ENEE633 Project 2 Report: Hand-written Digit Recognition and Transfer Learning

Fengyu Liu (#116789028)
Department of Electrical and Computer Engineering, University of Maryland

Abstract

The aim of this project is to implement different conventional classifiers to achieve hand-written digits recognition, as well as performing transfer learning in an image classification task. In the first part of the report, I describe how to achieve hand-written digit recognition using linear/kernel SVM with LIBSVM and a convolutional neural network with caffe. In the second part, I complete the identification of different monkey species through transfer learning with tensorflow. In the last part, I show some of the experiment results, and discuss their advantages, disadvantages and the various parameters involved.

I. Introduction

For Part 1, the aim is to achieve hand-written digits recognition. The given dataset is split into a training set and a testing set. The corresponding labels are also given. Firstly, I implement different Kernels and compare the results. In order to reduce computing time, I use PCA and LDA to reduce the dimensionality. Then, I also build a Convolutional Neural Network using Caffe. The architecture is well known as LeNet, which is introduced in LeCun's paper in 1998.

In Part 2, the aim is to performing transfer learning in an image classification task. The data set contains ten species of monkeys, but the training data is limited. Thus, I use some pretrained models and perform transfer learning to improve accuracy. I have tried some different pretrained models such as VGG16, VGG19 and ResNet, and unfreeze some of the convolutional layers in the model. I also did some experiments with them and the results are shown in the end of this report.

II. Datasets

In this project, there are two datasets: MNIST and Kaggle/10-monkey-species. They are used in different parts.
(1) MNIST: It has a training set with 60000 28 x 28 grayscale images of handwritten digits (10 classes) and a testing set with 10000 images.
(2) Kaggle/10-monkey-species: The dataset consists of two files, training and validation. Each folder contains 10 subforders labeled as n0 - n9, each corresponding a species form Wikipedia's monkey cladogram. Images are 400x300 px or larger and JPEG format (almost 1400 images).

III. Part 1: Hand-written Digit Recognition

A. Linear and kernel SVM

Support Vector Machine (SVM) is a kind of algorithm that can separate data into two classes by a hyperplane. The corresponding optimization problem can be written as:

$$\text{maximize } f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i y_j c_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \le c_i \le C \text{ for all } i. \tag{2}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ could be

$$k(\mathbf{x}_i, \mathbf{x}_j) = x_i^T x_j \text{ (Linear)} \tag{3}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = (x_i^T x_j + 1)^r \text{ (Polynomial)} \tag{4}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{x_i^T x_j}{\delta^2}) \text{ (Polynomial)} \tag{5}$$

In the task of identifying Hand-written Digit, every digit is a class, so there are 10 classes. In order to turn this ten classes problem into a combination of two classes problems, I trained ten classifiers using all of the training data, and the task of the i'th classifier is to identify if the image is in the class of i-1. In other words, there are two possible outcomes for a data point in each classifier: 'should be classified to i-1' or 'shouldn't be classified to i-1'.

One problem of this method is that an image may be accepted by several classes at the same time, or it may be rejected by all classes at the same time. To solve it, I use LIBSVM toolbox and classify the image into the class with the highest

probability. LIBSVM is a very concise tool and the code in MATLAB can be limited to one line. Note that for different operating systems and different software, the LIBSVM function used is different, so it must be generated in the source file in advance.

What's more, in order to reduce the computing time, I reduced the dimensionality using PCA or LDA algorithm in advance. According to my simulation, reducing the dimensionality to about 10 can help us complete the calculation in a reasonable time (within about half an hour), and does not have much impact on the accuracy.

B. Deep Learning

LeNet is one of the earliest convolutional neural networks in the world. Using convolution, parameter sharing, pooling and other operations to extract features through clever design, LeNet can avoid a large amount of computational cost, and finally uses a fully connected neural network for classification and recognition. It is also the starting point of a large number of neural network architectures recently.

LeNet consists of several layers, including a data layer, some convolutional layers, sub-sampling layers, inner-product layers and an output (softmax) layer. Please see the figure below for its architecture.

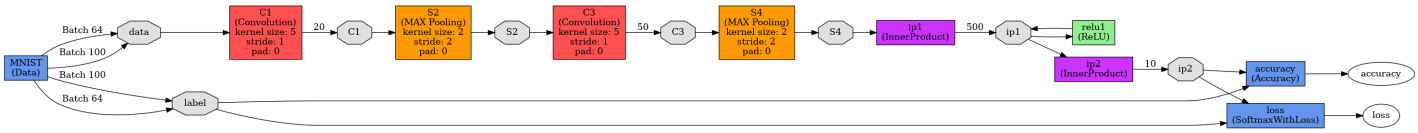

Fig. 1: Architecture of LeNet in my simulation (Generated by caffe)

1. Convolutional layer: The purpose of the convolution operation is to extract different features from the input data. The first layer of convolutional layer may only extract some low-level features such as edges, lines, and corners. More layers of networks can iteratively extract more complex features from low-level features.
2. Pooling layer: It is actually a form of sampling. There are many different forms of nonlinear pooling functions, of which Max pooling and average sampling are the most common. It is equivalent to converting a higher resolution picture into a lower resolution picture, and can further reduce the number of nodes in the final fully connected layer, thereby achieving the goal of reducing the parameters in the entire neural network. In general, I only use Max pooling in the simulation.
3. Full connection layer: It's the same as the method of ordinary neural network. Generally speaking, it's always in the last few layers.

IV. Part 2:Transfer Learning

Transfer learning, as its name implies, is to transfer the learned and trained model parameters to a new model in order to help the training of new model. When people are learning, they always apply their past experience and knowledge in similar fields. For example, when people study French, they can use their experience in learning English. The same is true for machines.

Considering that many data or tasks are related, we can use transfer learning to share the learned model parameters (also understood as the knowledge learned by the model) to the new model in some way to speed up and optimize The learning efficiency of the model.

When people are learning a new animal, usually only a few photos are enough. It's because people already have sufficient experience in this area. But this is not the case for machines. In this part, although the task of classing the species of monkeys is different from the task of distinguishing cats and dogs, the process could share many similarities, so they can also use experience to learn from each other. Thus, we can download a pretrained model for image classification (for example VGG, ResNet, InceptionNet) and only train the parameters of the last layer or a few layers. We can see that we will get more accurate results in less training.

V. Experiments

A. Results of Hand-written Digit Recognition

1) Hand-written Digit Recognition with SVM methods: The table below shows some data I got in my simulation of SVMs with LIBSVM. In PCA and LDA, I reduce the dimension into 10 in MATLAB.

SVM-Method	PCA-Linear	PCA-Polynomial	PCA-RBF	LDA-Linear	LDA-Polynomial	LDA-RBF
Accuracy	78.23%	91.70%	94.43%	87.12%	92.31%	93.09%

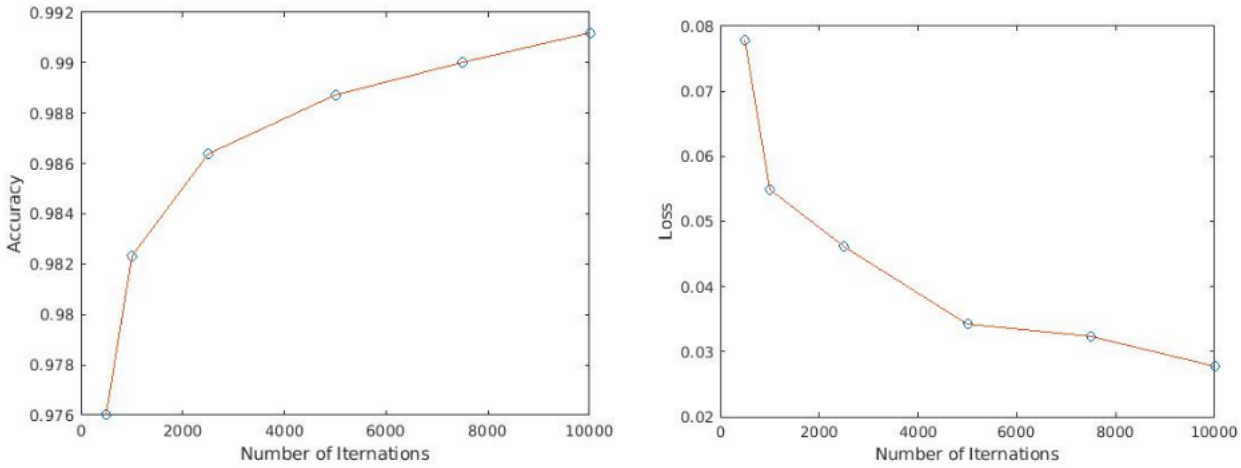

Fig. 2: Accuracy and Loss of the trained convolutional neural network with Caffe

2) Hand-written Digit Recognition with convolutional neural network: The table below shows some data I got in my simulation of convolutional neural network with Caffe.

# of iterations	500	1000	2500	5000	7500	10000
Accuracy	97.60%	98.23%	98.64%	98.87%	99.00%	99.12%
Loss	0.0777444	0.0549373	0.0460753	0.0341768	0.0323496	0.0277639

B. Discussion on the results of the Hand-written Digit Recognition

(1) All three kernels have reached a relatively high accuracy, but in each case, RBF always works better than linear and polynomial Kernels.
(2) LDA works better for linear and polynomial kernels, especially linear kernels. It might because it has the nature of linear classification.
(3) Convolutional neural network have much higher accuracy then SVM methods and speed up calculation effectively. In my calculation of SVMs, I reduce the dimension to 10 in advance but the program still takes about half an hour to give results. The process of dimensionality reduction also affects the accuracy of classification so it's hard to make the dimension smaller. For the neural network, the accuracy after 500 iterations has far exceeded the SVM method. It only takes a few minutes to finish if you use the CPU, and it may be faster if we use the GPU. This tells us that for image classification, convolutional neural networks are much more efficient.
(4) By increasing the number of iterations, the accuracy of the convolutional neural network will increase and the loss will keep decreasing. The accuracy can reach more than 99% in a reasonable calculation time.

C. Results of Transfer Learning

1) Without Transfer Learning: First, I write a simple convolutional neural network and train it with the training set. This neural network contains four convolutional layers and two pooling layers. The epoch is set to 50, and the batch size is set to 64. The results are shown below.

As we can see, when the number of epochs is near 50, the accuracy rate almost stops rising, but stays at about 0.7. This is because there is not enough data in the training set, thus we cannot achieve a high accuracy as in the previous part. The way to solve this problem is that we can use some models that have been trained in other dataset. Here, I use VGG19 as an example.

2) Transfer Learning with VGG19: The figures below show some data I got in my simulation. The pretrained model is VGG19 and only the last one layer or the last few layers are changed and trained. I set the batch size to 32 and the number of epoches is 10.

As we can see, although the VGG19 model cannot be used to identify the type of monkey initially, after one epoch, this classifier can get a surprising accuracy.

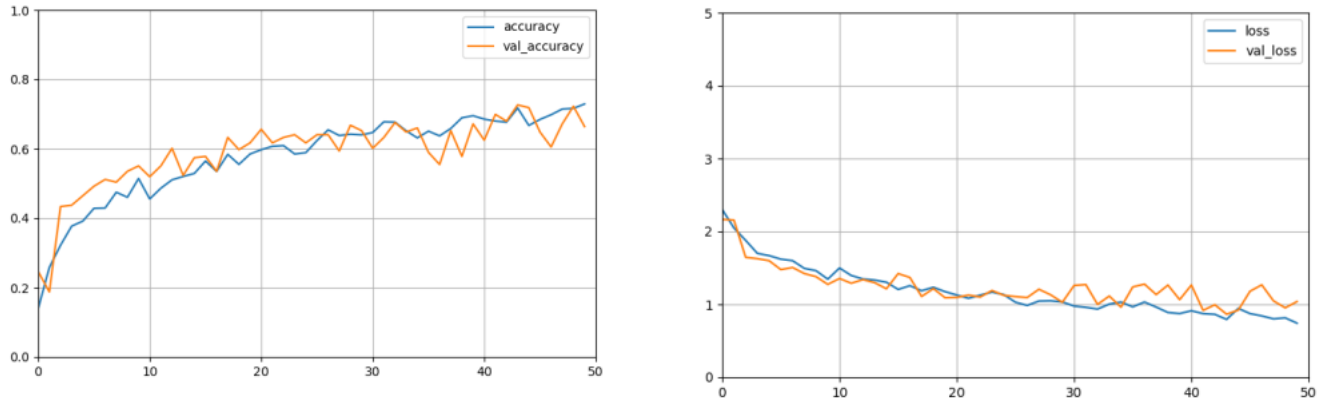

Fig. 3: Accuracy and Loss of a simple designed convolutional neural network without transfer learning

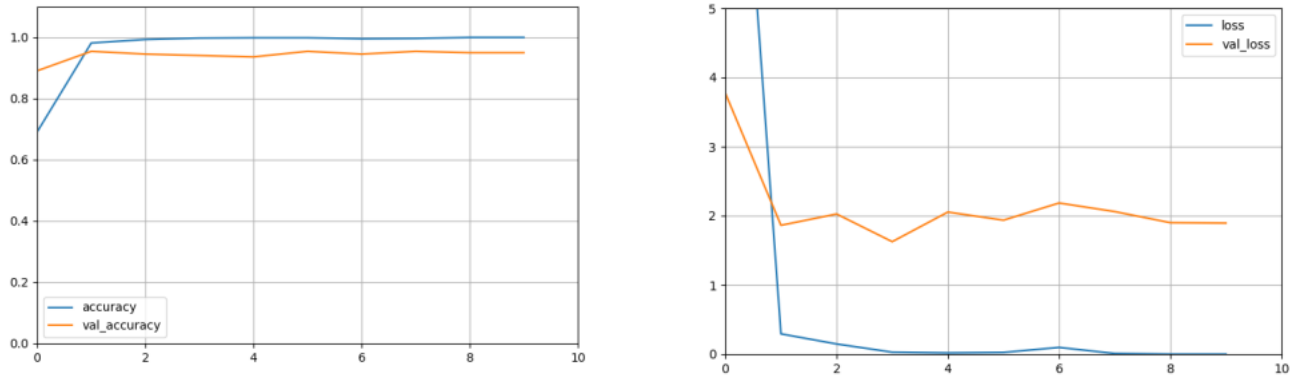

Fig. 4: Accuracy and Loss of the transfer learning with VGG19 model: Only the last fully connected layer is trainable. Learning rate: $lr = 1 \times 10^{-3}$ (default value)

D. Discussion on the results of transfer learning

(1) Although the accuracy of transfer learning may not be very high at the beginning, which means it cannot be used directly to class the pictures of monkey, it can achieve amazing accuracy in a short period of time (1-2 epoches), even if the training data is insufficient.
(2) The figure 5-8 shows the result of cases that more layers are unfreeze. Since the VGG19 model itself is very complicated, each layer of unfreezing will result in a large increase in the number of trainable parameters. Therefore, we need to consider a smaller learning rate.
(3) Proper unfreezing of several layers is beneficial to the improvement of accuracy.
(4) A very high learning rate will make the results to oscillate continuously, and the loss value may suddenly and greatly increase (see Fig. 5), while low learning rate may cause overfitting and slow convergence speed (see Fig. 7).

VI. Conclusion

In this project, I successfully used the SVM method and convolutional neural network (LeNet) to identify the hand-written digit, and used convolutional neural network and transfer learning to get the specie label of monkeys from images. The experimental results have shown that most of them are effective, but choosing the right method helps us to achieve much higher accuracy in much shorter time. The analysis of kernels showed that RBF kernel is the best choice in most instances, and the analysis of trainable layers and learning rates showed that we should choose the number of trainable layers and learning rate appropriately according to the size of the training set and the accuracy in the previous step. After many precise attempts and modifications, we can find the best parameters and obtain the highest accuracy with a small training dataset.

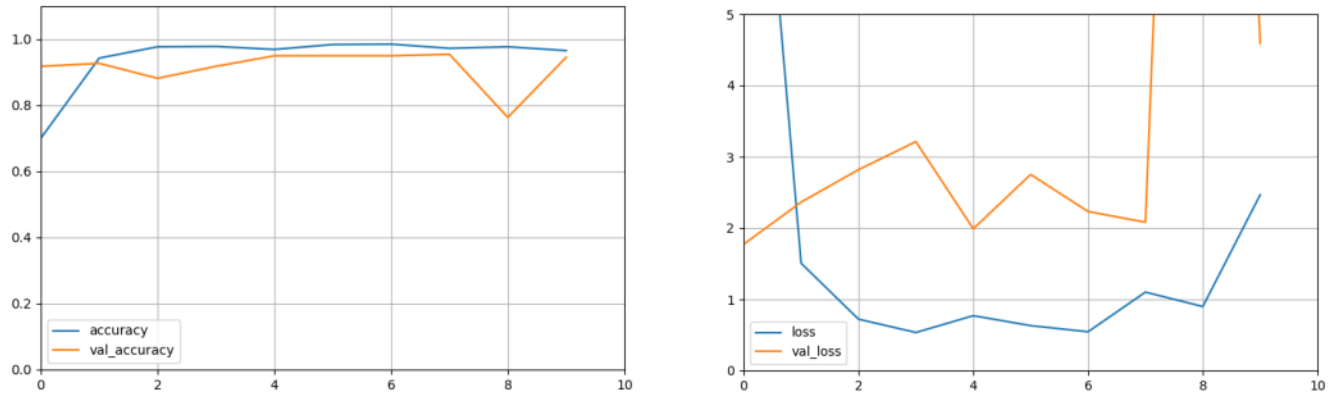

Fig. 5: Accuracy and Loss of the transfer learning with VGG19 model: The last fully connected layer and the last convolutional layer (-4 layer) is trainable. Learning rate: $lr = 1 \times 10^{-3}$. (default value), we can see lr is too large.

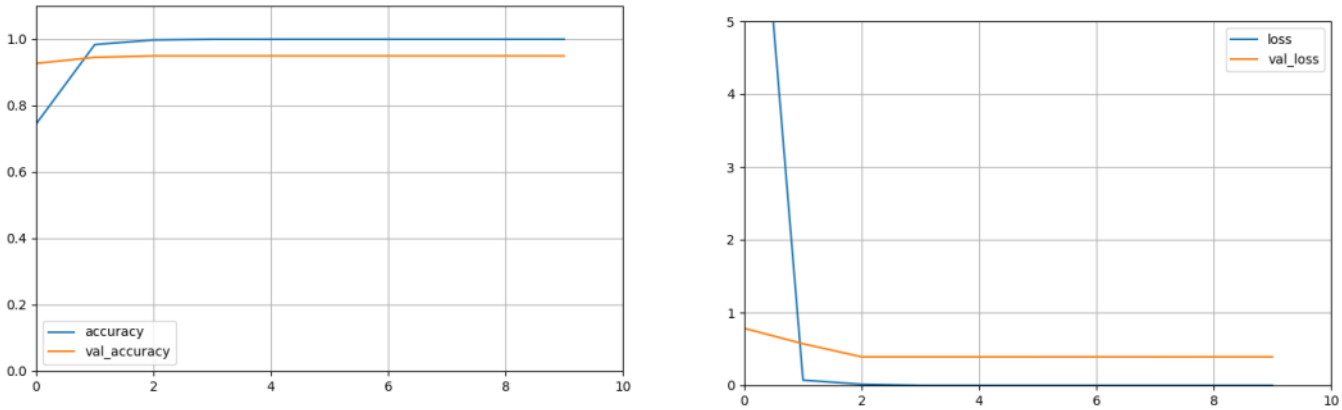

Fig. 6: Accuracy and Loss of the transfer learning with VGG19 model: The last fully connected layer and the last convolutional layer (-4 layer) is trainable. Learning rate: $lr = 5 \times 10^{-4}$. The final accuracy is about 95%.

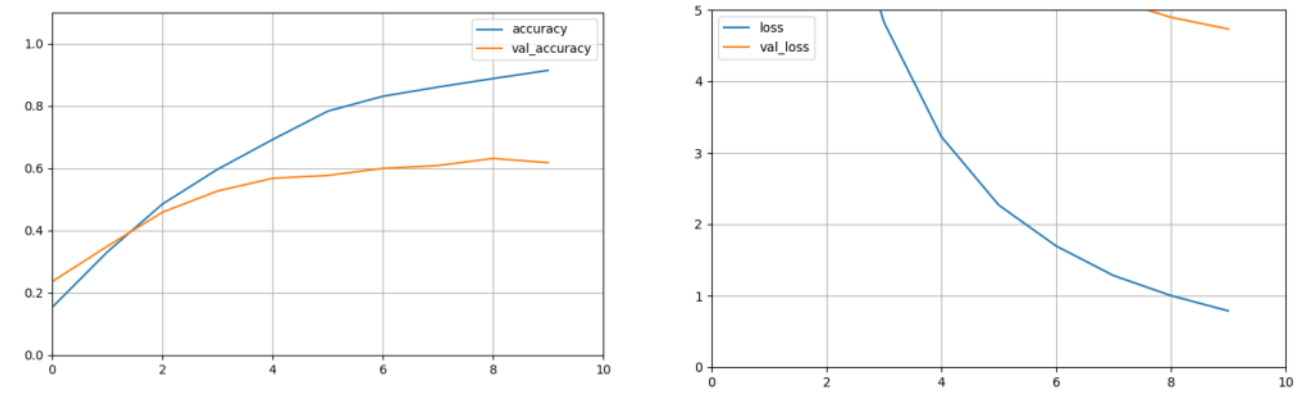

Fig. 7: Accuracy and Loss of the transfer learning with VGG19 model: The last fully connected layer and the last convolutional layer (-4 layer) is trainable. Learning rate: $lr = 1 \times 10^{-5}$. We can see it's overfitting, so lr is too small.

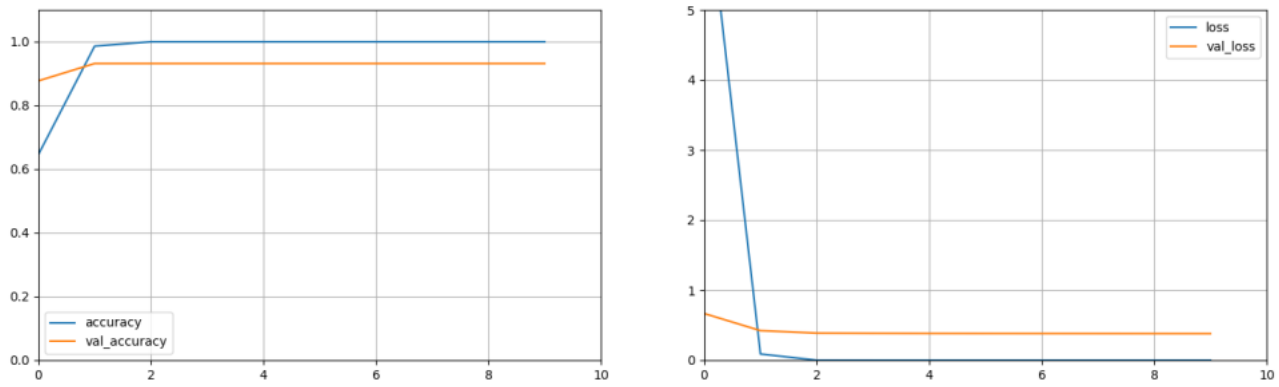

Fig. 8: Accuracy and Loss of the transfer learning with VGG19 model: The last fully connected layer and the last two convolutional layer (-4 and -5 layer) is trainable. Learning rate: $lr = 5 \times 10^{-5}$