Project Report

Xiaoli Qin, Binben Wang
xiq815, biw123

1 Introduction and Objectives

Support vector machine (SVM) is an approach for classification that was developed in the 1990s and has grown in popularity since then [3]. SVM is a traditional supervised machine learning method. It's Core idea is the principle of structural risk minimization. SVM is good for high dimension spaces and has good generalization ability. Compare with traditional neural networks,it is less likely to over fit.

The history of convolutional neural network (CNN) dates back to 1950s. The first modern CNN architecture was proposed by Yann LeCunn [6] in 1989. However, not lots of progress had been made so that people could apply those technologies to real world until 2000s. Thanks to developments of computer hareware and Internet, we can get cheap computation power and much larger labeled image datasets easily. CNN becomes more and more popular. In 2012, a team from University of Toronto won the ImageNet LSVRC-2010 contest by training a large, deep convolutional neural network called AlexNet [5]. And their error rates were considerably lower than the previous state-of-the-art [5].

We plan to re-implement a SVM and the AlexNet, and compare their error rates on CIFAR-10 [4] data set with the same amount of time. We can find which one is more efficient on image classification.

2 Details of Algorithms and your Implementation

2.1 Support vector machine

Support vector machine was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 [12]. For a linear SVM classifier, there is a linear mapping:

$$f(x_i, \mathbf{W}, \mathbf{b}) = \mathbf{W}x_i + \mathbf{b}$$

the matrix **W** and the vector **b** are the parameters of the function. x_i is the ith image feature vectors.$x_i \in R^D$ (with a dimension**D**), each associated with a label y_i. Here $i = 1...N$ and $y_i \in 1...K$. That is, there are N examples (each with a dimensionality D) and K distinct categories. Each row of **W** is a classifier for one of the categories.But we have to control over these weights so that the predicted class scores are consistent with the ground truth labels in the training data. We should define the loss function. For a Multiclass SVM loss, the SVM loss is defined as that the SVM wants the correct class for each image to have a score higher than the incorrect classes by some fixed margin Δ.Also we can extend the loss function with a regularization penalty R(**W**). The most common regularization penalty is the L2 norm. Thus the multiclass SVM loss becomes:

$$L = \frac{1}{N}\sum_i \sum_{j \neq y_i} [max(0, f(x_i; \mathbf{W})_j - f(x_i; \mathbf{W})_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$$

Making good predictions on the training set is equivalent to minimizing the loss. Recent algorithms for finding the SVM classifier include sub-gradient descent,coordinate descent,etc.

Data preprocessing for SVM

The first step of SVM is to transform the original color images into grayscale images. Due to the images are very tiny. They exhibited strong correlations between nearby pixels. When learning a statistical model of images, it might be nice to force the model to focus on higher order correlations [4]. This can be done through multiplying the data matrix by a whitening matrix. We used the ZCA whitening transformation to compute the whitening matrix.

Feature extraction

We evaluated many choices of feature extraction methods, such as computing features by directing convolution [9], local binary patterns (LBP) [8], histograms of oriented gradients (HoG) [2]. Finally, we found that HoG worked best among these methods. By comparing different sizes of block cell, size(4,4) is the best choice. After taking HoG for each image, the dimensionality of the HoG feature vector for each image is 2916. The dimensionality of feature vectors usually have an effect on the training time of LinearSVM classifier. Principal Component Analysis (PCA) is a good dimensionality reduction method [9].We investigated different scales of dimensionality reduction based on PCA.

Model training

In oreder to test the generalization ability of the SVM classifier, a 5 fold cross validation method is used. The original training data batches(total 50000 images) are divided into training set(total 40000 imgaes) and validation set(10000 images). If we say one training and validation process is an epoch, 5 fold cross validation is equal to running 5 epochs to train the classifier with different data. Finally the classifier with the highest validation accuracy will be chosen to predict labels of the test data.

2.2 AlexNet

AlexNet is an improved convolutional neural network (CNN), invented by Alex Krizhevsky, Ilya Sutskeverand Geoffrey Hinton [5]. AlexNet is very similar to another CNN architecture LeNet-5, which is a 7-level convolutional network proposed by Yann LeCun in 1998 [7]. LeNet-5 performs very well on the MNIST dataset with its 60000 handwritten digits. MNIST dataset was considered very large at that time. Thanks to developments of computer hareware, in particular GPUs, and Internet, we have more resources to get cheap computation power and much larger labeled image datasets.

I implemented our AlexNet in Python using TensorFlow.

Data Augmentation

Data Augmentation is an efficient way to enlarge training data to reduce the chance of overfitting. Label-preserving transformations is a common method to enlarge the dataset [5]. In our AlexNet implementation, we only flip training images horizontally and vertically, which triples our training data and preserves labels. In original AlexNet implementation, they also performed PCA on the set of RGB pixel values throughout the training set [5]. For simplicity, we just do not implement that.

Data Pre-processing

We only normalize over the whole dataset from each pixel to make sure that every values contribute to our convolutional neural network equally. Since all images in CIFAR-10 are 8-bit RGB images, we just divide all images by 255.

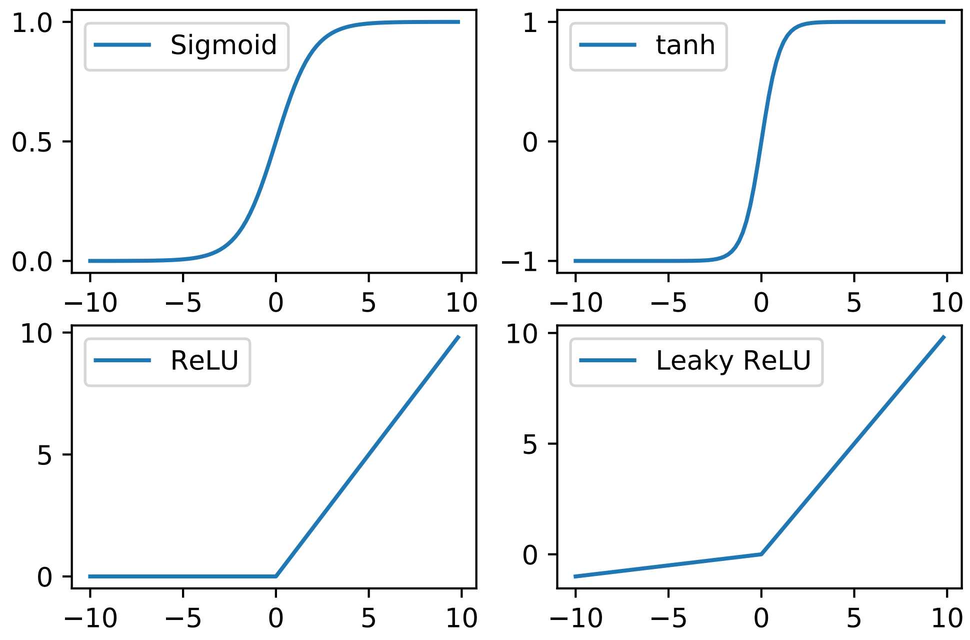

Figure 1: Sigmoid function, tanh function, ReLU function, and Leaky ReLU function.

Rectified Linear Unit Nonlinearity

Non-linear activation functions bring non-linearity to neural networks. Non-linearity helps us deal with much more complicated classification problems. If a neural network does not have any non-linearity, we can always combine all its layers into a single layer by matrix multiplication. There are many non-linear activation functions. Among them, Sigmoid function (1), tanh function (2) and Rectified Linear Unit (ReLU) (3) are very popular ones.

$$f(x) = \frac{1}{1 + e^{-1}} \tag{1}$$

$$f(x) = \tanh(x) \tag{2}$$

$$f(x) = \max(0, x) \tag{3}$$

Alex Krizhevsky et al. chose ReLU as the activation function of their AlexNet implementation. They found a 4-layer CNN with ReLUs required only 25% training epochs of the same CNN with tanh neurons to achieve same error rates [5]. Sigmoid function is not zero-centered. It may make the gradients of our weights all positive or negative in some special cases, which gives very inefficient gradient updates. Although tanh function does not have this problem, Sigmoid and tanh functions both have almost 0 gradient when $|x| \geq 5$. So gradients cannot be propagated back through neurons if inputs are too large or too small, which means weights and bias cannot learn from training data. For ReLU function, gradients keep the same if inputs are positive during back propagation. Also CNN with ReLUs converges much faster than CNN with sigmoids/tanhs. However, ReLU function is not zero-centered. And gradients become 0 if inputs are negative. Because of that, many modifications based on ReLU came out. Leaky ReLU (4) is one of them. It has all advantages of ReLU function and it is closer to zero mean outputs. And gardients will never become zero during back propagation.

$$f(x) = \max(0.1x, x) \tag{4}$$

Because ReLU function is good enough for our CNN architecture, we still choose ReLU as our activation funcion.

Local Response Normalization

These kinds of layers have a minimal impact and are not used any more. So we do not include these layers in our implementation.

Max Pooling

Max pooling is a tool to downsample our representations. It makes the representations smaller and more manageable. In our AlexNet implementation, we choose overlapping max pooling with a 4×4 filter and a stride of 2 in 1st, 2nd, and 5th convolutional layers. Alex Krizhevsky et al. found overlapping pooling is more difficult to overfit than non-overlapping pooling. The reason max pooling working is that if we know there is a high-value input, then the high-value input's relative location is more important than its exact location [1]. So we can simply drop other values in the filter except for the max value.

Dropout

Dropout randomly sets to zero the output of each hidden neuron with probability p [5]. It reduces the risk of overfitting and improves classification accuracy significantly [10]. We choose $p = 0.5$ in our implementation.

Training on a single GPU

GPUs are designed to compute the same instructionsin parallel, which is perfect for matrix multiplication and convolution. In 2005, some researchers implemented a 2-layer fully connected neural network on GPUs, which yielded over 3X speedup [11]. AlexNet was trained on GTX 580 GPUs originally. A single GTX 580 GPU has only 3 GB of memory, which is not enough for 1.2 million training examples [5]. So Alex Krizhevsky et al. decided to train their AlexNet across two GPUs. However, we can access much more powerful GPUs in 2018. So we train our AlexNet only on a single GPU. The GPU we choose is Tesla K80 on online platform FloydHub. Tesla K80 has 12 GB Memory, which is sufficient for training and testing our dataset. Also training AlexNet on a single GPU can eliminate communication time between two GPUs.

Overall Architecture

AlexNet has eight layers: the first five are convolutional layers and the rest of three are fully-connected layers. Because the size of images in CIFAR-10 is much smaller than the size of images in ImageNet dataset, we scale down our AlexNet architecture accordingly.

The first convolutional layer filters the $32 \times 32 \times 3$ input image with 32 kernels of size $4 \times 4 \times 3$ with a stride of 2 pixels following max pooling with a 4×4 filter and a stride of 2. The second convolutional layer filters the $8 \times 8 \times 32$ input image with 128 kernels of size $4 \times 4 \times 32$ with a stride of 2 pixels following max pooling with a 4×4 filter and a stride of 2. The third convolutional layer filters the $2 \times 2 \times 128$ input image with 256 kernels of size $4 \times 4 \times 128$ with a stride of 2 pixels. The fourth convolutional layer filters the $1 \times 1 \times 256$ input image with 256 kernels of size $4 \times 4 \times 256$ with a stride of 2 pixels. The fifth convolutional layer filters the $1 \times 1 \times 256$ input image with 128 kernels of size $4 \times 4 \times 256$ with a stride of 2 pixels following max pooling with a 4×4 filter and a stride of 2. The first two fully-connected layers have 1024 neurons each and the third layer has 10 neurons which are used to predict categories of input images (Figure 2).

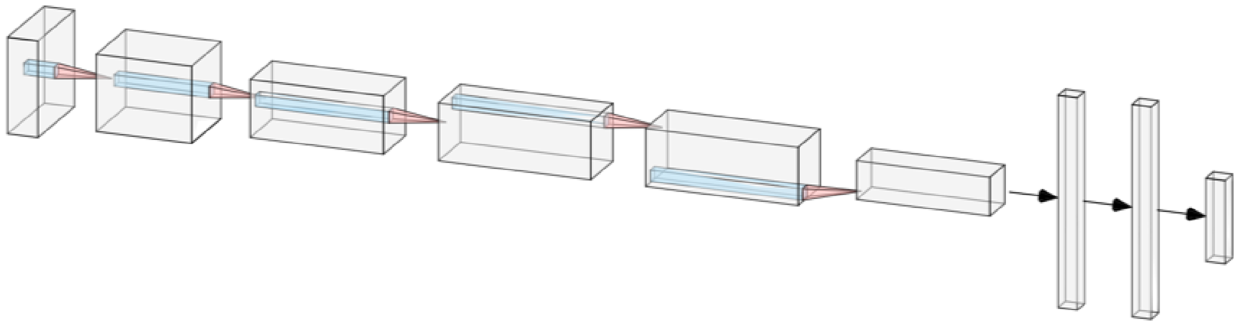

Figure 2: Architecture of our AlexNet implementation.

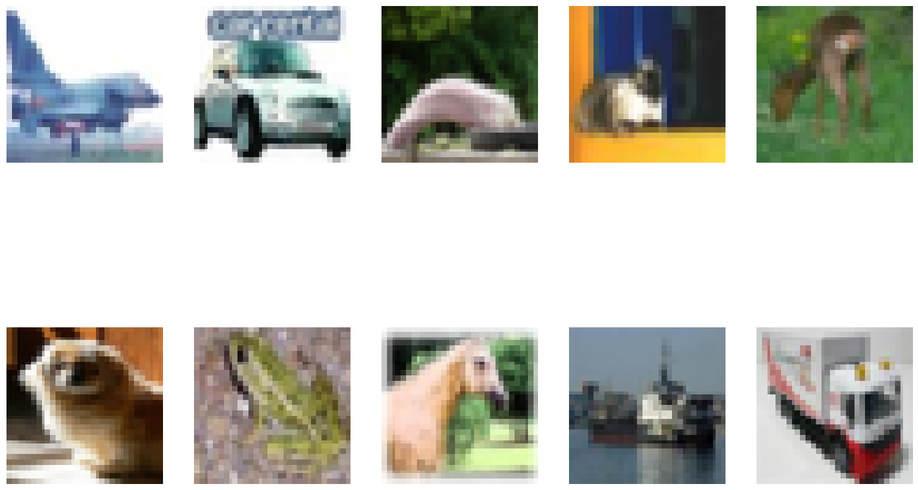

Figure 3: Image samples from left to right, top to bottom: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

2.3 The Dataset

CIFAR-10, after the Canadian Institute for Advanced Research, is a dataset of 60000 labeled 32×32 RGB images belonging to 10 classes [4]. The classes are *airplane*, *automobile* (but not truck or pickup truck), *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck* (but not pickup truck) [4] (Figure 3). Each class has exact 6000 images in it. All images were collected from Internet and labeled by humans. Each pixel of images is defined by a 3-D vector (r, g, b). And each component ranges from $0, 1, \cdots, 255$. CIFAR-10 is divided into five training batches and one test batch, each with 10000 images [4]. The test batch contains exactly 1000 randomly-selected images from each class [4]. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another [4]. Between them, the training batches contain exactly 5000 images from each class [4].

For the SVM, as we have mentioned above, 5 fold cross validation is implemented to select the classifier with the best generalization ability. The original training data batches(total 50000 images) are randomly divided into training set(total 40000 imgaes) and validation set(10000 images). The size of the testing images is 10000.

In the AlexNet implementation, we choose first 8000 images of each training batch as our training data, the rest of images in 5 training batches as validation data, and images of the testing batch as testing data. So we have 40000 training images, 10000 validation images, and 10000 testing images.

3 Experiments and Performance Evaluation Methods

1. We transform these colorful images into grayscale images. Next, we take the whitening for preprocessing.We randomly partition the original training samples into 5 equal sized subsamples. Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 4 subsamples are used as training data. The cross-validation process is then repeated 5 times, with each of the 5 subsamples used exactly once as the validation data. We compute the HoG descriptor for each image. The HoG descriptors of the training images are the input vectors of the SVM classifier, the category for each training image is the correct output. Finally we choose the trained classifier with the highest validation accuracy to predict labels of the testing set. Then we calculate the classification accuracy.

2. Due to the dimensionality of the HoG is descriptor 2916, which is high. We also take several different scales of dimensionality reduction on feature vectors. we reduce the dimensionality from 2916 to 1100, 800, 500 and 200 respectively by taking PCA and compare their training accuracy, validation accuracy, training accuracy and training time. Intel Xeon that has 2 Cores on online platform FloydHub The CPU for training the SVM classifier is same with the later ALexnet will use.

3. We train our AlexNet implementation on training data without data augmentation, training data with left-right-flipped data, and training data with up-down-flipped data. Then we compare their convergence rates and error rates to find whether simple data augmentation affects convergence rates and error rates. Because there are 80000 images in each epoch in the second and the third training sessions, we train the first session for 40 epochs and the rest two for 20 epochs to keep total number of training images the same. Also we use 40000 images as the x-axis unit instead of epoch.

4. We just keep one convolutional layer and second, third fully-connected layers to build another shallow CNN. Then we train both CNN architectures on training data without data augmentation. We can find out the effects of more convolutional layers and fully-connected layers.

5. We train AlexNet and SVM both on the same 40000 training images without any data augmentation. And then we compare their validation error rates on CIFAR-10 dataset with the same amount of training time. We also compare their testing error rates in the end. Since our implementation of the SVM does not have GPU acceleration so we train our SVM on a single CPU. And we train our AlexNet on a GPU and convert GPU training time to CPU training time based on the ratio of average CPU training time per epoch and average GPU training time per epoch.

4 Results and Discussion

1. From (Figure 4), it can be seen that the training accuracy and validation accuracy of the SVM are quite close in 5 five epochs. This shows that the SVM is classifier with good generalization ability and no overfitting occurs.

2. From (Figure 5) and (Figure 6), we can find that if we reduce the dimensionality of HoG feature vector to 1100 or 800, the variation of SVM training accuracy and validation accuracy is trivial.In particular, when the dimensionality is reduced to 800, the variation of validation accuracy is less

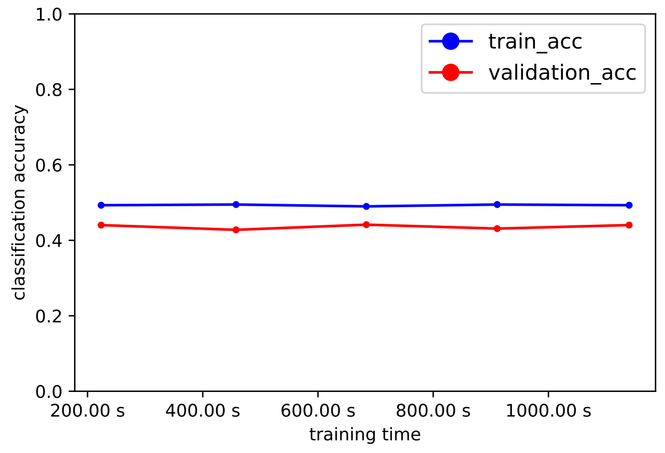

Figure 4: Training accuracy and validation accuracy of SVM

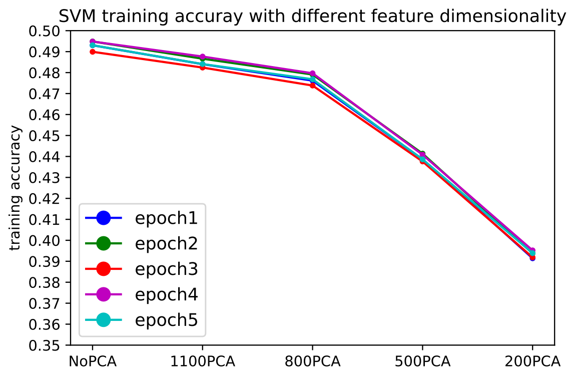

Figure 5: Training accuracy with different dimensionality reduction

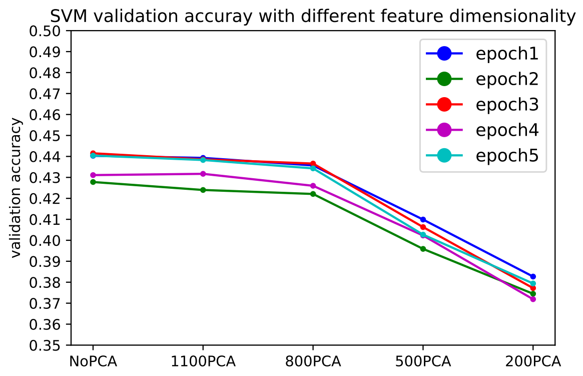

Figure 6: Validation accuracy with differentdimensionality reduction

than 1%. From (Figure 7) and (Figure 8), we can see that when the dimensionality is reduced to 800, the variation of testing accuracy is also less than 1%. But the whole training time decreases significantly. Without dimension reduction, the total training time on a CPU is about 20 minutes. However, when we reduce the dimensionality to 800, the training time is just around 3 minutes.

3. From (Figure 9), we can see that the second trained AlexNet has a slightly lower validation error rate than the first one. And it also converges a little bit faster and more stable because it has more training data, which indicate that appropriate data augmentation improves the performance of convolutional neural networks. However, AlexNet trained on training data with up-down-flipped images has a very high validation error rate. And we don't know why yet.

4. From (Figure 10), we can see that the shallow CNN converges faster and more stable. And it also has a slightly better performance on validation set than our AlexNet implementation. We think the reason is that CIFAR-10 dataset is not large enough to train AlexNet. And the zigzag of blue line can also tell that.

5. It took us around 20 minutes to train our SVM on a CPU and 15 minutes to train our AlexNet on a GPU which is equivalent to 2 hours on the same CPU with our SVM. From (Figure 11), Our AlexNet

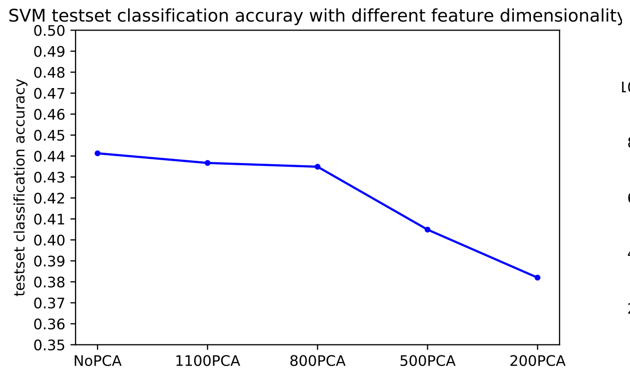

Figure 7: Testing accuracy with different dimensionality reduction

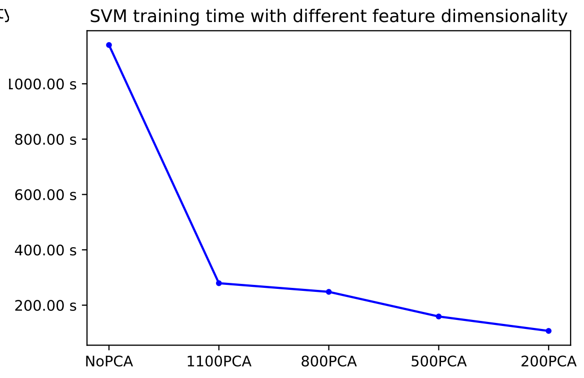

Figure 8: Training time with differentdimensionality reduction

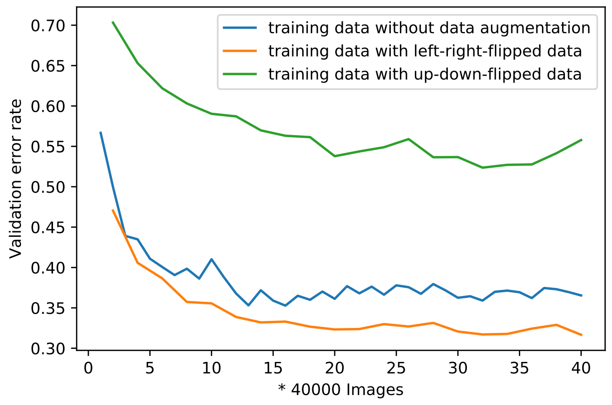

Figure 9: Train the AlexNet with data augmentation

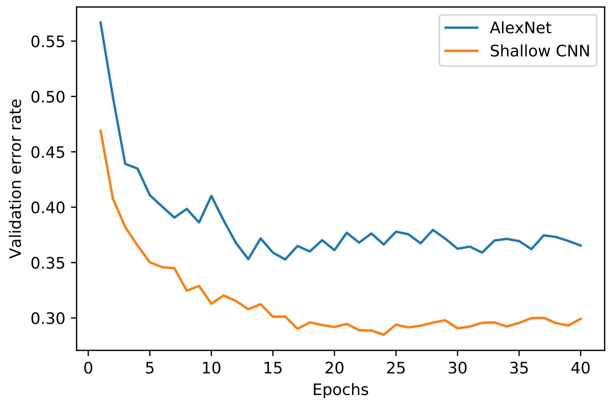

Figure 10: Comparison with a shallow CNN

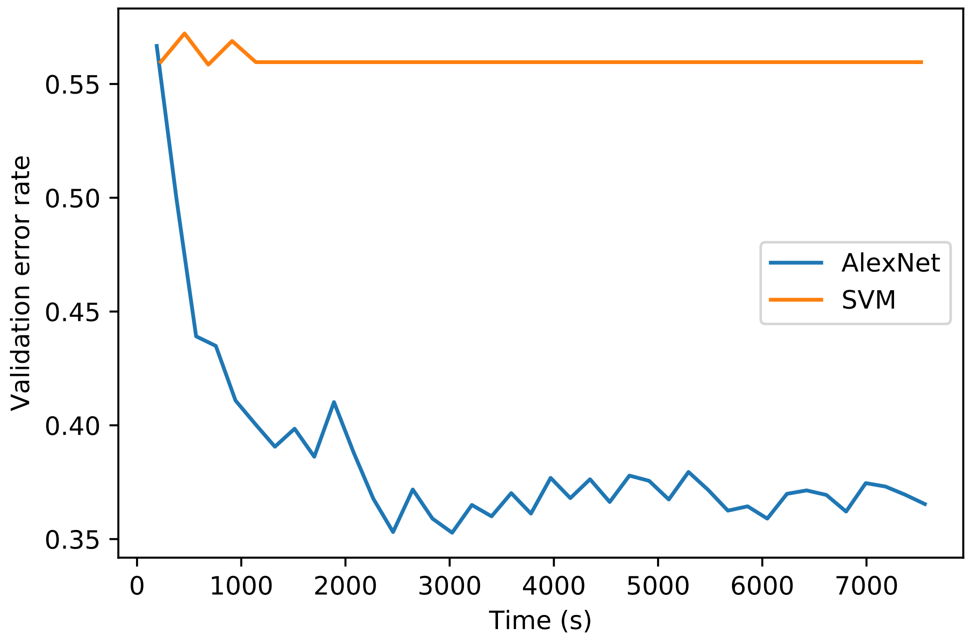

Figure 11: Comparsion between the AlexNet and a SVM.

implementation performs much better than SVM most of the time. It has around 35% validation error rate on the validation set. However, SVM has more than 50% validation error rate. The final testing error rate of HoG-SVM classifier is 55.87%, while the final testing error rate of Alexnet is 38.39%.

5 Conclusion

The final testing error rate of HoG-SVM classifier is 55.87%, model training time is about 20 minutes on a CPU. For the Alexnet,after 40 epochs, the final testing error rate of Alexnet is 38.39%,model training time is about 5 minutes on a GPU (2 hours in same CPU with SVM).Alexnet has a better potential to study complex patterns, when the training time is long enough.

Appendix A: Individual Contributions

Xiaoli Qin implemented the SVM and wrote related parts of the project report and Binben Wang implemented the AlexNet and wrote the related parts of the project report.

Appendix B: User Manual

Putting `cifar-10-python.tar.gz`, `svm_classifer.ipynb`, and `alexnet.ipynb` on the same directory, you can run `svm_classifer.ipynb` and `alexnet.ipynb` on Jupyter Notebook or PyCharm.

References

[1] Adit Deshpande. A beginner's guide to understanding convolutional neural networks part 2, 2016. [Online; accessed 4-December-2018].

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[9] Roberto Rigamonti, Matthew A Brown, and Vincent Lepetit. Are sparse representations really relevant for image classification? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1545–1552. IEEE, 2011.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[11] Dave Steinkraus, Ian Buck, and PY Simard. Using gpus for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120. IEEE, 2005.

[12] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, Sept 1999.