Automatic Deep Neural Network Hyper-Parameter Optimization for Maize Disease Detection

Subodh Bansal\textsuperscript{1} and Anuj Kumar\textsuperscript{2}
\textsuperscript{1} University Institute of Engineering and Technology, Panjab University, Chandigarh, India
\textsuperscript{2} Department of Computer Science and Applications, Panjab University, Chandigarh, India
subodhbansal991@gmail.com

Abstract. Deep Convolutional Neural Networks (DCNNs) have proved to be very useful for image classification. These need to be optimized for the dataset in hand, for its optimal use. DCNNs have many attached hyper parameters. These hyper-parameters are fine-tuned for optimizing the DCNN model. The present research uses the Bayesian Optimization technique for fine-tuning the hyper-parameters for AlexNet DCNN model. The model is trained and tested on Maize (corn) disease sub-dataset of the Plant Village dataset. The trained DCNN model with optimized parameters achieved the accuracy of 96.05%.

Keywords. Deep Learning, Convolutional Neural Networks; Transfer Learning, Computer Vision, Image Classification, Pattern Recognition, Bayesian Optimization.

1. Introduction

Deep Learning (DL) uses multi-layered artificial neural networks (ANN) for classification. An ANN tries to mimic a human brain’s neural connections for the development of the classification model [1]. A simplistic model of ANN is shown in figure 1. Every layer of the ANN has multiple nodes. Nodes of each layer are connected to those of the next and the previous layers through connections known as links. The strength of each link is determined by its weight and bais value. The first layer of an ANN is called the input layer, the last layer is called as the output layer and all layers in between are known as hidden layers. When different kinds of layers are stacked on top of each other, the developed structure is known as a DL Model.

Figure 1. Basic structure of neural network
A DL model uses Back Propagation Learning method for training. In this, annotated data is passed through a DL model and the model output is computed. In case the obtained output is not coherent with the annotations, the data is passed through the model in reverse, adjusting the weights and biases of each link to optimize the model according to the annotations. Thus optimizing the DL model for the dataset [2].

In back propagation learning, the degree to which the model adapts itself to the data is very crucial. If the model is over optimized on the training dataset, it will not able to classify similar unknown data with variations. This is known as the problem of over-fitting. On the other hand, if the model is under-optimized, then it will be able to classify the unknown data but will internally blur the boundaries between different classes, in which the data has to be classified. This is known as under-fitting. Both over-fitting and under-fitting is bad for a model as the accuracy of the model diminishes in both the cases.

The degree of adaptation of the model on the data can be varied using various hyper-parameters such as mini-batch size [3], number of training iterations, learning rate [4], gamma and momentum [5]. The DL models are black boxes and there is no efficient mathematical method yet developed for finding the co-relation between the values of hyper parameters and model optimization. Hence, to maximize the identification capability of the model, researchers tend to mix and match multiple values of different hyper-parameters to get the optimal values.

If the sample space is small, then the range of values for all hyper-parameters can be tested at certain pre-determined intervals, this method is known as grid search. Here, the number of experimentation runs increases exponentially by adding every new hyper-parameter. For example, if one has to test 10 values for each hyper parameter and try to optimize three of them, then the total number of tests needed to be run = 10x10x10 = 1000, if one experimentation run takes 30 minutes to complete, then the experimentation will take a minimum of 500 hours or 20 days to complete. This method will most likely find the optimal solution, but is quite expensive, especially for large data having high dimensionality as found in the real world scenarios. [6]

The Second technique used by the scientists is random search. In this, half or one-fourth of the grid points are explored and the best finds are supposed to be the actual best cases. The method does not promise to give the best results, but is computationally less expensive as compared to grid search. [7]

The technique explored in the present experimentation is Bayesian optimization. It takes the best of the two worlds. It tries to make intelligent guess for choosing the next random sample point based on prior information (points explored). This method also does not promise to find the most optimal result. But, is able to find pretty good results with a fraction of the computational power and time required as compared to grid search. [8]

The methods and materials in this experimentation, the obtained results and the drawn conclusions are discussed in the following sections.

2. Materials and Methods
Deep Convolutional Neural Networks (DCNN) has proved to be very helpful in the detection/classification of image data, the same was considered for this experimentation. The DCNN considered for this experiment is AlexNet [9], winner of the ILSVRC 2012 [10].

In the experimentation, firstly the dataset is normalized to standard 256 x 256 pixels. Then it is augmented to avoid over-fitting. Further, the images are divided into train and test sets in the ratio of 80:20. Then, the DCNN is trained, and the training is optimized using the Bayesian Optimization technique. Finally, the optimized results are plotted and documented.
The experimentation is carried out on an Ubuntu 18.04 machine with Nvidia GTX 1080 Ti GPU. The code is written in python language using PyCharm IDE [11]. The deep learning library used for the implementation of DCNN is Caffe [12].

2.1 Dataset
Maize (corn) sub-dataset of the Plant Village [13] dataset, widely used for the field of crop disease detection is used in the present experiment. The dataset consists of four classes as illustrated in table 1. Further, a sample image of each is shown in figure 2.

| Class Id | Condition     | Disease Name         | Number of Images |
|----------|---------------|----------------------|------------------|
| 0        | Healthy       | -                    | 1162             |
| 1        | Diseased      | Gray Leaf Spot       | 513              |
| 2        | Diseased      | Northern Corn Leaf Blight | 1192       |
| 3        | Diseased      | Common Rust          | 985              |

Figure 2. Sample images of maize crop leaf from plant village dataset of (a) gray leaf spot, (b) common rust, (c) northern leaf blight and (d) healthy leaves

2.2 Normalization
Normalization is the process of changing the pixel intensity and dimensions of the input images so that they can be used in the steps ahead. For normalization, the images in the dataset are squared and resized to 256x256 pixel size. The size is chosen as it is one of the standard sizes and is very near to the input image size needed by the AlexNet [7] DCNN model (227x227 pixels) used for this experimentation.

2.3 Augmentation
Augmentation is the technique of artificially increasing the size of the dataset and adding variations to it. The images of the dataset are transitioned using various transformation functions such as zoom, rotation, flipping, cropping and brightness alteration, to mimic various scenarios in which the image may be captured in the real-world scenario. It also helps in keeping the problem of over-fitting in check [14], [15]. Various augmentation techniques used in this experiment are illustrated in table 2, increasing the size of the actual dataset of 3852 images to 77,040 images.

| S. No. | Transformation | Description                  |
|--------|----------------|------------------------------|
| 1      | Rotate 0       | Original Image               |
| 2      | Rotate 90      | Original Image rotated by 900|
| 3      | Rotate 180     | Original Image rotated by 1800|
| 4      | Rotate 270     | Original Image rotated by 2700|
Further, the images of the dataset are split into training and validation sub-datasets. The training set is used for DCNN back propagation learning and the validation set is used for evaluation of the effectiveness of the training technique. The experiment utilized the 80:20 random split technique, i.e. the images of each class are divided into 80:20 ratio, the images in each sub-dataset are chosen at random. Thus, out of 77,040 total images, 61,632 images are used for training and the remaining 15,408 images serve as the validation set.

Table 3. AlexNet DCNN Model

| Layer Number | Layer Type | Feature Map Size | Kernel Size | Stride | Activation Function |
|--------------|------------|------------------|-------------|--------|--------------------|
| Input        | Image      | 227 x 227 x 3    | -           | -      | -                  |
| 1            | Conv 96    | 55 x 55 x 96     | 11 x 11     | 4      | Relu               |
|              | MPL 96     | 27 x 27 x 96     | 3 x 3       | 2      | Relu               |
| 2            | Conv 256   | 27 x 27 x 256    | 5 x 5       | 1      | Relu               |
|              | MPL 256    | 13 x 13 x 256    | 3 x 3       | 2      | Relu               |
| 3            | Conv 384   | 13 x 13 x 384    | 3 x 3       | 1      | Relu               |
|              | MPL 384    | 13 x 13 x 384    | 3 x 3       | 1      | Relu               |
| 4            | Conv 256   | 13 x 13 x 256    | 3 x 3       | 1      | Relu               |
|              | MPL 256    | 6 x 6 x 256      | 3 x 3       | 2      | Relu               |
| 6            | FCL 9216   | -                | -           | -      | Relu               |
| 7            | FCL 4096   | -                | -           | -      | Relu               |
| 8            | FCL 4096   | -                | -           | -      | Relu               |
| Output       | FCL 4      | -                | -           | -      | Softmax            |

*Abbreviations used in table 3: Convolution Layer (Conv), Max-Pooling Layer (MPL) and Fully connection Layer (FCL).

2.5 DCNN Model (AlexNet)

The AlexNet DCNN model is made up of eight layers, containing five Convolution and three Fully Connected layers. The input image size accepted by the model is 227 x 227 pixels RGB image. The model architecture is tabulated in table 3.

AlexNet utilizes some clever techniques to improve its performance:

i. Time complexity: the cost-effective ReLU activation function is used instead of tanh function to increase the training and testing speeds, thus reducing time complexity.

ii. Space complexity: max-pooling layers for dimension reduction.
iii. Over-fitting: it uses max-pooling and dropout layers to keep the problem of over-fitting in check, enhancing feature selection.

2.6 Hyper-Parameter Tuning

Hyper-parameter tuning or optimization is determining the best set of hyper-parameters for the DCNN model training to fetch the best results. In this experimentation, it is achieved by utilizing the Bayesian Optimization (BO) approach.

Some of the hyper-parameters are fixed, while others are tuned using the technique. The fixed ones are tabulated in table 4. The parameters fine-tuned using the BO approach along with the sample space (min and max values) are tabulated in table 5.

| Table 4. Fixed Hyper-Parameters |
|--------------------------------|
| Hyper-parameter | Value |
| Epoch Size | 61,632 |
| Epochs | 1 |
| Batch Size | 32 |
| Batches / Epoch | 1926 |
| Total Batches | 1926 |
| Learning Rate Decrease Policy | Step |
| Step Size | 192 |

| Table 5. Tuned Hyper-Parameters |
|--------------------------------|
| Hyper-parameter | Min-value | Max-value |
| Initial learing rate | 10-2 | 10-5 |
| Gamma (γ) | 0.1 | 0.9 |
| Momentum (µ) | 0.5 | 0.99 |

For tuning the hyper-parameters listed in table 5, initially, ten value trios of the random values of the hyper-parameters are taken. Training and validation of the DCNN is carried out to fetch the corresponding model accuracies. Then the input-output quads were fed to the Bayesian Optimization (BO) function. The BO function is run for 53 iterations. The dump of the same is listed in table 6.

| Table 6. Bayesian Optimization Dump |
|------------------------------------|
| Test run number | Initial learing rate | Gamma (γ) | Momentum (µ) | Accuracy (%) |
| 1 | 5.61 x 10^{-4} | 0.68 | 0.50 | 70.72 |
| 2 | 5.61 x 10^{-4} | 0.68 | 0.50 | 72.53 |
| 3 | 1.24 x 10^{-3} | 0.22 | 0.55 | 66.72 |
| 4 | 2.76 x 10^{-3} | 0.38 | 0.69 | 81.96 |
| 5 | 2.42 x 10^{-4} | 0.44 | 0.84 | 50.52 |
| 6 | 2.44 x 10^{-3} | 0.80 | 0.51 | 90.09 |
| 7 | 9.74 x 10^{-5} | 0.43 | 0.77 | 31.13 |
| 8 | 3.79 x 10^{-3} | 0.26 | 0.89 | 81.54 |
| 9 | 1.25 x 10^{-5} | 0.35 | 0.84 | 30.94 |
| 10 | 2.35 x 10^{-5} | 0.82 | 0.54 | 36.20 |
| 11 | 7.64 x 10^{-3} | 0.24 | 0.93 | 83.26 |
| 12 | 5.61 x 10^{-4} | 0.68 | 0.50 | 72.99 |
| 13 | 1.24 x 10^{-3} | 0.22 | 0.55 | 68.57 |
| 14 | 2.76 x 10^{-3} | 0.38 | 0.69 | 81.25 |
| 15 | 5.61 x 10^{-4} | 0.68 | 0.50 | 70.35 |
The accuracies obtained on different values of the independent variables, i.e. Base learning rate, Gamma and Momentum are plotted in figures 3, 4 and 5 respectively. Finally, the 3D projection of the quad using colour as the fourth dimension (obtained accuracy), is plotted using the matplotlib library’s scatter plot function in figure 6.
Figure 3. Accuracy vs Base learning rate

Figure 4. Accuracy vs Gamma
3. Conclusion
The paper successfully implemented the Bayesian Optimization approach for hyper-parameter optimization of AlexNet DCNN model for Maize (corn) sub-dataset of the Plant Village dataset. Three hyper-parameters: Base learning rate, Gamma and Momentum were tuned using the Bayesian Optimization technique. As a result, the DCNN model achieved an accuracy of 96.05% at the base learning rate of 0.01, gamma of 0.90 and momentum of 0.99.
References

[1] R. Ramachandran, D. C. Rajeev, S. G. Krishnan, and P. Subathra, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[2] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, “Deep convolution neural network for image recognition,” Ecol. Inform., vol. 48, pp. 257–268, 2018, doi: 10.1016/j.ecoinf.2018.10.002.

[3] I. V Tetko, D. J. Livingstone, and A. I. Luik, “Neural Network Studies. 1. Comparison of Overfitting and Overtraining,” in Information Computation Science, 1995, pp. 826–833.

[4] R. S. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014, doi: 10.1016/0370-2693(93)90272-J.

[5] G. H. Ilya Sutskever, James Martens, George Dahl, “On the importance of initialization and momentum in deep learning,” in Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013, no. May, pp. 24–32, doi: 10.1017/CBO9781131642396.

[6] “Grid Search for model tuning - Towards Data Science.” https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e (accessed Jun. 07, 2020).

[7] J. Bergstra, J. B. Ca, and Y. B. Ca, “Random Search for Hyper-Parameter Optimization Yoshua Bengio,” 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[8] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, “Hyperparameter optimization for machine learning models based on Bayesian optimization,” J. Electron. Sci. Technol., vol. 17, no. 1, pp. 26–40, Mar. 2019, doi: 10.11989/JEST.1674-862X.80904120.

[9] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Commun. ACM, vol. 60, no. 6, p. 2017, 2012.

[10] “ImageNet.” http://www.image-net.org/ (accessed Jun. 06, 2020).

[11] “PyCharm: the Python IDE for Professional Developers by JetBrains.” https://www.jetbrains.com/pycharm/ (accessed Jun. 06, 2020).

[12] Y. Jia et al., “Caffe: Convolutional architecture for fast feature embedding,” in MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia, Nov. 2014, pp. 675–678, doi: 10.1145/2647868.2654889.

[13] D. P. Hughes, M. Salathé, and M. Salathe, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” CoRR, vol. abs/1511.0, 2015, doi: 10.1111/1755-0998.12237.

[14] E. Istook and T. Martinez, “Improved Backpropagation Learning in Neural Networks With,” Int. J. Neural Syst., vol. 12, no. 3, pp. 303–318, 2002, doi: 10.1142/S0129065702001114.

[15] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” J. Big Data, vol. 6, no. 1, pp. 1–48, Dec. 2019, doi: 10.1186/s40537-019-0197-0.