Analysis of machine learning algorithms for small vision data

Junyeong Park
Department of Computer Science
Hanyang University
Seoul, South Korea
jyp10987@gmail.com

Abstract

The most recent machine learning trend is to build a large model based on a vast amount of data. However, sometimes there may not be much data available. In this situation, using a heavy model poses a risk of overfitting. This paper presents which algorithms are most effective to use, especially in the field of computer vision, where there is little data. We trained algorithms that are not burdensome to use with only a single CPU without special computing devices such as GPUs or TPUs. And then we compared each model based on the time it took to be trained, the inference speed of the model, and translation invariance. The experimental results demonstrate the convolutional neural network is the best model in general computer vision with a small dataset in terms of accuracy and robustness.

1 Introduction

Since Rosenblatt proposed the term perceptron in 1957, many machine learning algorithms have been developed, such as support vector machine, and deep learning. Today, the most commonly used method is deep neural networks that have a lots of parameters to tune, for example, ResNet and Transformer based on a powerful compute equipment and a large dataset. However, although these methods are very powerful, the following problems exist.

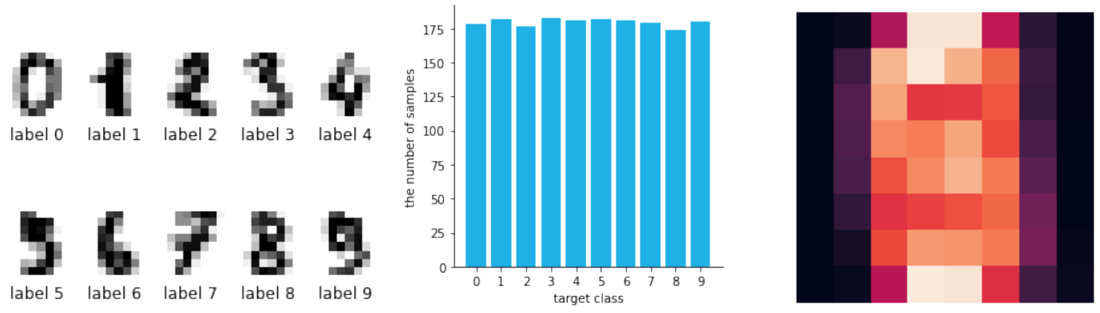
- Consumption of enormous computational resources
- Requires a large amount of training data
- Take a long time to train
- Easily overfitted

Hence, it is difficult to use a huge model in situations where computational resources are insufficient and data are not enough, and instead, a small model may be a better choice. Small models are generally faster than large models and are learned well enough with even small dataset.

In this paper, we compared the accuracy and speed of some algorithms, and investigated suitable model for use in computer vision tasks.

Hand-written digit dataset (MNIST) includes 1,797 grayscale images whose width and height are 8 respectively. Each image contains one of the numbers from 0 to 9. All input data is expressed as a matrix composed of values between 0 and 16. To be more specific, in an image, pixels without digits are represented by numbers close to zero, and pixels with digits are represented by numbers close to 16. Our goal is to use this dataset to build a model that predicts which of the numbers an image is. Fig. 1–(b) shows that the number of samples in each class is nearly equal, so that we do not need to care about the processing of the imbalanced dataset. And in Fig. 1–(c), numbers are mostly located in the middle of the image.

The remainder of the paper is organized as follows. Section II introduces how we set up and trained models. In Section III, each model is compared based on the results of the experiment. Finally, we conclude the paper in Section IV.



(a) Sample Data (b) Data Distribution (c) Mean Image

Figure 1: (a) Sample images for each class. (b) The number of samples in each class. (c) Image where (i, j) pixel is an average of (i, j) pixels from all images.

2 Experiments

2.1 Training

We trained classifiers to solve the problem with machine learning algorithms: Logistic Regression, Support Vector Machine, Fisher Discriminant Analysis, and Neural Networks. For each algorithm, we check the speed of training, inference speed, and translation invariance under the same condition. In the following sub-sections, we will describe how we used each algorithm.

2.1.1 Logistic Regression

Logistic Regression is designed for binary classification tasks. Given an input vector, it outputs yes or no. Therefore, we need to adopt a heuristic, one-vs-rest, because our task is multi-class classification. It divides multi-class dataset into several binary classification problems. A logistic regression model is then optimized on each binary classification problem. Finally, we use the Limited Memory BFGS (LBFGS) algorithm to optimize the model by minimizing the objective function following.

$$L(\mathbf{w}, b) = \sum_{i=1}^{N} \log\left(\exp\left(-y_i\left(\mathbf{x}_i^\top \mathbf{w} + b\right)\right) + 1\right) + \frac{1}{2}\mathbf{w}^\top \mathbf{w} \tag{1}$$

2.1.2 Support Vector Machine

Support Vector Machine is also a binary classifier. Hence, we used one-vs-rest heuristic for the same reason as in logistic regression. And we experimented with a few different kernel functions: Linear kernel, Gaussian kernel, Sigmoid kernel, and Polynomial kernel with degree 2 and 3.

2.1.3 Fisher Discriminant Analysis

Fisher Discriminant Analysis, unlike the previous two algorithms, allows for multi-class classification. And it has closed-form solution rather than using iterative method. We used least square solver to fit the model.

2.1.4 Multi Layer Perceptron

We design a network with the architecture following:

(1) A fully connected layer to a hidden layer of size 64

(2) A rectifier non-linearity

(3) A fully connected layer to a hidden layer of size 32

(4) A rectifier non-linearity

(5) A fully connected layer that outputs a vector of size 10, corresponding to logit probabilities for each class

The model parameters θ are updated by gradient descent using Adam optimizer on the loss function following:

$$L(\theta) = -\mathbf{t}^\top \log \mathbf{p} + 10^{-4} \times \|\theta\|^2 \quad (2)$$

2.1.5 Convolutional Neural Network

Unlike other algorithms above, convolutional neural network gets the grayscale of $8 \times 8 \times 1$ itself as input without flattening. The input data are processed by the feature extractor.

The feature extractor applies the following modules:

(1) A convolutional of 16 filters of kernel size 3×3 with stride 1 and zero-padding.
(2) A rectifier non-linearity
(3) A max-pooling of kernel size 2×2 with stride 2
(4) A convolutional of 32 filters of kernel size 3×3 with stride 1 and zero-padding.
(5) A rectifier non-linearity
(6) A max-pooling of kernel size 2×2 with stride 2

The output of the feature extractor is passed into a fully connected layer that outputs a vector of size 10, corresponding to logit probabilities for each class. We optimized the model in the same way as in the case of the multi layer perceptron.

2.2 Validation

To measure the accuracy of each model and check whether the model is in overfitting or not, we first split the MNIST dataset into two subsets. One is the training set. It is used to update the parameters of the models. The other is the validation set, which consists of 200 data pairs (20 samples per class, about 11% of the total data). It is not used to train the models directly. Two datasets do not have any overlapped samples.

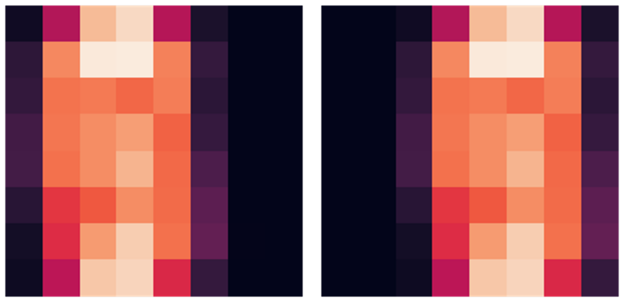


Figure 2: Mean image of images in translated validation set. Compared to Fig. 1-(c), they are skewed to the left or right.

To test translation robustness, we also created a new validation set, translated validation dataset. It consists of images of the validation dataset translated left or right. It consists of a total of 400 image sets, with 200 left-biased and 200 right-biased. However, we do not include this kind of data in the training set. In other words, the models experience only center-centered images in the training phase.

3 Results

In following sub-sections, we will show the comparison between algorithms in two perspective: efficiency, and translation invariance.

3.1 Efficiency

Efficiency does not simply mean speed. Basically, it can be said that a model with a high accuracy guaranteed and an acceptable speed is efficient. From this point, the most efficient models are the Support Vector Machine with Polynomial kernel of degree 3 and the Convolutional Neural Network. In Table 1, Support Vector Machine with Polynomial

Model name	Top-1 acc.	Training time (sec)	Inference speed (img/sec)
Logistic Regression†	0.92	0.28	257231
SVM (Linear)	0.93	0.02	72707
SVM (Gaussian)	0.95	0.04	20958
SVM (Sigmoid)	0.85	0.11	11992
SVM (Polynomial deg2)	0.95	0.03	49184
SVM (Polynomial deg3)	**0.97**	0.03	54403
FDA	0.91	0.01	767676
MLP†	0.93	0.49	603443
CNN†	**0.97**	2.31	249295

Table 1: Comparison between the algorithms. Training time and inference speed measured on an Intel i9-10940X CPU. MLP and CNN were implemented using PyTorch, and the rest were implemented using scikit-learn. †Allowed to use multi-threading in inference and training.

kernel with degree 3 and Convolutional Neural Network are the most accurate with 97% accuracy. However, there is a big difference between the two algorithms in terms of time. The training speed is about 80 times faster in Support Vector Machine with Polynomial kernel with degree 3 than in Convolutional Neural Network. The inference speed is about five times faster for Convolutional Neural Network. The difference in the training time is due to the fact that Convolutional Neural Network requires a lot more calculations compared to Support Vector Machine. On the other hand, because batch operations that process multiple samples at once are supported in Convolutional Neural Network, there is a difference in the inference speed. Therefore, Convolutional Neural Network and Support Vector Machine with Polynomial kernel with degree 3 are both efficient, depending on which one we focus on.

3.2 Translation Invariance

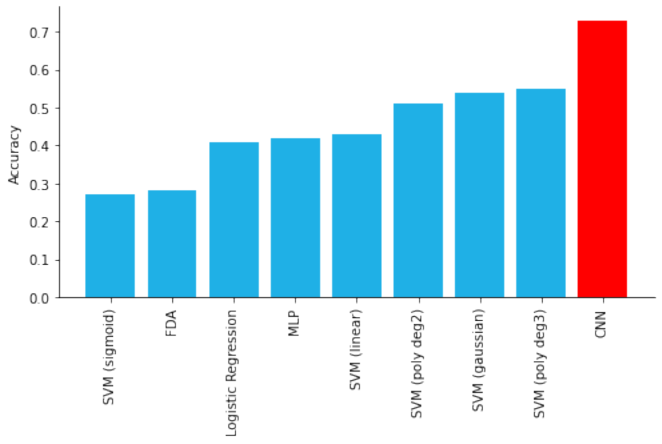


Figure 3: Accuracy of each model on translated validation dataset.

Translation invariance is one of the important things for computer vision tasks. Computer Vision problems are finding important information among given images. However, the information does not always appear in the same location in the image. It may be in the middle of the image or may be biased in one direction. Hence, being strong against translation means that the generalization performance of the model is excellent.

Fig. 3 demonstrates the generalization performance of each model for translation. Convolutional Neural Network has the most overwhelming performance compared to other models. On the other hand, Support Vector Machine with Polynomial kernel of degree 3, which showed similar accuracy to Convolutional Neural Network in the original validation dataset, significantly decreased performance in the translated validation set. Since not all models have experienced data that are not centered in the training process, these results imply that Convolutional Neural Network has better generalization power for computer vision.

4 Conclusion

In this paper we have compared some light-weight machine learning algorithms with two perspective: efficiency and translation invariance. We trained the models using only the CPU with a small dataset. In addition, we created translated validation set that is modified version of the original dataset to test translation invariance. As a result, in general, Convolutional Neural Network is the most suitable model for computer vision tasks. However, if the condition that all images are centered is added, Support Vector Machine is also good for using because it is efficient enough.