Deep Learning Model Performance Evaluation System for Target Detection

Ayoung Kim1, Minseo Kim1,\*, HyeonKyu Kim 1,\*, EunKyu Seok 1,\*, and Yaejin Lee 1,\*

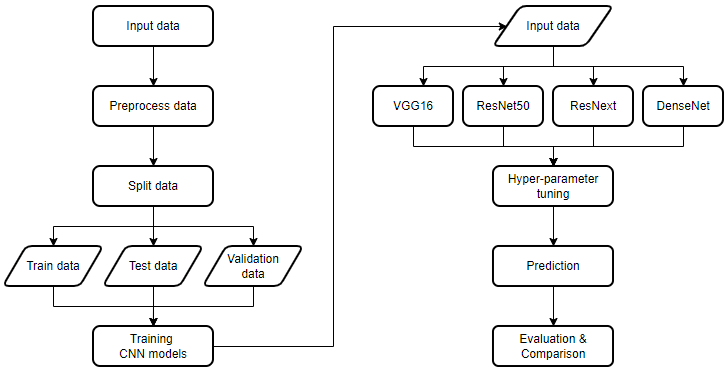
1 Information Convergence, Kwangwoon University, Korea

**Abstract:** Starting with LeNet, deep learning technology has been continuously developed over time. However, during the process of increasing the deep neural network model’s accuracy and performance, the number of layers built into the deep learning model increased, and this led to the increasing number of parameters. Particularly, hyper-parameters, parameters that cannot be trained by a neural network, require a lot of experience and time to optimize. In this work, we introduce a new system that aids users to easily choose a deep learning model and hyper-parameters optimized and performs best to the user’s input dataset. The system automatically compares different convolutional network models that had been optimized with suitable hyper-parameters. Finally, the system shows the user best-performed model with optimized hyper-parameters.

1. Introduction

In 1989, Lecun, Yann presented LeNet, the first neural network model. LeNet had several deficiencies including the gradient vanishing problem and XOR problem, however, by solving those drawbacks, deep neural network models had been improved persistently over the last decades. Of its improved performance and accuracy, deep neural network models had been used in various ways including image processing and image classification (CNN). Despite the continuous development of deep learning network, the process of optimizing hyper-parameters for input data is a tricky task that should be kept studied. Unlike other parameters that can be earned by training neural network model, hyper-parameters should be decided by humans and cannot be carried out automatically. Moreover, the final effect of hyper-parameters on the final performance of neural network models is considerable. Thus, the necessity of a system that automatically finds the best-performing neural network model with optimized hyper-parameters for specific imported data is emerging.

**2. Methodology**



**Figure 1.** The flow chart of system.

Hyper-parameters are parameters that cannot be earned by the model’s training; therefore, humans must intervene in order to find the right optimized value for input data. Thus, our system allows this complicated task to be progressed completely automatically through several experiments from candidate values given by the user.

As Figure 1 shows, first, the user inputs image data to the system with classes that the dataset has and some candidate values that human wants to test out. Table1 shows the default candidate hyper-parameter values. Then, the system splits the data into train data, test data, and validation data in the proportion of 4:4:2. Train data gets imported into each CNN model. In this system, we test VGG16, ResNet50, ResNext, and DenseNet. The system imports the pre-trained CNN models for both better accuracy and time performance. Next, each model tests various hyperparameters with candidate values that the user gave and gets evaluated respectively by three perspectives; accuracy, f1-score, and inference time. Lastly, the system shows the user the result in a separate evaluation index. The user can see the result of the best-performed models.

**Table 1.** Default values of each hyper-parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Candidate 1** | **Candidate 2** | **Candidate 3** | **Candidate 4** | **Candidate 5** |
| drop-out rate | 0.2 | 0.3 | 0.5 | data | data |
| learning rate | 0.01 | 0.005 | 0.001 | 0.0005 | 0.0001 |
| optimizer | SGD | ADAM |  |  |  |

The candidate CNN models that our system experiments with are VGG16, DenseNet, ResNet50, and ResNext. Our team had considered several various CNN models since the final results can be changed by which model our system compares. Lastly, we decided on those four models from a research paper, “A Transfer Learning Evaluation of Deep Neural Networks for Image Classification”. The paper explains different CNN models’ structures and characteristics, and it also compares their performance in several ways. Those four models we selected were models that were highly performed in the paper’s evaluation.

3. Tool Support

The system consists of four different classes. First ResizeImage class is a module that allows the user to input image data and several parameters like image classification classes or candidate hyperparameters. Next, the Comparison System class is for loading different CNN models and evaluating them respectively. Also, there are four different classes for each CNN model (myDenseNet Class, myResNet Class, myResNext Class, myVGG Class) that process fine-tuning task and optimize their hyper-parameters through experimenting every candidate hyper-parameter value that the user gave. Lastly, show\_graph class visualizes the result of the evaluation to a user.

**3.1 ResizeImage Class**

init()

* parameters: input dataset path, size of the image
* loads input data from the path and does some preprocessing such as flattening image data

resizing()

* resize images and change data into the format of CNN model input

**3.2 Comparison System Class**

init()

* set the default value of hyper-parameters

datasplit()

* splits input data into train, test, and valid data with a proportion of 4:4:2

setSystem()

* loads each pre-trained CNN model

runSystem()

* preprocess data and train models with train data

getResult()

* evaluates the model with three evaluation index (accuracy, f1-score, inference time), and deduce the ranking result

**3.3 myVGG19 Class, myDenseNet Class, myResNet Class, myResNext Class**

preprocessing module

* preprocess image data for each type of CNN model’s input type

finetuning module (model\_vgg16(), model\_densenet(), modelResnet50(), model\_resnext())

* optimize hyperparameters through several experiments with given candidate values

predictResult()

* predicts test data and return evaluation score for each model and each hyper-parameter value set respectively

**3.4 show\_graph Class**

show\_accuracy()

* show the best-performing CNN model’s flow of accuracy in the training phase

show\_loss()

* show the best-performing CNN model’s flow of loss in the training phase

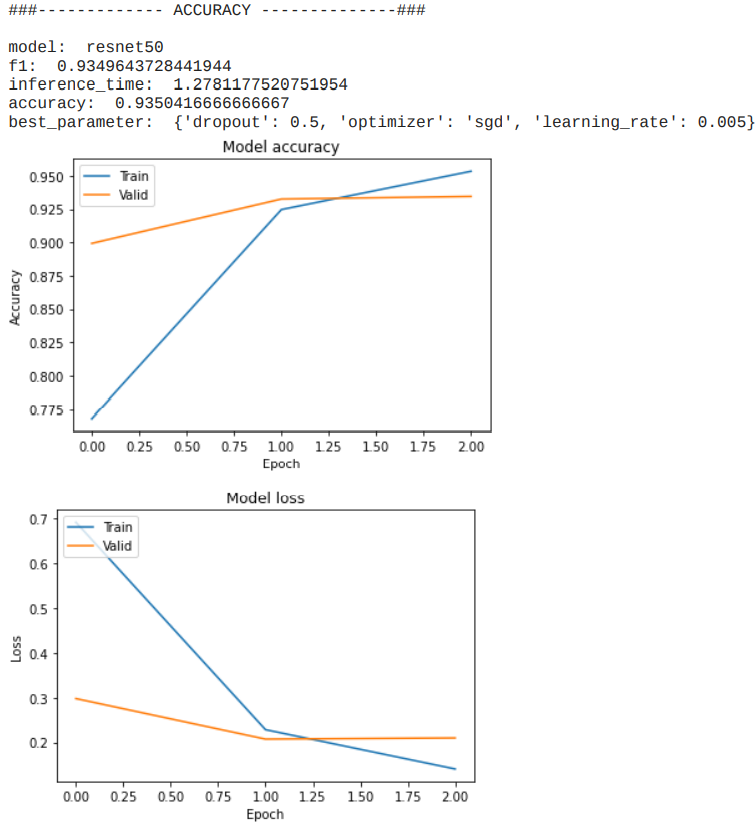
show\_result()

* show the best-performing CNN model’s f1-score and inference time

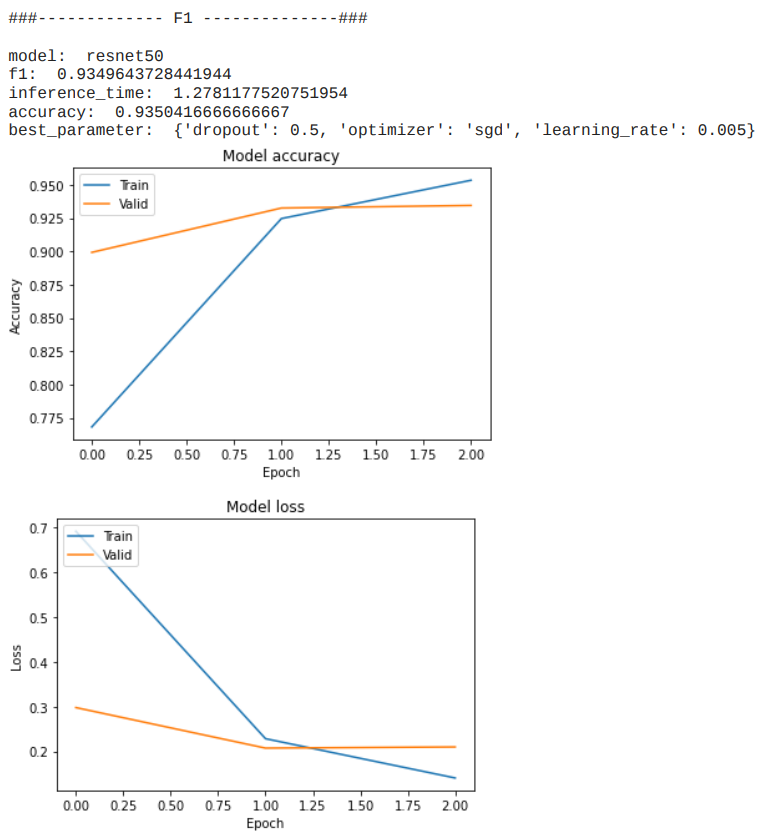
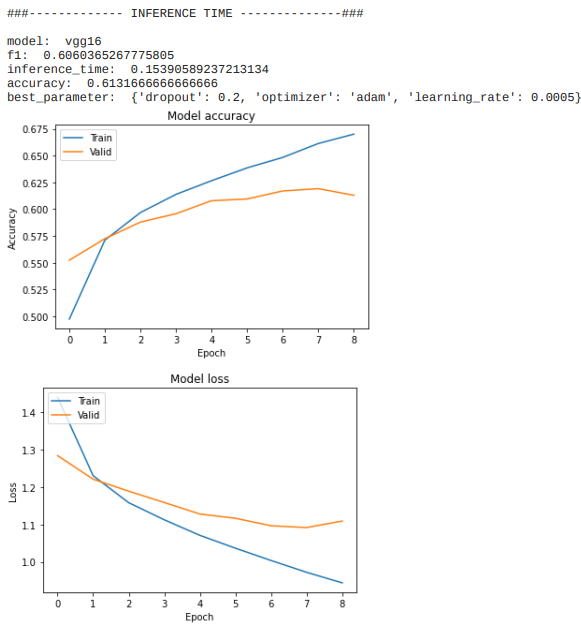
4. Results

With the methodology outlined in Sec. 2, Sec.3, we applied this system to CIFAR-10 dataset. Since our candidate CNN models were pre-trained with ImageNet dataset, we judged that CIFAR-10 dataset is suitable for a case study. Because of the limit on our study environment, we used batch size 64, and 10 epochs.

The following figures are the result of the CIFAR-10 case study. The best performed model in accuracy and f1-score was ResNet50 (hyper-parameter: drop-out rate=0.5, learning rate=0.005, optimizer=SGD) (Figure2, Figure3). From the perspective of inference time, the best-performed model was VGG16 (hyper-parameter: drop-out rate=0.2, learning rate=0.0005, optimizer= ADAM) (Figure4).



**Figure 2.** The performance of best-performed model in accuracy



**Figure 3.** The performance of best-performed model

in f1-score

**Figure 4.** The performance of best-performed model in inference time

5. Conclusions

This paper introduced a new system that compares different CNN models that have been optimized by specific input data without any human intervention in between. As a result, the system offers the fastest, perfectly target detecting model with optimized hyper-parameters. Thus, this system can be used in any way that involves target detecting tasks including the car license plate identification systems or in the national security center. Moreover, our system compares different CNN models from three different perspectives; accuracy, f1-score, and inference time. However, additional performance measures and evaluation indexes can be implemented and tested to enhance the performance of this system.

References

* *벤자민 플랜치 외 1명, “실전! 텐서플로 2를 활용한 딥러닝 컴퓨터 비전”, 위키북스*
* *오일석, “컴퓨터 비전”, 한빛아카데미*
* *Abou Baker, N.; Zengeler, N.; Handmann, U. A Transfer Learning Evaluation of Deep Neural Networks for Image Classification. Mach. Learn. Knowl. Extr. 2022, 4, 22–4.*
* *Canziani, A.; Paszke, A.; Culurciello, E. An Analysis of Deep Neural Network Models for Practical Applications. arXiv 2017, arXiv:1605.07678.*
* *Chen, L.; Li, S.; Bai, Q.; Yang, J.; Jiang, S.; Miao, Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. Remote Sens. 2021, 13, 4712.*
* *Fink, M. Object Classification from a Single Example Utilizing Class Relevance Metrics. In Advances in Neural Information Processing Systems; Saul, L., Weiss, Y., Bottou, L., Eds.; MIT Press: Cambridge, MA, USA, 2005; Volume 17.*
* *Gibb Spencer, Hung Manh La, and Sushil Louis, S. (2018). A genetic algorithm for convolutional network structure optimization for concrete crack detection. 2018 IEEE Congress on Evolutionary Computation (CEC), 1-8. IEEE.*
* *He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. arXiv 2015, arXiv:1512.03385.*
* *Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. In ICPR, pages 3791–3798, 2020. 5.*
* *Kawaguchi, K.; Huang, J.; Kaelbling, L.P. Effect of Depth and Width on Local Minima in Deep Learning. Neural Comput. 2019, 31, 1462–1498. [CrossRef]*
* *Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A survey of the recent architectures of deep convolutional neural networks. Artif. Intell. Rev. 2020, 53, 5455–5516.*
* *Nitish Srivastava, Geoffrey Hinton , Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.*
* *Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Adv. Neural Inf. Process. Syst. 2019, 32, 8024–8035.*
* *Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 2014, arXiv:1409.1556.*
* *Zagoruyko, S.; Komodakis, N. Wide Residual Networks. arXiv 2017, arXiv:1605.07146.*
* *Zaheer, R.; Shaziya, H. A Study of the Optimization Algorithms in Deep Learning. In Proceedings of the 2019 Third International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 10–11 January 2019; pp. 536–539.*