AML-CW-1
A hyperparameter selection for image classification performance

Youngbin Lee[1]

Abstract

This report provides experiments and discussions on four different machine learning models (KNN, SVM, Decision Tree, Random Forest) in image classification tasks on CIFAR-10 dataset.

1. Introduction

1.1. Image Classification

Image classification is a supervised learning problem. In this problem, datasets contain images and labels that are objects to identify in images. Machine learning models are trained to learn features of such images and after training, models can distinguish different image labels by predicting most probable labels from the test datasets.

It is well known that deep learning models such as convolutional neural networks can extract features from images well, but machine learning models still can learn features to predict image labels. Therefore, in this report, image classification tasks will be conducted with four different machine learning algorithms.

1.2. Hyperparameter Selection in Machine Learning Performance

Hyperparameters are parameters that users define to give machine learning models diverse variations. Hyperparamter optimization is one of main challenges in the field of machine learning. This is because it affects performance greatly.

One of the most common method of fining the best set of hyperparameters is to search for every possible combinations of hyperparameters by specifying certain ranges. Also, cross-validation makes hyperparameter search more reliable when training data is not sufficient. Therefore, in this paper, the best set of hyperparameters will be compared with cross-validation and validation on a separate validation set.

[1]Department of Industrial Engineering, UNIST. Correspondence to: Youngbin Lee <young@unist.ac.kr>.

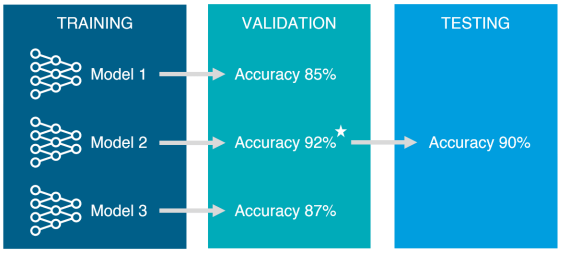

Figure 1. Validation on a separate validatation dataset

1.3. Problem Definition

In this report, experiments aim to find the best model and optimal set of hyperparameters that gives the best performance. Hyperparameters are examined by searching all possible combinations within the ranges for each parameter. Performances are measured by accuracy, which tells the percentage of all samples predicted correctly.

2. Experiments

All submissions must follow the specified format.

2.1. Data Description

CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes. There are 50000 training images and 10000 test images. Examples of classes include whale, bed, chair, bee, bus, tank, etc.

2.2. Hyperparameter Selection Strategies

In this paper, two strategies are used to find the optimal set of hyperparameters.

First strategy is to use a separate validation dataset to validate trained model. In this case, models are trained with training dataset with a set of hyperparameters. Then, trained models are evaluated by validation dataset. Optimal hyperparameters are determined to be the ones that have the highest accuracies when testing by validation data. Figure [1]

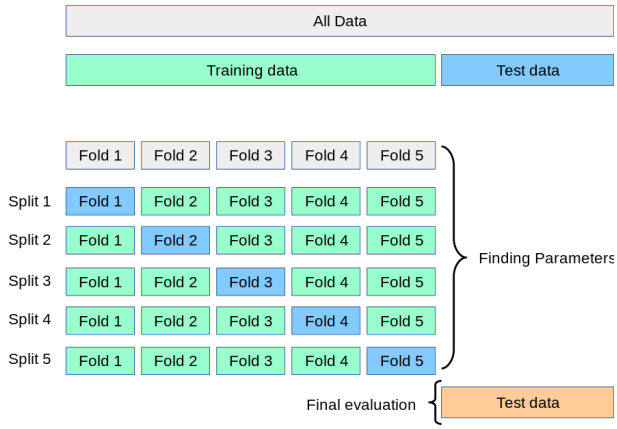

Figure 2. 5-fold Cross-validation

Second strategy is to perform k-fold cross validation. In this case, training dataset is divided into k-folds and the models are trained k times. At each iteration of k times, models are trained with training data without one fold out of k-folds that is selected to be validation set. After k iterations, the model's validation performances are averaged to get the final validation performance. Optimal hyperparameters are determined to be the ones that have the highest average accuracy. Figure [2]

In both of two cases, the optimal hyperparameters are finally used for a final test with a test dataset to get an actual performace.

2.3. Machine Learning Models

2.3.1. Runtime and Accuracy

To see how runtime and accuracy change according to the number of training dataset, models are trained with different number of data. In this experiment, wandb.ai is used for tracking runtime and accuracy. In Figure [3], the range of train dataset and hyperparameters used are speficied for each models.

It is expected that accuracies go up as the number of train data grows. However, in results in [Figure 4-6], the unlikely cases happened at some models where the accuracies go down at larger number of train data. This is because the range of the number of train data was too small due to the resource and time limit. It seems that if the range was wider with larger numbers of train data, accuracies would be going up as the number of train data grows.

2.3.2. Hyperparameter Selection

To find the best set of hyperparameters, different kinds of hyperparameters are searched for each models. Parameters used and their ranges are specified in Figure [7]. In the figure, numbers of data for training, validation, testing are shown as well.

Optimal sets of hyperparameters found are listed with their test set accuracies are listed in Figure [8]. Methods in the figure indicates (1) Default: No hyperparameter searching was conducted (2) Validation: Hyperparameters are determined by validation on a separate validation set (3) Cross-validation: Hyperparameters are determined by cross-validation.

It was expected that hyperparameter sets found from cross-validation would have the highest test accuracies. However, in my results, hyperparameters did not change from default because not much of parameters were searched due to resource and time limit. For example, KNN and SVM models had to be trained with only 1000 train set and tree-based models had to be trained with only 500 train data.

Tree-based models have particularly low accuracies, which may have been caused by a small number of features used. In other words, when greedy searching to find the best feature and the best split thresholds in tree algorithms, only 10 features were searched to produce results in time.

If the experiments were conducted with sufficient amount of training dataset and sufficient range of hyperparameters to be searched, results would have been more reliable. For example, hyperparameters may have been very different from default parameters and hyperparameters selected from cross-validation would have had a lot higher accuracies because it brings a robust performance.

The visualization of hyperparameter combinations in [Figure 9-17, Appendix] shows some hyperparameters are better than the others. For example, KNN models worked better with L1 distance metric than the others. But the effect of varying the number of neighbors was negligible.

Non-linear SVM model shows that the key determinant of model performance is the value of gamma, where smaller values brought better accuracies.

Decision Tree shows a clear finding that minimum number of samples to split should be smaller. However, when using a large number of data and this parameter is too small, tree models can be overfitted. Therefore, a sufficient amount of train dataset and cross-validation could have found the optimal number for this parameter - small enough, but not too small to overfit.

3. Conclusion

In this report, four different machine learning models (KNN, SVM, Decision Tree, Random Forest) are conducted to perform image classification tasks. To find the best model, a range of hyperparameters are tested by greedy search. The

Model	Train set range	Test set	Hyperparameters	Value
KNN	[1000, 2000, 3000, 4000]	100	N_neighbors Distance	3 L1
Linear SVM	[1000, 2000, 3000, 4000]	100	C	1
Non-linear SVM	[1000, 2000, 3000, 4000]	100	C Gamma	1 0.2857
Decision Tree	[1000, 1500, 2000]	100	Min_samples_split Max_depth	100 3
Random Forest	[500, 600, 700]	50	Min_samples_split Max_depth	10 3

Figure 3. Train dataset range

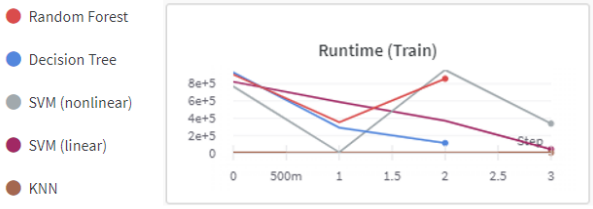

Figure 4. Runtime (train)

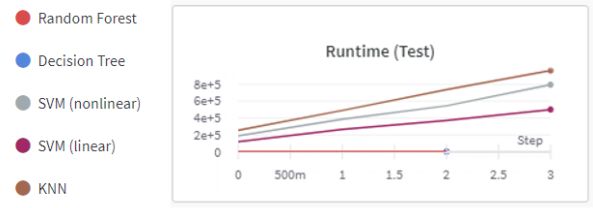

Figure 5. Runtime (test)

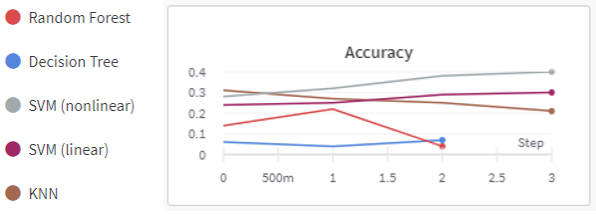

Figure 6. Accuracy

results show that SVM model worked best with parameters C=1 for linear SVM model, and C=30 and Gamma=0.0001 for nonlinear SVM model.

However, the experiments were not fair for tree-based models because the number of train dataset was smaller than the other models due to the resource and time limit. Also, for all of algorithms, it would have been better if the search range was wider and a sufficient amount of data was used to train.

Model	Hyperparameter	Range	Train set	Valid set	Test set
KNN	N_neighbors Distance	[3, 5, 7] [L1, L2, minkowski]	1000	100	100
Linear SVM	C	[10, 30, 100]	1000	100	100
Non-linear SVM	C Gamma	[10, 30, 100] [0.01, 0.001, 0.0001]	1000	100	100
Decision Tree	Min_samples_split Max_depth	[100, 150] [2, 3]	500	50	50
Random Forest	Min_samples_split Max_depth	[100, 150] [2,3]	500	50	50

Figure 7. Hyperparameter search ranges

Model	Method	N_neighbors	Distance	Test Acc
KNN	Default	3	L1	**0.31**
	Validation	3	L1	0.16
	Cross-validation	3	L1	0.16

Model	Method	C	Test Acc
Linear SVM	Default	1	0.24
	Validation	1	**0.33**
	Cross-validation	1	**0.33**

Model	Method	C	Gamma	Test Acc
Non-linear SVM	Default	1	0.2857	0.28
	Validation	100	0.0001	0.31
	Cross-validation	30	0.0001	**0.33**

Model	Method	Min_samples_split	Max_depth	Test Acc
Decision Tree	Default	100	3	0.06
	Validation	100	3	**0.16**
	Cross-validation	100	2	0.08

Model	Method	Min_samples_split	Max_depth	Test Acc
Random Forest	Default	100	2	0.02
	Validation	100	2	0.08
	Cross-validation	100	2	**0.12**

Figure 8. Optimal hyperparameters

3. Appendix

Visualization of different hyperparameter sets and corresponding accuracies are shown in [Figure 9-18]

4. References

https://www.cs.toronto.edu/ kriz/cifar.html

Claesen, M., De Moor, B. (2015). Hyperparameter search in machine learning.arXiv preprint arXiv:1502.02127.

https://developers.google.com/machine-learning/practica/image-classification?hl=ko

https://www.tensorflow.org/datasets/catalog/fashion$_mnist$

https://paperswithcode.com/dataset/caltech-256

https://scikit-learn.org/stable/modules/cross$_validation.html$

https://modern-manual.tistory.com/19


```
Maximum Accuracy: 0.2000
Optimum n_neighbors: 3.000000
Optimum distance: 1.000000
```

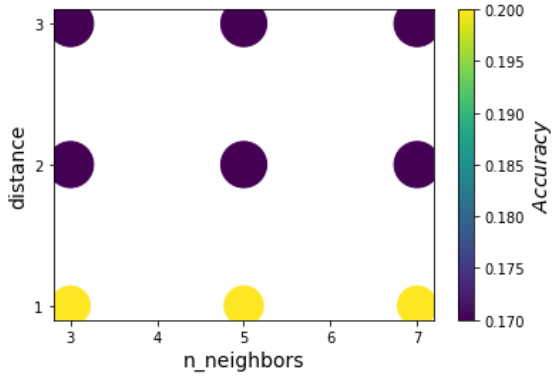

Figure 9. KNN with validation

```
Maximum Accuracy: 0.1920
Optimum n_neighbors: 3.000000
Optimum distance: 1.000000
```

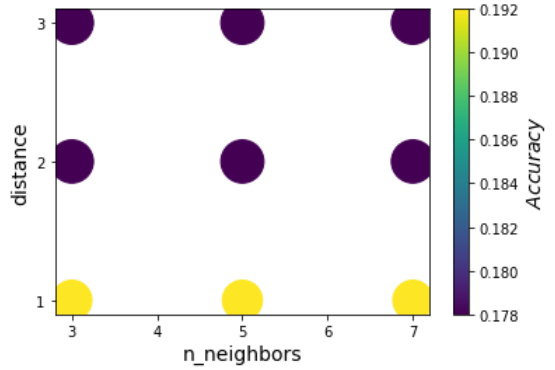

Figure 10. KNN with cross-validation

```
Maximum Accuracy: 0.2500
Optimum C: 1.000000
Optimum kernel: 1.000000
```

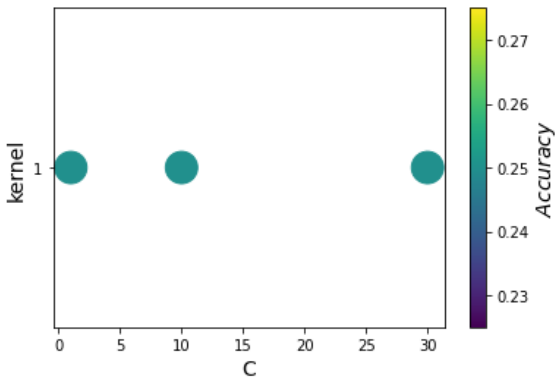

Figure 11. linear SVM with validation

```
Maximum Accuracy: 0.2360
Optimum C: 1.000000
Optimum kernel: 1.000000
```

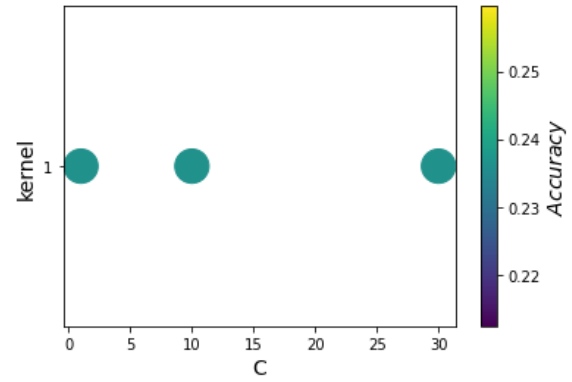

Figure 12. linear SVM with cross-validation

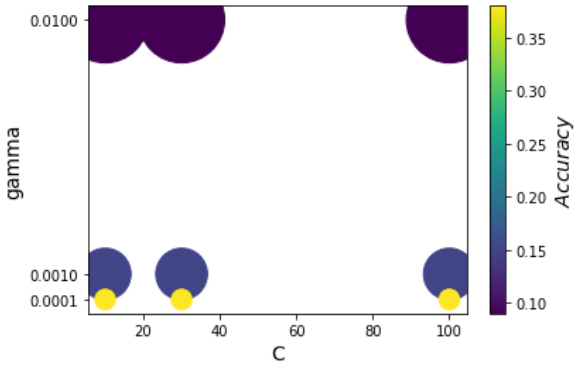

Figure 13. nonlinear SVM with validation

```
Maximum Accuracy: 0.2530
Optimum C: 30.000000
Optimum gamma: 0.000100
```

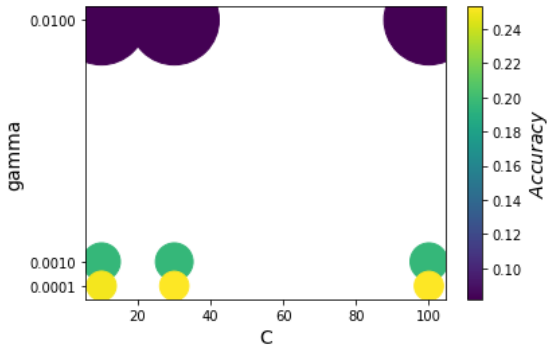

Figure 14. nonlinear SVM with cross-validation

```
Maximum Accuracy: 0.2000
Optimum min_samples_split: 100.000000
Optimum max_depth: 3.000000

C:\Users\YOUNGBIN\AppData\Local\Temp\ipykernel_29620
  size_list=30/(size_list)**2
```

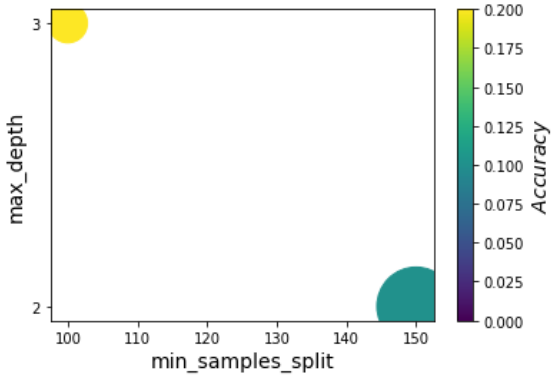

Figure 15. Decision Tree with validation

```
Maximum Accuracy: 0.1280
Optimum min_samples_split: 100.000000
Optimum max_depth: 2.000000
```

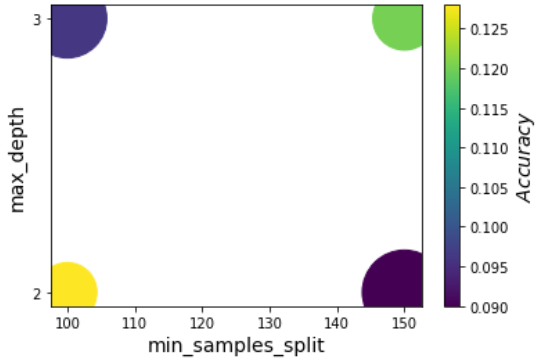

Figure 16. Decision Tree with cross-validation

```
Maximum Accuracy: 0.2000
Optimum min_samples_split: 100.000000
Optimum max_depth: 2.000000
```

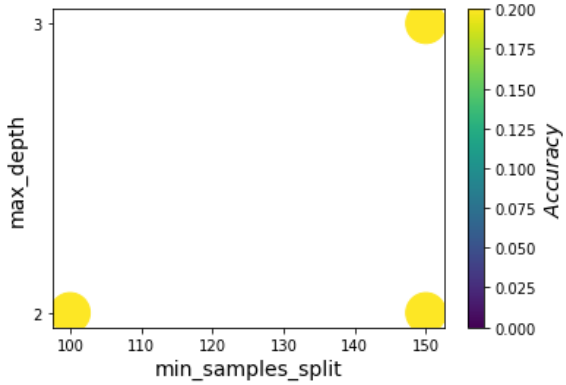

Figure 17. Random Forest with validation

```
Maximum Accuracy: 0.1200
Optimum min_samples_split: 100.000000
Optimum max_depth: 2.000000
```

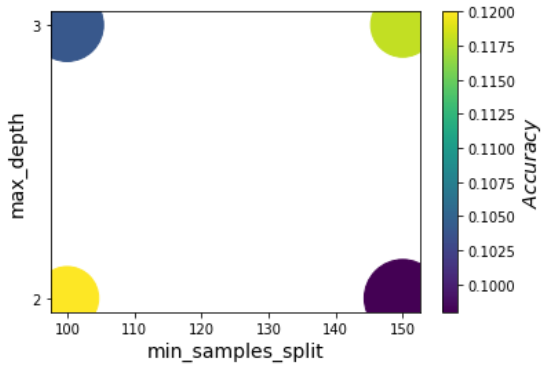

Figure 18. Random Forest with cross-validation