Using DNNs to Train Blade Design Dataset

Enyi Jiang, Volodynyr Kindratenko

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

Abstract

In this report, I will explore the performances of training several deep neural network structures using blade design dataset. DNNs could help us extract the important features from the original dataset which are necessary to output the correct targets we want (the pressure loss coefficient and deviation angle). Considering the size of the dataset, I have tried to find the feasible layer configuration and data preprocessing methods to train the data into convergence. In the process of experiments, I have gained a deeper understanding upon different factors in the deep nets which may contribute to various results and deviational output results on the targets.

1. Introduction

Researchers are interested in the nonlinear relationship between blade geometry parameters especially 1) the pressure loss coefficient and 2) deviation angle and quantities of interest (such as surface pressure distribution, drag/lift ratio) [9]. One of the most useful ways to extract these relationships is using deep learning. CNN is a very popular model which is usually used in image feature detection and extraction. In our situation, our inputs are the simulation results of some physical quantities, which could be treated as different channels of an image since we can know the value in every single position (pixel) in the image. We have used several responses to estimate the target parameters we want using several recent famous DNNs, like ResNet50, VGG-16, and AlexNet. It is essential to use multiple responses since the single response may not contain sufficient information to output the blade geometry parameters. To uncover the hidden relationship among parameters, DNN modeling is rather worth exploring and experimenting on this blade dataset.

2. Methods

2.1 Data Preprocessing

Data Preprocessing is a rather vital step for training a dataset to convergence. The quality of data and the useful information that can be derived from it directly affects the ability of our model to learn. Many deep learning methods are sensitive to the range and distribution of attribute values in the input data. [4] The main goal for this part is to make both the input and targeted data distribute more normally and evenly so as to become easier to train. I will introduce the approaches which I have employed as follows.

Standardization: This method basically manages to transform the input data into the values with zero mean and one standard deviation. This step is important since if we have several inputs in our

DNNs and some of them have bigger values and others are close to zero, then our neural networks would tend to attach more importance to these inputs with a larger value.

Delete Outliers: Outliers are some of the data points in the input dataset which have very extreme or incompatible values with the rest data distributions. Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results [2]. In our blade design dataset, some of the points are not physics which have pressure loss coefficient bigger than 1 or smaller than 0, so are needed to remove them manually. Usually, we could detect our outliers by using Boxplot, scatterplot or Z-score approaches [5].

Power Transform: When we do the data preprocessing, we need to look into the distribution of our targets. If they are mostly condensed in a small range, then it would become much harder for our DNNs to be trained to these mostly identical values. Thus, to change our targets into a better distribution, we could power transform the targets, resulting in a more evenly distributed target values, which could improve our training process with a better performance.

2.2 ResNet50

ResNet50 is a famous deep neural network which has a highly complex and unique structure with great performance on image classification. Usually having deeper layers in a DNN could lead a more accurate result of training. However, this kind of rewarding diminishes and even degrades as we continually increase the number of layers. Thus, what ResNet has done is defining a residual function using H(x) = F(x)+x, where F(x) and x represents the stacked non-linear layers and the shortcuts (input=output) respectively [3]. Those shortcuts act like highways and the gradients can easily flow back, resulting in faster training and much more layers [8]. The structure of ResNet could be separated into two kinds: identical block and convolutional block, consisting of several rounds of convolution, batch norm, and ReLU layers with one shortcut in between. The overall structure of ResNet is generally adding these blocks together.

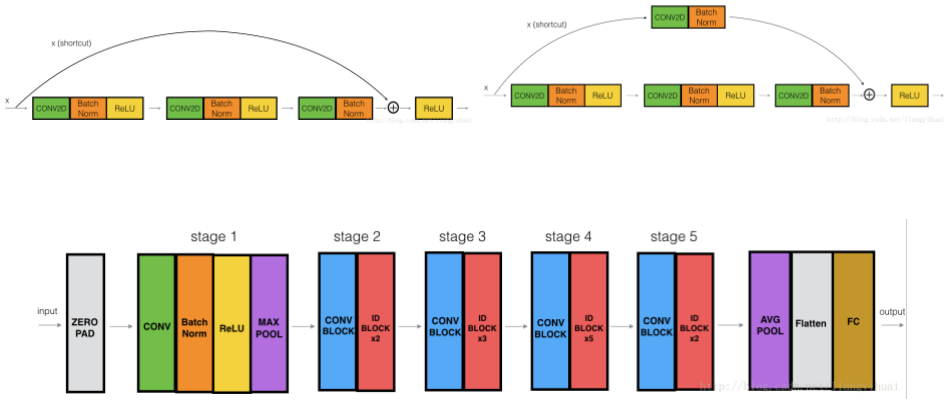

Figure 1 Network structure of ResNet50 [5]

VGGNet consists of 16 three by three convolutional layers with ReLU and max-pooling layers inside. It has a simpler architecture compared with ResNet but with many filters, leading to a depth of 4096 output in the end. It can fully extract the features from the picture to do the classification. Considering our situation of predicting parameters, I have changed the softmax layer into several fully connected layers to output the regression values.

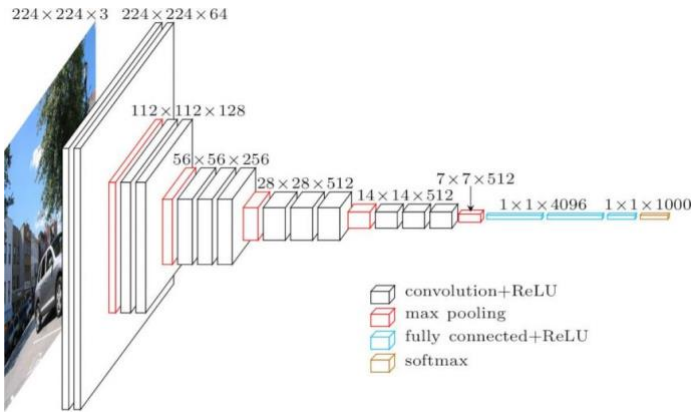

Figure 2 Network structure of VGG-16 [6]

AlexNet consists of 11x11, 5x5,3x3, convolutions, max pooling, dropout, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer [1]. Compared with VGG-16 Net, it has fewer filters with different sizes from big to small. Besides, it includes more drop out and normalization layers after each stage.

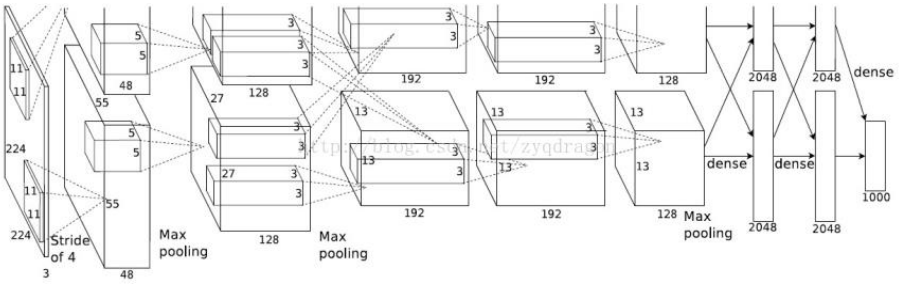

Figure 3 Network Structure of AlexNet[1]

3. Experiments and Results

3.1 ResNet (loss at 300 steps):

It seems that the network fails to converge. However, with some data preprocessing, I found that the testing loss has become smaller than before.

Table 1. Training and Testing Loss for ResNet after 300 Steps

	training loss	testing loss
without training data standardization	19.8341	2.99279e+25
with training data standardization	21.9114	381.66

3.2 VGG-16 (loss at 300 steps):

Generally speaking, it seems that VGG-16 performs better on testing dataset compared with Resnet50. However, both methods seem not to converge very well. After applying the image preprocessing method, the results seem to not vary too much.

Table 2. Training and Testing Loss for VGG-16 after 300 Steps

	training loss	testing loss
without training data standardization	22.112	381.66
with training data standardization	19.0888	381.66

3.3 Improved VGG-16:

I have modified the structure of VGG-16 with less out channels, simpler structure and doing more data preprocessing. I will illustrate my ideas for changing below:

i. Standardize targets and delete input images outliers. This could produce a more feasible training dataset which is easier for the DNN to train.
ii. Using Gradient descent optimization method to slowly get to the optimal loss value. Using other optimizers seemed to lead to vanishing results (all zeros).
iii. Using less convolutional layers to prevent from overfitting.

Result Visualization: The training result of the target 0 data failed (all zeros output), but target 1 is performing better (nearly converges). Compared with the training data, testing data behave worse. This neural network still exists some problems so it failed to converge completely.

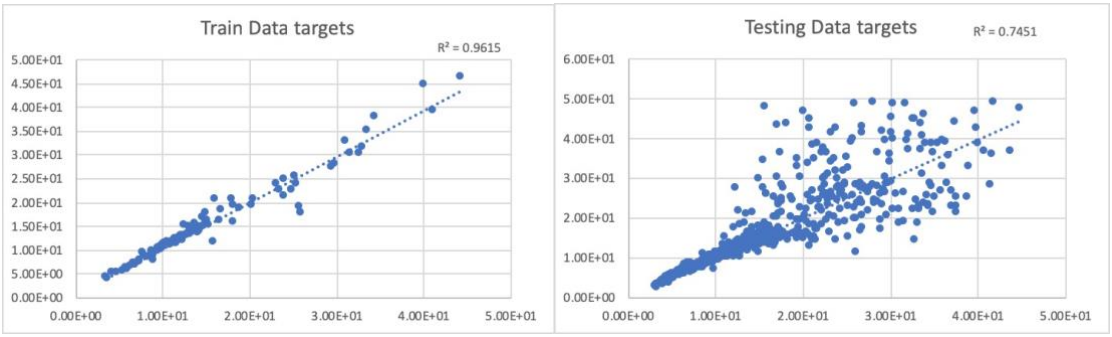

Figure 4 Result Visualization for Target 1 on Training and Testing Dataset

3.4 Improved VGG-16 Method 2:

When I took a look into the target values, I found that they are generally rather close with respect to each other. So I have adapted the power transformation method to the targets. The desired data distribution would be skewness to be close to 0 and kurtosis to be close to 3. Meanwhile, I have added more drop-out & batch-norm layers to better prevent overfitting. However, these transformations seem to help solve the all-zeros problem for target 0 while both targets failed to have good behavior on their predictions. Result Visualization:

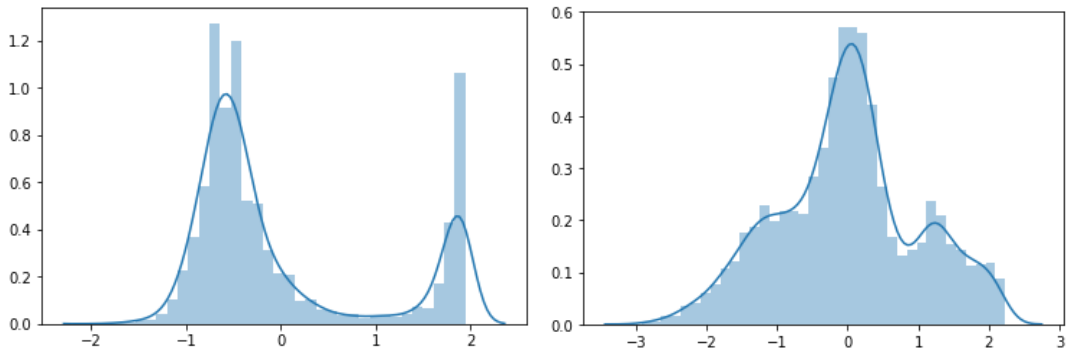

Figure 5 Result Visualization of Target 0 and 1 distributions after transformation. (a) pressure loss coefficient (b) deviation angle

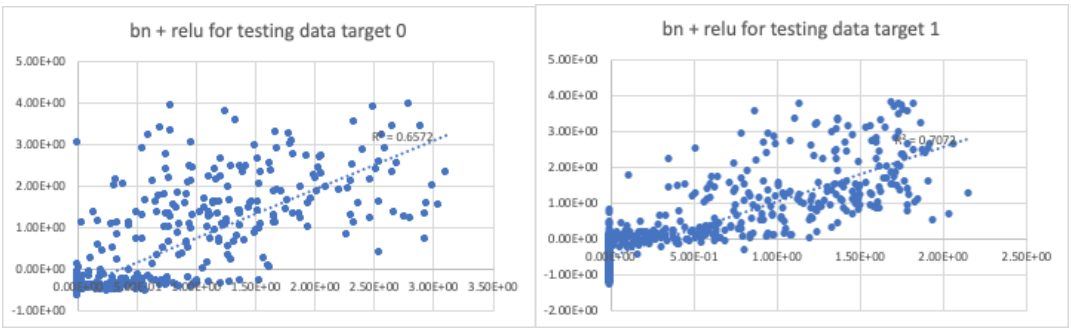

Figure 6. Result Visualization of the target testing results. (a) pressure loss coefficient (b) deviation angle

3.5 AlexNet:

Using this DNN resulted in the best performance up to now. Both targets were much closer to convergence after employing all the data preprocessing methods. Result Visualization:

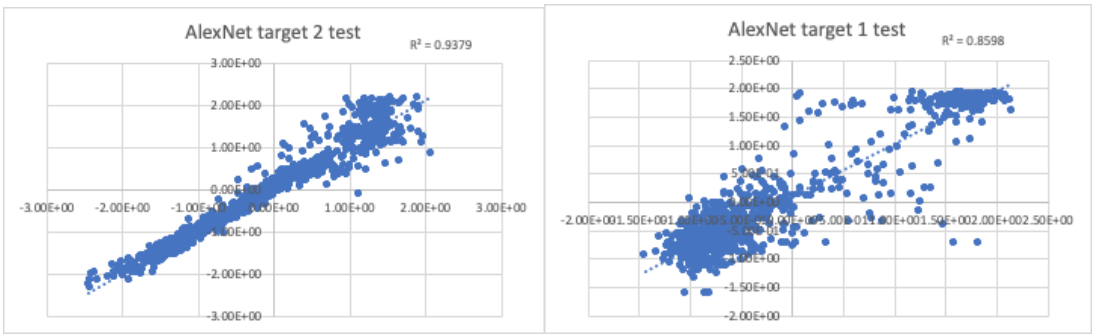

Figure 7 Result Visualization of the target testing results. (a) pressure loss coefficient (b) deviation angle

4. Discussion and Conclusion

In this project, I have realized the importance of data preprocessing and tried to find out how to choose a suitable DNN for training a relatively small dataset. For input data and target data, we need to use different methods. We need to make the weights between inputs to be more equivalent by using standardization and deleting some outliers to prevent overfitting. While our targets should have a good distribution with good skewness and kurtosis values. Apart from that, knowing more about the inner

characteristics of our dataset is rather important. For our blade design data, its number of samples is not so big. However, both ResNet and VGG-16 Nets have rather complicated structures with the overwhelming power to extract features from the inputs, which could easily lead to overfitting problems. They are usually used for training a large scale of data, therefore they may not be the best choice under our circumstances. I need to simplify their structures in order to fit into our situations. Especially for VGG-16 Net, when I included a smaller number of convolutional layers, it performed better than the original one, which confirmed this highly-likely overfitting problem. Also, I have found that VGG-16 uses 3x3 filters which tries to extract all the features from every small region of pixels, which varies from our needs, for our output parameters may just relate to part of the region in an image. Consequently, it is likely to encounter overfitting problems when training VGG-16.

In contrast, AlexNet has used a bigger filter at first, then use a smaller filter afterward. This kind of operation helps to condense the features and locate the region of interests in a more effective way. Besides, AlexNet has included more layers such as ReLU, drop out and normalization to prevent overfitting. Maybe this is the reason that it could lead to a good result than other nets.

5. Reference

[1] Das, Siddharth, and Siddharth Das. "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and More ...." *Medium*, Medium, 16 Nov. 2017, medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.

[2] "How to Identify Outliers in Your Data." *Machine Learning Mastery*, 7 June 2016, machinelearningmastery.com/how-to-identify-outliers-in-your-data/.

[3] Jay, Prakash, and Prakash Jay. "Understanding and Implementing Architectures of ResNet and ResNeXt for State-of-the-Art Image..." *Medium*, Medium, 7 Feb. 2018, medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624.

[4] Kumar, Dhairya, and Dhairya Kumar. "Introduction to Data Preprocessing in Machine Learning." *Towards Data Science*, Towards Data Science, 25 Dec. 2018, towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d.

[5] Sharma, Natasha, and Natasha Sharma. "Ways to Detect and Remove the Outliers." *Towards Data Science*, Towards Data Science, 22 May 2018, towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba.

[6] "VGG16 - Convolutional Network for Classification and Detection." VGG16 - Convolutional Network for Classification and Detection, 21 Nov. 2018, neurohive.io/en/popular-networks/vgg16/.

[7] "使用 Tensorflow 实现残差网络 ResNet-50." *使用 Tensorflow 实现残差网络 ResNet-50 - 蜗牛爱上星星 - CSDN 博客*, blog.csdn.net/liangyihuai/article/details/79140481.

[8] Lei Sun. "ResNet on Tiny ImageNet." http://cs231n.stanford.edu/reports/2017/pdfs/12.pdf

[9] Shirui Luo. "Turbomachinery Blade Design Using Deep Learning."