Is Deep Learning Always an Elixir?

Jingsheng Deng

Research School of Computer Science, Australian National University, Canberra Australia
u6847863@anu.edu.au

Abstract. The neural network has been proven to be a powerful tool for complex machine learning tasks in many areas and directions. However, the high cost of computation is always feeble for deep neural network because of their lots of parameters and complicated structures. Therefore, how to improve its efficiency and reduce computational complexity are urgent tasks by now. In this paper, we apply a technique named Cascade Neural Network to build an efficient and powerful model. We modified the training procedure and used different optimizers, and the results show that this technique can indeed boost a neural network's ability while keeping fewer parameters. We also use CNN as the baseline for the task. Our task is to judge whether the anger of a human face is genuine or not, using a range of features or the original data of the observers' pupil. The classification hyperplane is so complex that a simple two-layer or three-layer network can hardly handle it and give a satisfying result. However, since the number of parameters increases dramatically when the number of layers increases, it may lead to a large cost improvement on computation. After using the Casper Neural Network technique, we balanced the performance and computation cost in the end.

Keywords: Neural Network · Convolutional Neural Network · Cascade Structure · Emotion Veracity Detection

1 Introduction

For some current machine learning tasks, traditional statistic methods are not powerful enough, and only neural network [7] can handle such involuted problems. A neural network's ability grows with the increases of the neural network's depth and the number of neurons of each layer. But a neural network's parameters increase exponentially with its depth, which probably leads to a computational explosion. Therefore, how to improve a neural network's efficiency is worth to be discussed. In this paper, we used an anger veracity detection task with dataset [1] to study deeper in this problem. In this dataset, each sample is a series of pupil sizes of participants watching videos of human faces. The pupil size is captured for each frame of the video and the length of videos and the number of pupil sizes are different. As the decision hyperplane is high dimensional, a two-layer or a three-layer neural network's performance is poor. But if we use a truly deep neural network, it may contain a lot of parameters, which lead to much slower training and evaluation. Thus, we introduced a technique named Cascade Neural Network [6, 2]. We modified the original training procedure in [6, 2], and experimented with different optimizers. We try to use a Convolutional Neural Network (CNN) [4] versus constructing features and using a normal neural network. The results show

that by constructing features and using a normal neural network, this technique can indeed provide a powerful model with more non-linearity and few parameters, which means an efficient model, but when using a CNN model [4], we can hardly get a good result. Also, the results show that the RMSprop optimizer works best for our model in this task.

In general, we modified the training procedure of the Cascade Neural Network and experimented with different optimizers on traditional neural network and CNN [4]. On the anger veracity detection task, using this technique with a modified training procedure gives better results, and the RMSprop optimizer works best.

2 Method

Our network structure looks like [6, 2]. When we experiment with constructed features and a normal neural network, the network start with two neurons, one is the output neuron (as the task is a 2-class classification problem), one takes the 6 features as input whose output together with the 6 features, are fed into the output neuron. After training some epochs, a new neuron is added to the network and works as the output neuron, it takes all 2 outputs of the 2 neurons we described before and the 6 features. Then each time a new neuron is added into our network, it works as the output neuron and takes all the outputs of other neurons as well as the 6 features. Thus, we could notice that the previous output neuron is now connected to the new neuron added into the network and act as an input. When experimenting with CNN [4], what we do is the same work, but the neurons are replaced with convolutional layers, and every time a new convolutional layer is added, it has more input channels which like the newly added neurons when using a deep neural network. When using CNN [4], an important difference is that we choose to use 2 independent networks, and the final output is the concatenation of them. One is used to detect if any real anger faces exist, and the other is to detect if a pretended anger exists. Adding convolutional layers is applied to both. We choose to use 2 independent networks because we believe the patterns of pupil size change are different when a participant is watching a video of real and pretended anger, and when we complement the cascade network technique [6, 2], the output of each convolutional layer has only one channel, the network may be confused if we ask it to detect 2 different things and describe them differently in 1 channel. Although we can use convolutional layers with 2 output channels, the parameters may increase exponentially, and that is also the reason for choosing to use 2 independent networks.

Our training procedure is different from the model described in [6], as a new neuron in that paper is first trained to learn the residual loss of the model, and then it is added into the model, but in our method, it is directly added into the model and trained in the usual way to minimize the cross-entropy loss. We do this modification because we believe that our training procedure is more straightforward, and during the training process, the new neuron can automatically learn the residual loss. Inspired by [6, 2], we also choose different learning rates for the newly added neuron and other neurons, and the learning rate for the newly added neuron is much larger than these for the other neurons since the newly added neuron has not been trained before and need larger speed to reach convergence.

3 Experiment

3.1 Dataset

The dataset used in this paper is an anger veracity detection dataset proposed by [1]. The task is to detect whether a face in a sample is expressing real anger or not using pupil size data. The samples are a series of pupil sizes of participants watching different videos measured on them. One participant watching one video forms one sample. For each sample, the number of pupil sizes depends on how many frames this video has. This dataset contains 390 samples. We construct 6 features (mean, std, diff1, diff2, PCAd1, PCAd2) to train the neural network. This dataset is not split into the training set and testing set by the authors, so we split them randomly into 80% for training and 20% for testing. When using CNN [4], all the pupil sizes are normalized together to have a mean of 0 and a standard deviation of 1 (standard normalization). While using the normal neural network, the 6 features are also standardly normalized.

3.2 Implementation

Our model starts with 2 neurons as described in the last part. And after all training processes, it will contain 11 neurons totally, and one of them is the final output neuron. The model is trained for 100,000 epochs, and after every 10,000 epochs, a neuron is added into our model except the last 10,000 epochs. For CNN [4], the neurons are replaced by convolutional layers for the two network we mentioned. It is trained for 10,000 epochs and a convolutional layer is added every 1000 epochs. For CNN [4] model, since the input is a 1-dimension series, the convolutional layer is also 1-dimension. The kernel size is set to 5, with the stride of 1 and zero paddings of size 2 to keep the output of the same size as the input. We experimented with 3 different optimizers, SGD, RMSprop and Adam [3]. Because of their properties, the learning rates, momentums for them are different. Please check the detailed settings in table 1. The weight decay is 0.0001 for all the 3 optimizers. For comparison, we also trained a neural network with one hidden layer of 10 neurons using SGD as a baseline, all the hyper-parameters for this model are the same as our model trained with SGD optimizer except the weight decay is set to 0 because the model is too simple, it can hardly overfit the dataset. All the hyper-parameters are selected empirically.

Table 1. Learning Rate and Weight Decay for Different Optimizers

Optimizer	Learning Rate (Old and New Neurons)	Momentum
SGD	0.03, 0.1	0.9
RMSprop	0.003, 0.01	0.9
Adam [3]	0.0003, 0.001	Not Applicable

4 Results

4.1 Normal Neural Network

Figure 1 shows the accuracy performance comparison between the baseline and our proposed model. We can conclude that firstly, our model's accuracy is lower than our baseline because at that time our model has too few neurons. And after 50,000 epochs, our model gains its 5th neuron, and its performance increased a lot and surpassed our baseline. We can also find that our model's accuracy is not stable. The reason is that untrained new neurons are added into it every 10,000 epochs, and they need to learn some periods to have a decent performance. Our baseline's performance stops increasing after a few epochs, which means it meets its limit very soon. Our baseline model has 81 parameters, and our proposed model has 132 parameters, we can see that the number of parameters does not increase a lot, but its performance increased more than expected.

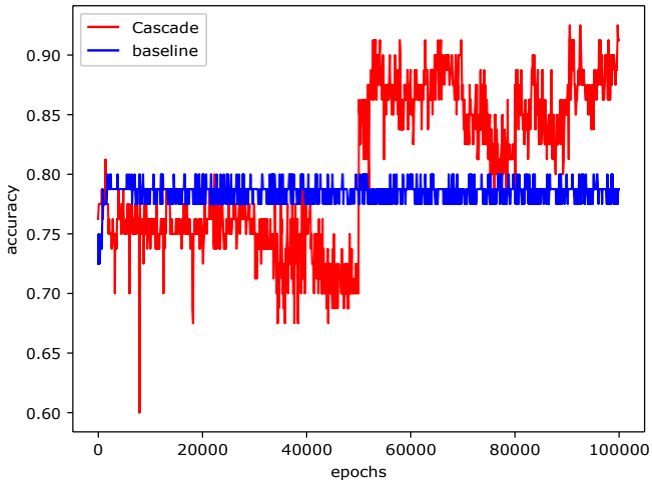

Fig. 1. Accuracy Performance Comparison between Baseline Model and Our Proposed Model

Table 2 shows the best accuracy gained using different optimizers, we can find out that RMSprop works best for this model and this task. We also compare our results with results reported in [1], and it shows that our model's performance is the same as the method proposed in [1].

Table 2. Accuracy Performance of Different Optimizers

Optimizer	Highest Accuracy
SGD	0.8625
RMSprop	0.95
Adam [3]	0.9125
Chen et al. [1]	0.95

4.2 CNN [4] Model

Figure 2 shows the accuracy performance of our CNN [4] model. We can see that our CNN [4] model's accuracy is much lower than chosen neural networks. The highest accuracy we obtain using CNN [4] model with different optimizers and hyper-parameters is 69.23%. During training, we observe that the loss on the training set keeps decreasing, but figure 2 shows that the accuracy is oscillating but not increasing, which we know means that the model has overfitted.

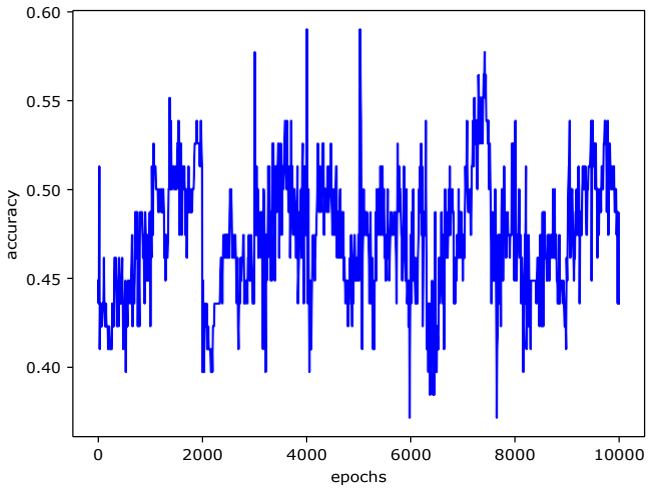

Fig. 2. Accuracy Performance of our CNN [4] model (with RMSprop optimizer)

CNN [4] model has shown great ability on many figure classifications tasks. However, the low performance of our CNN [4] model indicated that this structure might not fit the task. The features we constructed for our normal neural network is intuitive to obtain and enables our normal neural network model to achieve highlighted performance, but when these features are not provided, we can see that even a clever model like CNN [4] can hardly learn these features by itself, which proves that deep learning is not so powerful as we think, many simple non-linear operators are so hard for a deep learning

model to handle with, and constructing features is still a significant step in deep learning areas.

5 Conclusion and Future Work

We modified the Casper [6] method and proved our modified version's effectiveness. We compared different optimizers for this task and our model, and the experiment results show that RMSprop is the best optimizer for this task and our model. We displayed that CNN [4] structure is not always powerful and constructing features [5] is still important for deep learning. In the future, we would like to further improve our model's efficiency or reduce the computation cost during training, such as freeze the neurons added at an early stage so that gradients for them are no longer needed. Also, how to enable neural networks to learn some non-linear functions is worth researching more.

References

1. Chen, L., Gedeon, T., Hossain, M.Z., Caldwell, S.: Are you really angry? detecting emotion veracity as a proposed tool for interaction. In: Proceedings of the 29th Australian Conference on Computer-Human Interaction. pp. 412–416 (2017)
2. Khoo, S., Gedeon, T.: Generalisation performance vs. architecture variations in constructive cascade networks. In: International Conference on Neural Information Processing. pp. 236–243. Springer (2008)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
4. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4), 541–551 (1989)
5. Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E.B., Turaga, D.S.: Learning feature engineering for classification. In: Ijcai. pp. 2529–2535 (2017)
6. Treadgold, N.K., Gedeon, T.D.: A cascade network algorithm employing progressive prop. In: International Work-Conference on Artificial Neural Networks. pp. 733–742. Springer (1997)
7. Wang, S.C.: Artificial neural network. In: Interdisciplinary computing in java programming, pp. 81–100. Springer (2003)