Robust Optimization of Convolution Natural Network

Chongjie Ye
The Chinese University of Hong Kong, Shenzhen, China

Abstract. Deep learning has played a very important role in computer vision. However, most of the methods used in computer vision highly rely on human to adjust the hyperparameter. That takes researchers lots of time, but the results sometime could not be most optimized. Besides, many architectures cannot perform robustly in training with noised data. This essay aims to solve the hyperparameter optimization problem by adapting the fruit fly optimization algorithm and suppose a high robust Convolution Natural Network including a Gaussian filter. Compared with methods such as FaceNe, InceptionV3 and Resnet5, GauCNN perform higher efficiency and accuracy with noise data.

Keywords: Convolutional Neural Networks, Fruit Fly Optimization, Robust, Noise.

1. Introduction
Deep learning is used for many applications like image classification, object detection, image segmentation, and etc [1-4]. In deep learning, there are two kinds of popular neural networks: Recurrent Neural Networks (RNN) and Convolution Neural Networks (CNN). They have advantages in specific area. For instance, CNN has great performance in image classification [5].

CNN is widely used in deep learning applications. As Pouyanfar (2018) concludes, there are three main advantages of using CNN: parameter sharing, sparse interactions, and equivalent representations. In other words, compared to traditional fully connected network, CNN has fewer parameters and faster to train. Despite a great performance, CNN has a disadvantage: its performance highly relies on the choice of hyperparameters, and for a deep CNN a large number of hyperparameters should be introduced [6]. Namely, to develop a good-performing network require human to select and modify the hyperparameter manually, and usually it takes them lots of time.

Beside, today's CNN network like FaceNet or Resnet has a almost perfect performance [4]. However, as for some dataset collected from real environment, sometimes there are some problems with the images like containing noisy data, mainly due to "malfunctions, unfortunate calibrations of measurement equipment or network problems during the transport of sensor information to a central measurement collection unit" [7]. There are four main kinds of noise in natural word: Gaussian Noise, Impulse Noise and Poisson Noise [8]. In fact, noise can significantly influence the performance of a machine learning model [7].

To solve above problems, this essay use Fruit Fly Optimization to improve the network's performance by simulating the fruit flies' searching food process to reduce the selecting hyperparameter time. Besides, this essay supposes GauCNN to improve the robustness of CNN by adding a Gaussian filter before the convolutional CNN network. The essay contains four-part: Introduction, Proposed Method, Experiment, and Conclusion.
2. Proposed GauCNN Method

Intuitively, adding noise makes the image dirty and profoundly influence the process of training and testing. This essay suggests a GauCNN architecture with FFO Module to reduce the effect of noise as much as possible.

2.1. Fruit Fly Optimization (FFO) Module

Fruit fly optimization (FFO) algorithm is a new method to find global optimization based on the performance of fruit fly foraging behavior [9]. It simulates fruit-fly-food-searching behavior by assuming a smell function. FFO algorithm has an advantage that it is easy to understand and implement [10].

The Fruit Fly Optimization module takes the following steps:

Set initial fruit fly location. The $X_{00}, X_{10}$ are the hyperparameter of CNN, and $Y_0$ is the test accuracy for image classification.

For a fly group of size $fly\_group\_size$, give random movement ($random\_value \times repeated\_times$) to every fruit fly: ($i$ represented which fly $X_0, X_1$ belongs to, $i \leq fly\_group\_size$), and repeated\_times is the times the same accuracies repeated, which is used to prevent the fly stuck at a constant axis. The move of one fly can be represented as:

$$X_{0i} = Activation(X_{0i-1} + random\_value \times repeated\_times) \quad (1)$$

$$X_{1i} = Activation(X_{1i-1} + random\_value \times repeated\_times) \quad (2)$$

Activation function here works to correct the $X_{0i}$ and $X_{1i}$ to make sure the value is valid (like preventing $X_{0i}, X_{1i}$ to become negative because CNN network parameter should usually be positive).

$$Y_i = Test(X_{0i}, X_{1i}) \quad (3)$$

$Test(\cdot)$ is to calculate the accuracy of test dataset after the model is trained with parameter $X_{0i}$ and $X_{1i}$.

We suppose the food location is at $Y = 1$. Therefore, the smell function will be:

$$Smell(Y_i) = \frac{1}{1-Y_i} \quad (4)$$

As $Y_i$ becomes larger, $Smell(Y_i)$ will be larger. Therefore, we can use $Smell(Y_i)$ to sift out the best fly.

Specifically, for each generation, we would find $[bestSmell, bestIndex] = max(Smell(Y_i))$ and set $X_{00} = X_{0bestIndex}, X_{10} = X_{1bestIndex}$. $X_{00}, X_{10}$ are the initial value of the next generation.

Repeat the 2nd and 3rd steps for total\_generations times. The optimized result of this module will be $X_{0bestIndex}, X_{1bestIndex}$ of the final generation.

2.2. GauCNN Architecture

GauCNN architecture is based on Gaussian Filter-CNN architecture, where the CNN layers are brought from part of VGG16 architecture. The proposed structure are shown as Fig.1.
Fig. 1 The architecture of the proposed GauCNN network

Gaussian filters have been widely used in image processing and computer vision [11]. Its primary function is to smooth noise and preserve edge details [12]. A study by [13] shows that Gaussian filters have a relatively stable and effective performance in denoising images with Gaussian noise, speckle noise, salt and pepper noise, and Poisson noise.

This module takes a tensor \( X_n \in R^{160*160*3} \) as input. It can be expressed with the followed formula:

\[
A_n = \text{Gau}(X_n), Y_n = \text{CNN}(A_n)
\] (5)

Where Gau indicates the Gaussian filter, and CNN indicates Convolutional Neural Networks. Besides, \( A_n \) is another tensor with 160 * 160 * 3 resolution while \( Y_n \) indicate the kind of object in the input image.

**Gaussian filter:** The Gaussian filter enables the network to denoise the input features, especially those with high levels of noise, i.e., Gaussian noise, Poisson noise, and salt and pepper noise. It is an M*M convolution kernel whose value follows two-D Gaussian distribution [12]. Specifically, the filter can be expressed as:

\[
G(x, y) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right)
\] (6)

Where \( x, y \) is the row and column number respectively, and \( x \in [1, M], y \in [1, M] \). \( \sigma \) is the variance of Gaussian filter, which determines how much noise to be smoothed out[12]. The too high variance will cause original image information distorted. Therefore, the essay set \( \sigma = 0.5 \) in this experiment. This convolution kernel is used to perform convolutional image processing on the input feature.
Convolutional Neural Networks: Inspired by [14], this essay adds a batch normalization layer after each convolutional layer to improve the network performance. The whole architecture is shown as Figure 1.

3. Experiment

In this section, we evaluate the Fruit Fly Optimization method on our GauCNN and compare GauCNN with several popular convolutional architecture such as FaceNet [1], InceptionV3 [2], and Resnet50 [3].

3.1. Parameter Settings

In this experiment, we set the filter number of all convolution layers as $X_{0i}$ and the length of the dense layer before the output layers as $X_{1i}$. Since our network is not a very deep convolution network, we set total_generations = 15, fly_group_size = 30, epoch = 1000, learning rate = 1, optimizer = Adadelta [15], and batch size=16. The $X_{00}, X_{10}$ in FFO are the filter number of the third and fourth convolutional layers in GauCNN architecture.

3.2. Dataset

This experiment uses a face recognition dataset collected from the Internet to train, which consist of 3K images of 4 celebrity (2K for training, 1K for testing). This dataset has been clearly labeled. Besides, 10% of the training data was used for validation.

3.3. Fruit Fly Optimization

Table.1 shows the increase in accuracy as generation increases. Y starts from 0.855, with default X0(32) and X1(64). The accuracy experiences a rapid increase in the first two generations with 5.8%, while it stuck at 0.913 for the next 7 generations. The accuracy increases as X0 and X1 increase. In the 12th generation, the accuracy increases by 0.5% as X0 and X1 drop to 65 and 140, respectively. Finally, the accuracy peaks at 0.940 in the 13th generation.

| Generation | X0  | X1  | Y    |
|------------|-----|-----|------|
| 0          | 32  | 64  | 0.855|
| 1          | 23  | 66  | 0.900|
| 2          | 41  | 55  | 0.913|
| 3          | 41  | 55  | 0.913|
| 4          | 30  | 51  | 0.918|
| 5          | 30  | 51  | 0.918|
| 6          | 30  | 51  | 0.918|
| 7          | 30  | 51  | 0.918|
| 8          | 30  | 51  | 0.918|
| 9          | 30  | 51  | 0.918|
| 10         | 174 | 303 | 0.928|
| 11         | 174 | 303 | 0.928|
| 12         | 65  | 140 | 0.933|
| 13         | 92  | 168 | 0.940|
| 14         | 92  | 168 | 0.940|
| 15         | 92  | 168 | 0.940|

This experiment shows that fly optimization has a great impact to model’s accuracy and can be used as a useful tool to optimize the model’s architecture.
3.4. GauCNN Optimization

In this part, different models be trained with data added Poisson noise, Impulse noise, and Gaussian noise, respectively. Fig. 2–4 show a comparison between the accuracy of each model as sigma or percentage rise and GauCNN perform robustly compared with other models.

![Fig. 2 Accuracy with Poisson-noise train set and original test set](image1)

As for Poisson-noise data (Fig.2), all model's accuracy decreases gradually as the noise sigma increase. InceptionV3, FaceNet and GauCNN perform high accuracy when sigma is less than 0.15, while Resnet50's accuracy is low throughout the process. However, InceptionV3's accuracy drops significantly as noise sigma is larger than 0.15. Although GauCNN and FaceNet's accuracy decrease as noise sigma increase, GauCNN stills perform a relatively high accuracy at sigma 0.5 while FaceNet's accuracy drops below 0.4 at the same sigma.

![Fig. 3 Accuracy with Impulse-noise trainset and original testset](image2)

Fig. 3 illustrates a graduate and stable decreases in all methods as the Impulse noise sigma increases. Compared to the accuracy of Poisson-noise training set, the gap between InceptionV3 and FaceNet decrease, while GauCNN still performs a much higher accuracy than the other three methods.
In Fig. 4, GauCNN shows a similar performance to Facenet for Gaussian-noise training set, though GauCNN performs a higher accuracy as Gaussian noise’s sigma is larger than 0.4. Beside high robust, GauCNN also performs high efficiency. GPU hour is evaluated for an NVIDIA V100 (Fig. 6).

Table 2. Time cost of object classification

| Training Time/per 1000 epochs | GPU hour |
|------------------------------|----------|
| GauCNN                       | 0.2      |
| FaceNet                      | 0.28     |
| Resnet50                     | 0.35     |
| InceptionV3                  | 0.26     |

4. Conclusion

In this paper, a deep convolution neural network with FFO optimization module is proposed for robust optimization. Unlike most past architecture requiring human to adapt the hyperparameter to get a higher model performance. Our architecture is able to adjust to the training set automatically by using the FFO module. Also, in the case that training data which has high noise signal, our architecture could also handle it and get high performance in the testing.

References

[1] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” 2015 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, pp. 815–823, Jun. 2015, doi: 10/gdcfkq.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” ArXiv151200567 Cs, Dec. 2015, Accessed: Apr. 19, 2020. [Online]. Available: http://arxiv.org/abs/1512.00567.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” ArXiv151203385 Cs, Dec. 2015, Accessed: Apr. 19, 2020. [Online]. Available: http://arxiv.org/abs/1512.03385.

[4] S. Pouyanfar et al., “A Survey on Deep Learning: Algorithms, Techniques, and Applications.” Association for Computing Machinery, Sep. 18, 2018, Accessed: Apr. 19, 2020. [Online]. Available: https://doi.org/10.1145/3234150.

[5] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “CNN-RNN: A Unified Framework
for Multi-Label Image Classification,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2285–2294, Accessed: Apr. 19, 2020. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Wang_CNN-RNN_A_Unified.CVPR_2016_paper.html.

[6] Z. Li, W. Yang, S. Peng, and F. Liu, “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects,” ArXiv200402806 Cs Eess, Apr. 2020, Accessed: Apr. 19, 2020. [Online]. Available: http://arxiv.org/abs/2004.02806.

[7] E. Kalapanidas, N. Avouris, M. Craciun, and D. Neagu, “Machine Learning algorithms: a study on noise sensitivity,” p. 10.

[8] A. K. Boyat and B. K. Joshi, “A Review Paper: Noise Models in Digital Image Processing,” ArXiv E-Prints, vol. 1505, p. arXiv:1505.03489, May 2015.

[9] “A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example: Knowledge-Based Systems: Vol 26, No null.” https://dl.acm.org/doi/10.1016/j.knosys.2011.07.001 (accessed May 27, 2020).

[10] W. Sheng and Y. Bao, “Fruit fly optimization algorithm based fractional order fuzzy-PID controller for electronic throttle,” Nonlinear Dyn., vol. 1–2, no. 73, pp. 611–619, 2013, doi: 10.1007/s11071-013-0814-y.

[11] G. Deng and L. W. Cahill, “An adaptive Gaussian filter for noise reduction and edge detection,” in 1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference, Oct. 1993, pp. 1615–1619 vol.3, doi: 10.1109/nssmic.1993.373563.

[12] M. Mafi, H. Martin, M. Cabrerizo, J. Andrian, A. Barreto, and M. Adjouadi, “A comprehensive survey on impulse and Gaussian denoising filters for digital images,” Signal Process., vol. 157, pp. 236–260, Apr. 2019, doi: 10.1016/j.sigpro.2018.12.006.

[13] M. Gupta, H. Taneja, and L. Chand, “Performance Enhancement and Analysis of Filters in Ultrasound Image Denoising,” Procedia Comput. Sci., vol. 132, pp. 643–652, Jan. 2018, doi: 10/ggqm98.

[14] M. Simon, E. Rodner, and J. Denzler, “ImageNet pre-trained models with batch normalization,” ArXiv161201452 Cs, Dec. 2016, Accessed: Apr. 02, 2020. [Online]. Available: http://arxiv.org/abs/1612.01452.

[15] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” ArXiv12125701 Cs, Dec. 2012, Accessed: Apr. 19, 2020. [Online]. Available: http://arxiv.org/abs/1212.5701.