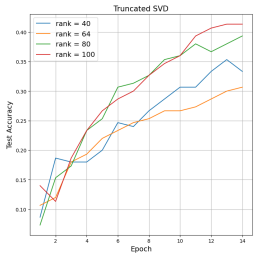

Figure 13. Truncated SVD Accuracy Tradeoff Comparision

dividually for our SimpleCNN model. The highest experimental learning rate is chosen to be 0.1, which shows underwhelming performance for our model, indicating suboptimal convergence. Too note, batch size = 5 with other hyperparameters under default settings in SimpleCNN provide the maximum recognition accuracy of 70%.

4.4. Truncated SVD

Following the work of Fast R-CNN [1], we experiment with out SimpleCNN model by truncating the weights of the first fully connected layer with different ranks of singular value decomposition (SVD) and show the results in Figure 13. As expected, lower ranked truncation provides lower performance. Even with the close to full rank, the evaluation accuracy barely surpasses the 40% bar.

4.5. Optimal Model and Finetuning a Pre-trained Model

After all analyses, we choose learning rate = 0.0001, batch size = 5, batch normalization as the normalization

Figure 14. SimpleCNN vs ResNet-50 Finetuning

method, softmax loss, and dropout enabled for our SimpleCNN model. We also finetune a pretrained model ResNet-50 [2] with our training images with the chosen final hyperparameters for our scratch model. We only modify the final fully connected layer of the model to fit our number of classes. We train and finetune these models respectively for 100 epochs and show the performance in Figure 14. We see that our model gets the classification accuracy of 70.7% while ResNet-50 achieves staggering 95.3% classification accuracy, the best performance among all the methods discussed in this paper. To note, our model shows a consistent non-improving accuracy even in 100 epochs. These observations reaffirm the preference of finetuning an existing model rather than training a new model from scratch when the training set is quite small.

5. Conclusion

Comparing with all the methods discussed in this paper, we recommend to use finetuned ResNet-50 for the classification task of the given Caltech101 dataset. The reason is due to the much smaller size of dataset. Given that we could perform data augmentation on the dataset, we might would get better performance in our scratch CNNs. However, training CNNs not only needs time, but also requires better training resource. In absence of good resource, building a model using K-means Codebook with Random Forest based classification instead is a good accuracy-resource trade-off deal.

References

[1] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 4

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4

[3] Tae-Kyun Kim. KAIST CS485: Machine Learning for Computer Vision Lectures. 3

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12,

page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 3

Appendix

Default Parameters for Training Trials for Random Forest

If not stated otherwise in the discussion,

Hyperparameters/Settings	Default Value
Vocabulary Size	400
Number of Trees	300
Degree of Randomness	8
Maximum Tree Depth	6
Weak Learner	Axis-aligned
Split Criteria	Information Gain

Table 1. Default Parameters for Random Forest

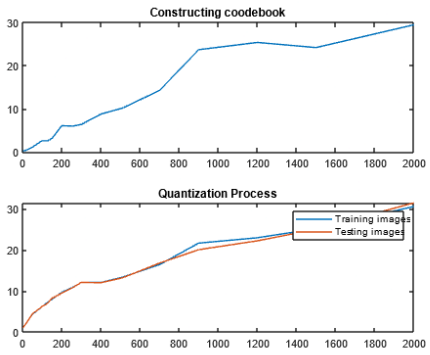

Figure 15. K-means Codebook Construction & Optmization

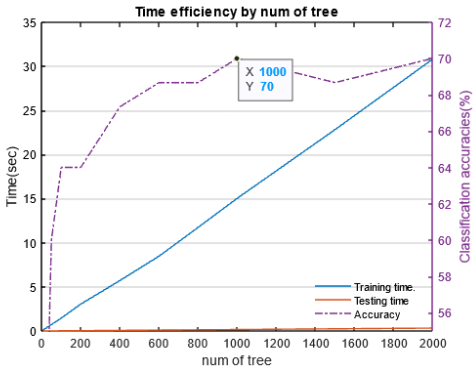

Figure 16. K-means Random Forest with respect to Number of Trees

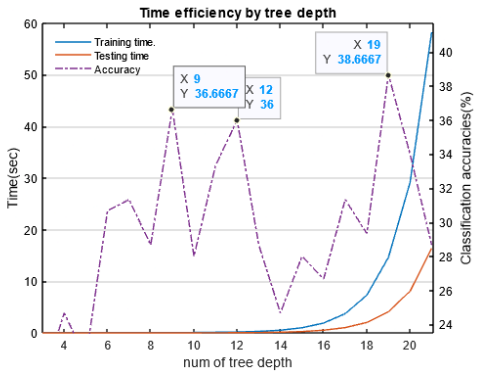

Figure 17. K-means Random Forest with respect to maximum tree depth

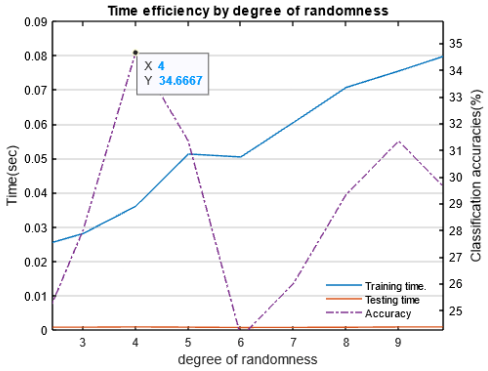

Figure 18. K-means Random Forest with respect to degree of randomness

Environment Control for Convolutional Neural Networks

All the tasks for this section have been performed in Google Collaboratory on NVIDIA V100. And to preserve resources, the epoch for all trials and models was set to 15 unless stated otherwise.

Default Parameters for Training Trials for Convolutional Neural Networks

If not stated otherwise in the discussion,

Hyperparameters/Settings	Default Value
Learning Rate	0.001
Batch Size	10
Normalization	BatchNorm
Criterion	Softmax
Optimization	Adam
Dropout	p = 0.5

Table 2. Default Parameters for Built CNNs