A Comparison of Classifications in Handwritten Digit Recognition – Final Report

Siyang Xue and Tiancheng Liu

1 Problem Description

Digit recognition has been a wide-spread and high-impact pattern recognition task, due to its simplicity: the background and the foreground (digits) are often well-separated, and the classes (categories) are limited to 10 (10 digits). Many researchers have benchmarked different methods on those datasets. Y. LeCun et al.[10] extracted their Mixed National Institute of Standards and Technology (MNIST) dataset from the National Institute of Standards and Technology (NIST) dataset, and made it normalized and available on their webpage[1]; they also provided early works on different classifications configurations in 1995, where linear classifiers, k-NN classifiers, and multiple variations of neural networks are investigated; a more complicated and complete comparisons can be found on the same webpage. A more recent work by CL Liu et al.[12] benchmarked methods of wider range on multiple datasets, on both feature extraction methods and classification methods.

What proposed is to further and more deeply investigate aspects of the digit recognition task. What we care about is how the different options in the classification pipeline affect the performance: *a*) How different normalization techniques influence the data quality, and how to provide a measurement (or at least, an intuition) on that; *b*) how SIFT [13], HOG [5] and other feature extraction methods perform on feature extraction/generation; *c*) what is the suitable dimension reduction method for the task; *d*) how the classifiers and the parameter setting affect the performance of the task.

The problem of digit recognition is a simple image classification problem, and the techniques tested on the dataset can be extended to used in other image classification problems. The computer vision community has shown a consistent interest on the digit recognition, not only because the datasets have relatively small size to validate the assumptions, and to test the new techniques, but also because the task alone is a simplified version of optical character recognition (OCR), which is also worth researching.

2 Data Description

The data we used is MNIST dataset[10], which is well-known and widely-used for digit recognition. All data are normalized beforehand – by bounding-box normalization and centering;

Some important information of the datasets is listed as follows:

- MNIST: Total training set of 60,000 images; test set of 10,000 images; all the digits are normalized to size 20×20 and put in the middle of a 28×28 image.

[1] `http://yann.lecun.com/exdb/mnist/`

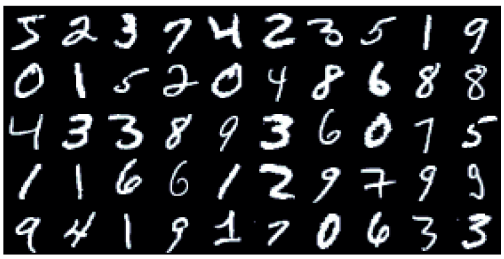
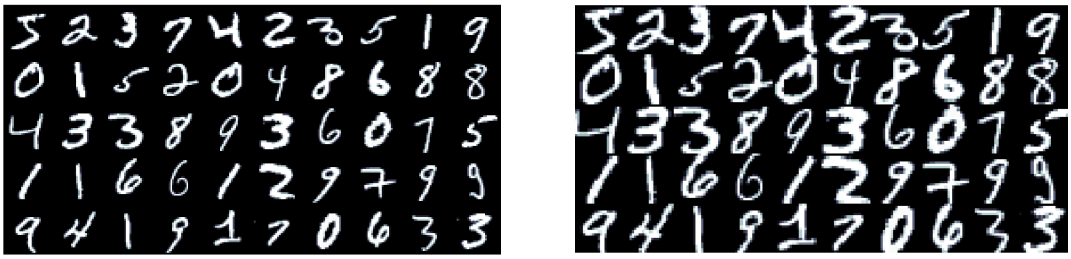


Figure 1: Digits from MNIST Dataset



(a) (b)

Figure 2: Normalization of the dataset: (a) MNIST(original) (b) MNIST(normalized)

In this final stage of our report, we have extended to work on the full set of MNIST dataset, that is, with 60000 digits as training samples, and 10000 digits as testing data. (In the 50% report we trained and tested on a randomly sampled subset of MNIST, with 600 digits from each subject(0-9) for training, and 100 digits from each subject for testing).

3 Methods

3.1 Preprocessing

The preprocessing step includes 3 sub-steps: normalization, feature extraction/generation, and dimension reduction.

3.1.1 Normalization

The normalization process removes the size variation and brightness variation of the digits. For the MNIST dataset, as suggested by [10], we removed the padding area to shrink each image from 28×28 to 20×20, which made the matching distance metric more precise and reduced the computational cost of further analysis. All the intensity data are normalized by shifting the mean value to zero, and also normalized by the standard deviation of the pixel values, which removes the brightness variance between different images of digits. The result of the normalization can be seen in Figure 2.

3.1.2 Feature Extraction

Feature extraction/generation methods based on gradient and local invariant feature detectors are investigated. To be specific, in this report, we investigated two common feature extraction strategy in the Computer Vision society: a) HOG features and b) Scale-invariant feature transform (SIFT) features.

HOG Features

Histograms of Oriented Gradients(HOG)[5][6] is a feature extraction method based on detection of local gradient distributions. Not only detection of edge positions, but also intensity normalization is included in this method, making it invariant to the shadows or illumination difference of the sample image. A lot of HOG implementations are proposed, with different parameter and slightly differently designed histogram calculating schemes. In this project, we will focus mainly on a Felzenszwalb et al.[6] version of HOG, for its relatively fewer features in selecting, as we only care about the distinguish from 0 to 9 in handwritten digits recognition, and too many features could result in a input whose dimension is too large for the following classifier.

First, in order to identify the local edge orientation and intensity, the finite difference filter $[-1, 0, 1]$ and its transpose, are applied to the image to calculate its gradient in 2 directions, the resulting gradient image is then divided into cells of 4. In each cell, a histogram of the gradient orientation is created by dividing the orientations θ of the gradient evenly into 9 bins, and each vote has the weight of r(i.e. the magnitude of the coordinate pixel). Therefore, for a $w \times h$ image, a $w/4 \times h/4 \times 9$ cell feature map C is created.

Then, in order to achieve an invariance to local gradient bias, for each element of the feature map $C(i, j, :)$, we normalized it together with its 8-neighbor elements and create a 4×9 matrix.

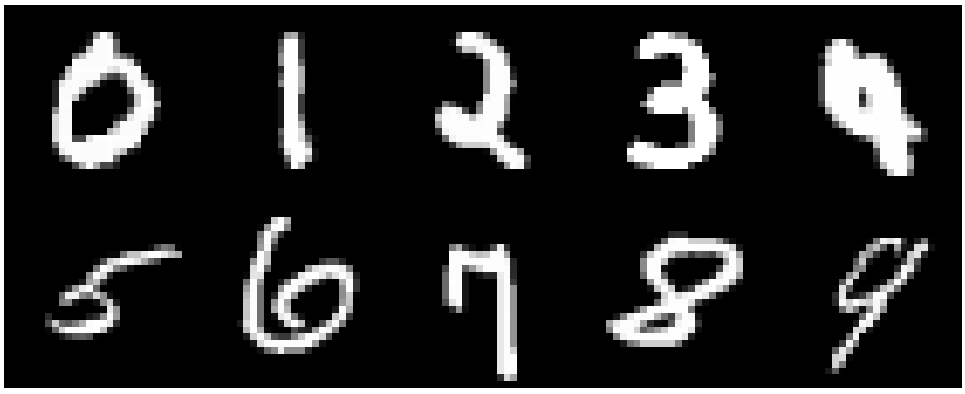
$$HOG(i,j,:,:) = \begin{bmatrix} C(i,j,:)./N_{-1,-1}(i,j,:) \\ C(i,j,:)./N_{-1,+1}(i,j,:) \\ C(i,j,:)./N_{+1,+1}(i,j,:) \\ C(i,j,:)./N_{+1,-1}(i,j,:) \end{bmatrix}$$

$$N_{\delta,\gamma} = (||C(i,j,:)||^2 + ||C(i+\delta,j,:)||^2 + ||C(i,j+\gamma,:)||^2 + ||C(i+\delta,j+\gamma,:)||^2)^{\frac{1}{2}} \quad (1)$$

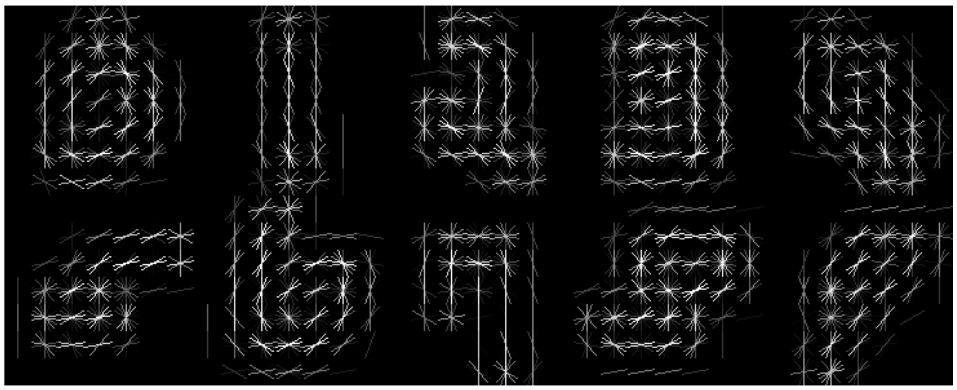
[6] Each matrix is then reshaped into a 36 dimensional vector, resulting a overall $w/4 \times h/4 \times 36$ feature map.

Finally, the now normalized feature map is compressed to 31 features per cell, using PCA projecting the features to 31 most variant dimensions in the $w/4 \times h/4$ features of the image. Here, we choose the reduced dimension to 31 according to the singular value decomposition result, in which the last 5 singular values approximate to 0. This step is different from the later dimension-reduction step of all the images in the training set in that it calculates the

principal components within the same image and is aimed at eliminating the specific minor noises of a particular image.



(a)



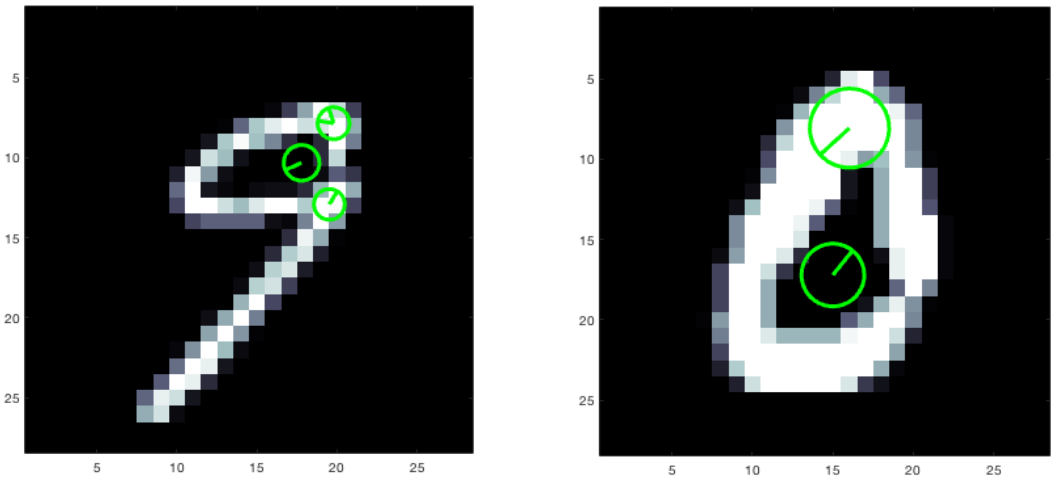
(b)

Figure 3: (a)handwritten digits 0-9 from MNIST (b)HOG features of handwritten digits 0-9

The original images, and the HOG features of digits from 0 to 9 are shown in Figure 3(a) and Figure 3(b), respectively. As shown in the figures, the HOG features well describe the outlines of the digits. A digit written with a thinner line(such as the '9' in Figure 1) does not make much difference in the HOG feature description with digits written with thicker strokes, making the features resistant to the irrelevant factors.

SIFT Features

Scale-invariant feature transform is a popular feature extraction technique introduced by David Lowe in his 2004 paper [13]. The process consists of typically three steps: a) Detection of scale-space extrema, where image pyramid is generated using a set of different-scaled Gaussian kernels, and the extrema is detected in a spatial-hierarchical neighborhood; b) Accurate keypoint localization, where the extrema of low contrast and extrema near the



(a) (b)

Figure 4: Handwritten digits with their SIFT features: (a) A handwritten 9 (b) A handwritten 0

image boundary are discarded; c) Orientation assignment, where each keypoint is assigned several orientations according to its spatial neighbors.

The SIFT features are often applied in object detection, face recognition, and other related tasks [17, 5], due to its robustness against image scaling, image rotation, and certain kinds of noise. In these applications, the SIFT features (keypoints) are extracted, and oftentimes, it is followed by extracting the HOG features of the spatial neighborhood of the keypoints, to generate a feature descriptor [13].

However, this feature extraction strategy is ineffective for the MNIST dataset. In the experiment, we found all the images have only a few feature points to be extracted (≤ 4), which is insufficient to describe the image (see Figure 4); some image even failed on SIFT, that means it have no SIFT feature to extract. Furthermore, it may not be a desirable choice for digits dataset, where we have 6 and 9 with the same topology, and the rotation invariant property of SIFT would made it impossible to distinguish the above 2 digits. Therefore, in this report, we do not use SIFT as a feature extraction technique, and only HOG is used to generate feature descriptors.

However, for further study, if we are to investigate faces, SIFT would be a good choice for extracting the keypoints in a more complex context, and the the invariant properties of the SIFT would help the recognition.

3.1.3 Dimension Reductions

The dimension reduction methods used in this report can be categorized into 2 groups: linear and nonlinear. For linear method, we apply the conventional PCA; both kernel method (Kernel-PCA) and manifold learning method (LLE) are investigated as nonlinear methods. Experiments are conducted to compare the effectiveness and efficiency of the methods.

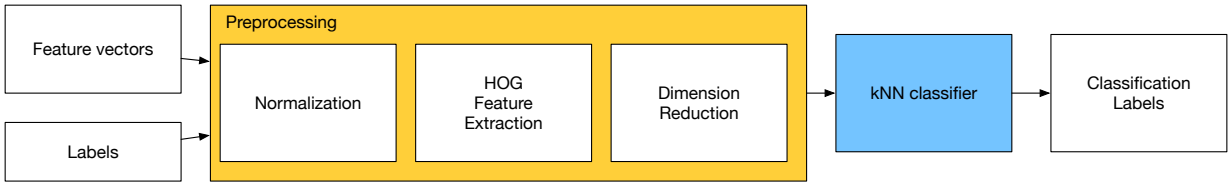


Figure 5: The classification pipeline with kNN classifier

3.2 Classification

Three different classification methods are used in this report. k-NN[1] classifier is used for investigating the dimension reduction methods; several experimental comparisons based on k-NN classifiers are conducted to highlight the difference of the performance, and to give a clue about what are the desirable preprocessing methods for digit recognition tasks. The SVM classifiers are reported as desirable classifiers of the task because of its high-accuracy, flexibility in its kernel choices, and also low-computational complexity [15, 2, 12]; on the other hand, the SVM classifiers are sensitive to kernel choice, and in terms of feature transformation, the "kernel tricks" that we used in the dimension reduction methods are already transformed the feature space according to the kernels, and it might not be desirable to use the kernels a second time. In early literature, neural networks are proposed to solve the problem of handwritten digits recognition, and the accuracies are comparable to SVM classifiers [8, 11] despite its high computational cost in training process. In recent years, the trend of deep learning brings back convolutional neural network (CNN), and the state-of-art results are obtained by CNNs [7, 4, 3].

3.2.1 k-NN Classifier

The k-NN[1] Classifier we used in the project is based on euclidean distance ($l2$-norm), such that for any two feature vectors $\mathbf{x}$ and $\mathbf{y}$, the distance between are defined as

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2 \tag{2}$$

The performance of the classifications are measure by the overall accuracy (OA), that is, the ratio of corrected labeled feature vectors (digits) in all feature vectors. The pipeline that we used with k-NN classifier can be seen in Figure 5.

In this report, in order to highlight the effect of the preprocessing steps, we have different settings of parameters, the parameter settings are as follows:

- With/without normalization
- With/without HOG feature extraction
- Dimension Reduction: {None, PCA with kernel: Linear, Polynomial, Gaussian, LLE}

Thus we have a total of $2 \times 2 \times 4 = 16$ different classification instances.

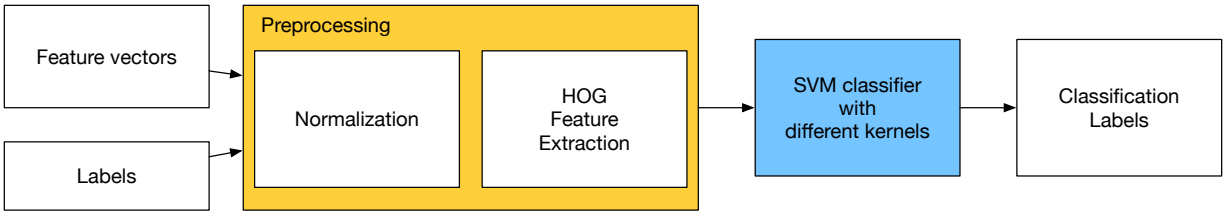


Figure 6: The classification pipeline with SVM classifier

3.2.2 SVM Classifier

Support vector machine (SVM)[8, 11] finds the hyperplane that best seperates the two classes in the high-dimensional space. According to representer theorem, the solution of the optimization problem can be expressed in another space by a linear combination of inner products defined in that space. That is, the "kernel trick" allows us to tranform the feature space into another Rreproducing Kernel Hilbert Space (RKHS), and hence find the nonlinear boundary between the classes in the original feature space. The performances of the SVM classifiers are sensetive to the kernel types and parameters.

In this report, we investigate 3 different kernel of SVM classifiers in this report: linear kernel, polynomial kernel, and radial basis function (rbf) kernel. For the polynomial kernel, we tried different values of degree d, where $d \in \{2, 3, 5, 7\}$; for rbf kernel, we tried the standard deviation $\sigma \in \{3, 4, 5, 7\}$. As mentioned before, the SVM classifiers transformed the features by theirselves according to the kernels, and thus the classification pipeline with SVM classifier is shown in Figure 6.

A somewhat related method called relevance support machine (RVM) [16] is also investigated. RVM has the same function form as SVM, but generally need much more data and tend to get local minima, because it is based on EM methods. On the subsampled training set and testing set (6000 training samples and 1000 testing samples), RVM does not provide better generalization than SVM (87% OA compared to 93% OA). It also has higher computational cost compared to SVM, and we therefore were not able to run the RVM on the whole dataset.

3.2.3 CNN Classifier

Convolutional Neural Networks(CNN)[7] intuitively pulls its layers within the receptive fields of each units, just like extracting features from lower to higher levels. This hierarchy enables CNN to capture the overall topological properties of the objects more easily and improves the performance of recognition incredibly[10]. The organization of CNN is built in the following steps:

1. For each layer of the network, let the input image I be of $n \times n$ pixels, and let the receptive fieild of each pixels be a square of $r \times r$ (here, R is a small odd number), let K_i represents the $r \times r$ kernels;

2. Calculate the convolution of I and $K_i(i = 1, 2..., m)$, resulting m images of $(n - r + 1) \times (n - r + 1)$;

3. Down-sample the images with stripe s, the output of this layer is built up with the resulting m $(n - r + 1)/s \times (n - r + 1)/s$ images;

Preprocessing In order to fit the input image in the overall architecture of the convolutional networks, each 28×28 input image is put in an all-zero matrix of 32×32(details in the following section).

In this report, we design two architectures of CNN, one is based on the Lenet [10] and the other is modified according to our experimental results. In the following sections, we will analyse and compare the architecture, training methods and performances of these two architecture of CNN.

Overall Architecture As shown in Figure 7 and 8, both neural networks is built up of 3 layers of convolutional networks and several layers of fully-connected neural networks.

Original Architecture The initial input of the network is a 32×32 matrix, and with 6 convolutional 5×5 kernels in layer 1, and down-sampling stripe 2, the resulting output are 6 14×14 images. Similarly, the number of 5×5 kernel for layer 2 is 16 and the down-sampling rate is 2, resulting in 16 5×5 images. Finally, with no down sampling and 120 5×5, the output of layer 3 is exactly 120 1×1 units.

The following 2 fully connected layers acts as hierarchical classifiers: 84 sub-features are first selected from the 120 features, and are then classified into 10 classes of digit 0 to 9.

The kernel size 5×5 is chosen, as this size would best characterize the local features, without too much overlapping, but still have padding between the neighbor output, and the number of kernels are chosen roughly empirically, for fewer kernel features will affect the performance while an increase in the number of features leads to no obvious improvement.

Modified Architecture Based on the performance of the previous architecture of CNN, we descovered that the model tends to overfit because of too many parameters in the 2 fully-connected layers. As extending the training set would be impractical for our experimental condition currently, we removed one of the 2 fully-connected layers and add more dimensions to the convolutional kernels, which could be viewed as changing the overall classifier to a more feature based model.

As shown in Figure 8, the modified architecture is made up of 10 layers: layer 1, 4, 7 are convolutional layers with convolution kernel of 5×5, $5 \times 5 \times 20$ and $4 \times 4 \times 50$ respectively. Between the 3 convolutional layers, there are 2 maxpooling layers with strip 2. Maxpooling works similar to subsampling but pools the local maximum in each 2×2neighbour, and has been tested to be more proper in selecting invariant features in convolutional networks [**?**].

A batch normalization[**?**] layer is also added after each of the 3 convolutional layers. This layer operates across the whole batch of image maps(in our experiment, each batch

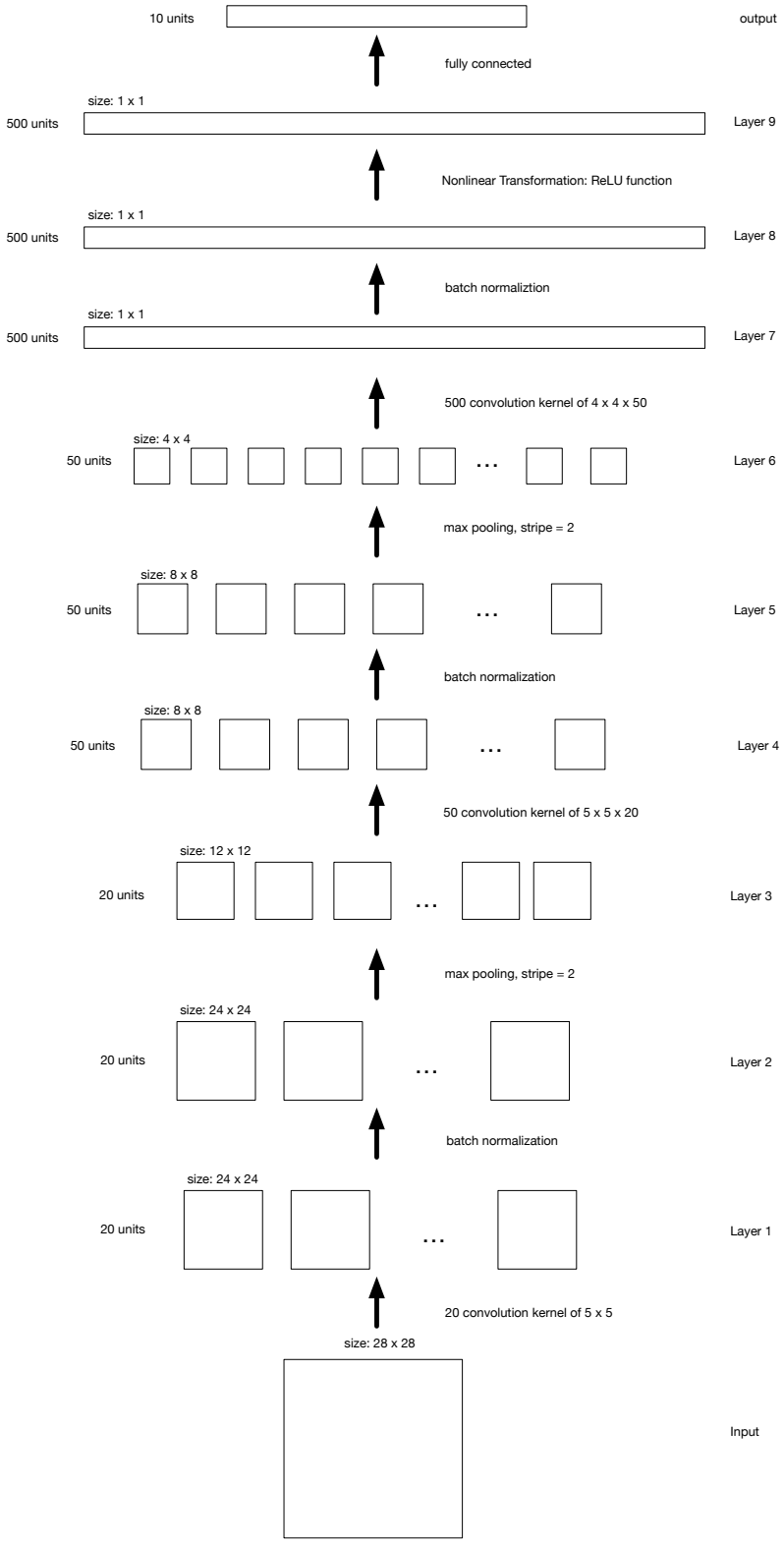


Figure 7: The architecture of CNN(Lenet model)

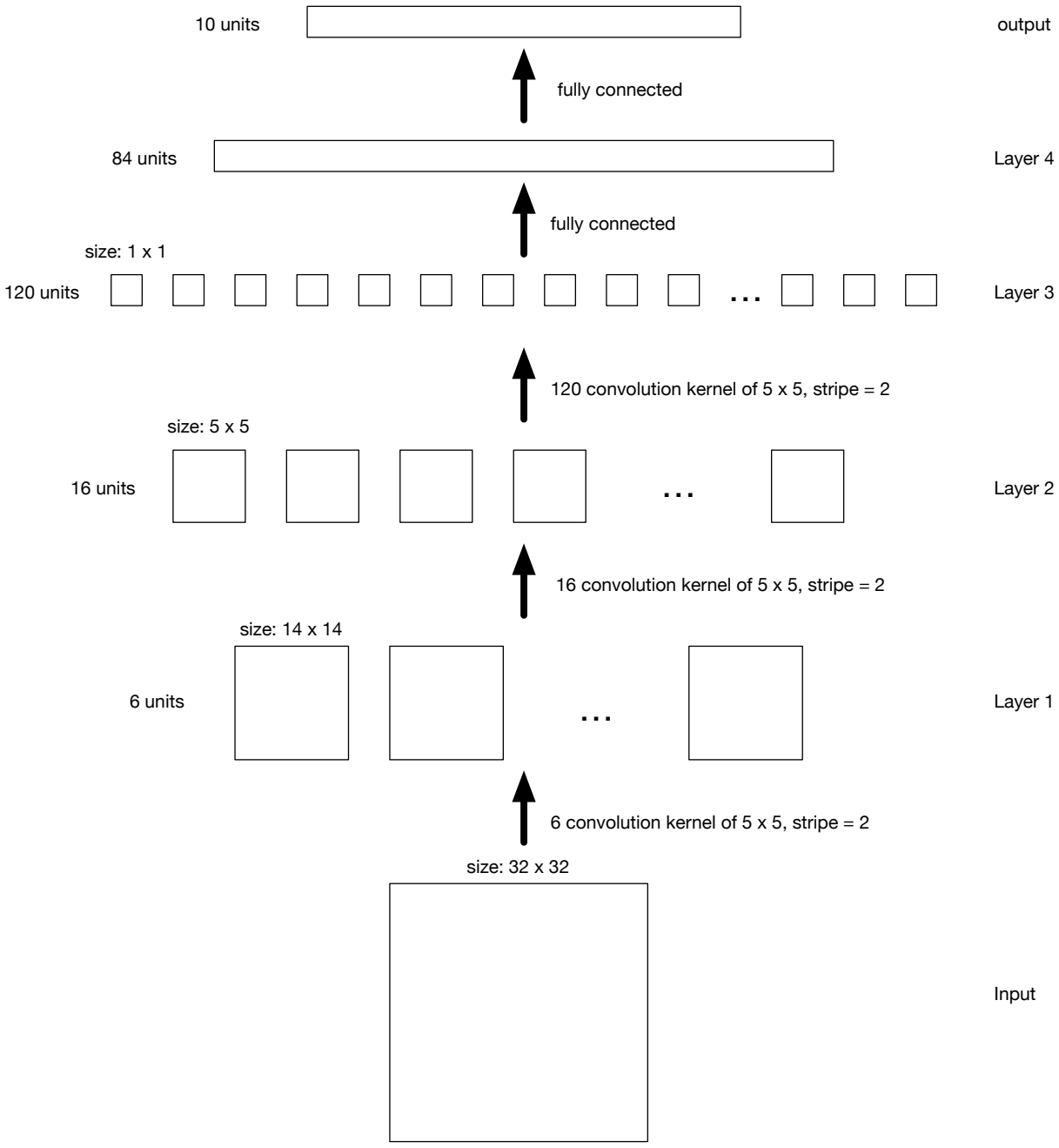


Figure 8: The architecture of CNN(Our model)

contains 100 input images), and output the 100 feature maps normalized with their means and deviations.

$$y_i = w\frac{(x_i - \mu)}{\sqrt{\sigma^2 + \epsilon}} + b, \quad \mu = \frac{1}{100}\sum_{i=1}^{100} x_i, \quad \sigma^2 = \frac{1}{100}\sum_{i=1}^{100}(x_i - \mu)^2$$

Finally a ReLU activation function layer is added before the fully connected layer to make sure all the inputs are non-negative.

Training The training of the CNN follows the back-propagation learning scheme[9], which fits the input with the standard output by minimizing its training error.

In the LeNet version of CNN, we calculate and update the parameters with each input images, which means the training process is extremely time-consuming. (We reference the original training parameter to the work of Patrice Y. Simard [14])

In our modified version of CNN, with more dimensions in the convolution kernels, training complexity could be higher. In order to solve this problem, we introduce a batch-based training method to increase the computational efficiency:

- Randomly divide the training set into batches containing 100 training samples each.
- For each batch, calculate the derivative of each input for each layer on the same parameter.
- Update the parameters of CNN with back-propagation, using the derivatives of averaged loss function over each parameter.
- Keep the already trained net parameters and repeat the previous steps 20 times.

4 Results

5 Results

5.1 k-NN classifier

In this section, we discuss the differences of the performances of the classifier with input types: features without selection, HOG features selected with raw data, and further HOG features selected with normalized data over the several dimensional reduction methods. Running time and classification accuracy are both included in the measurement of the performance. The results can be seen in Figure 9 and Table 1.

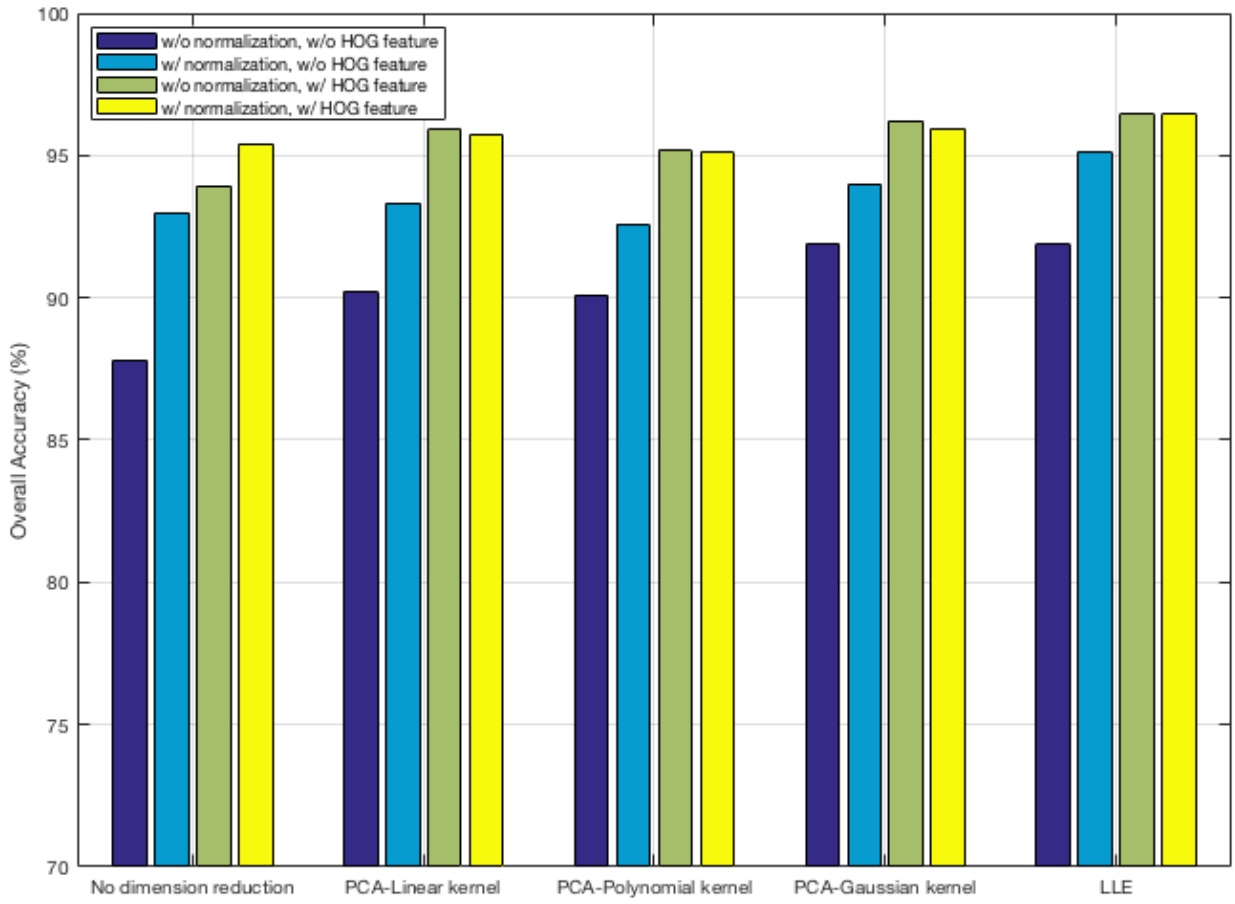


Figure 9: The result of digit recognition of classification pipeline with kNN classifier

Table 1: Accuracy and running time of classification pipeline with k-NN classifier on average

	Normalization	HOG Feature	No Dimension Reduction	Kernel PCA			Locally Linear Embedding (LLE)
				Linear	Polynomial	Gaussian	
Overall Classification Accuracy (%)	w/o	w/o	87.80	90.20	90.10	91.90	91.90
	w/	w/o	93.00	93.30	92.60	94.00	95.10
	w/o	w/	93.90	95.50	95.20	96.20	96.50
	w/	w/	95.40	95.70	95.10	95.90	96.50
Running Time (s)	w/o	w/o	2.50	5.72	76.68	80.20	38.95
	w/	w/o	0.50	0.61	73.50	76.91	18.71
	w/o	w/	14.70	17.01	99.61	107.39	101.39
	w/	w/	7.24	7.98	92.52	101.76	45.01

5.1.1 Comparison on Normalization

As can be seen in Table 1, both the HOG feature extraction and normalization process raise the performance of classification; however, the normalization process with HOG feature extraction does not always outperform the pipeline with HOG features alone regrading the overall accuracy, but at the same time, the running time of the former are always less than the latter, which demonstrates that the computational complexity is reduced.

5.1.2 Comparison on Feature Selection

As can be seen in Table 1, basically a increase of accuracy of 4% to 7% is achieved by selecting HOG features. Better edge detection and shadow invariant property of the HOG features might contribute to this increase. The increase is especially large in the case of no dimension reduction (7.6%) as the classification performance is more dependent to the feature quality with no further dimension reduction step. The running time is longer with feature selection, as the original dimension of the data is 784, and the selected feature dimension here is 775, so there is no much decrease in the dimension reduction process, and the increase in computational complexity comes from the HOG feature detection step. This increase, however, is minor (less then 10 percent) compared to the accuracy improvement.

5.1.3 Comparison on Dimension Reduction Methods

The dimension reduction methods we used for this project are kernel-PCA and Locally Linear Embedding (LLE). While we include PCA with linear kernel in the comparison, the comparison include 3 different scenes, while the last scene can also be divided into 2 groups:

- No dimension reduction
- dimension reduction with linear assumption: PCA with linear kernel
- dimension reduction with nonlinear assumption:
  - kernel-based dimension reduction: PCA with polynomial kernel and PCA with gaussian kernel
  - manifold-based dimension reduction: LLE

As we can see in Figure 9 and Table 1, we can seen that LLE is the desirable dimension reduction method in the task, because it reached a high accuracy, and also of relative low computational cost compared to kernel-based nonlinear dimension reduction methods.

5.2 SVM classifier

SVM classifier is also investigated in this report, we apply the SVM classifier on the HOG features. The reason that we do not use the SVM on the dimension reduction is that the kernel of SVM itself already transformed the feature space.

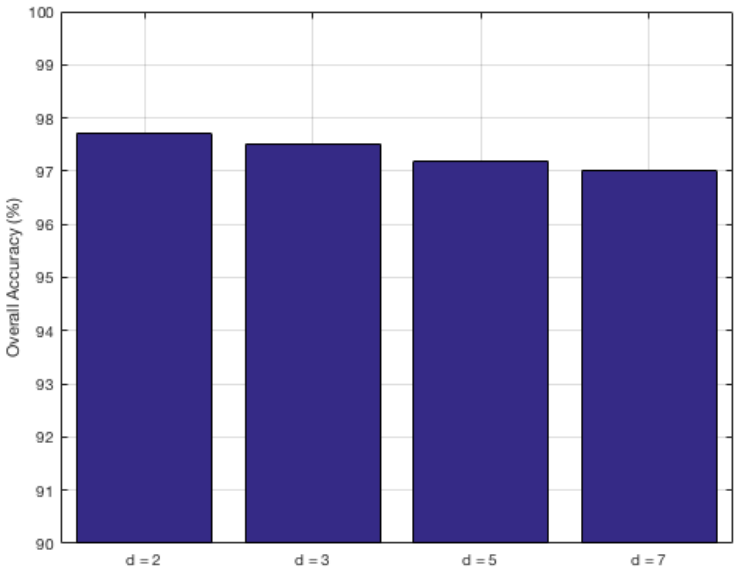


Figure 10: The classification accuracy of SVM with polynomial kernels under different degrees

Table 2: Accuracy on CNN classifiers

	LeNet Architecture(%)	Modified Architecture(%)
samples training set	92.7	93.4
60000 training samples	95.5	99.18

The overall accuracy of the linear-kernel based SVM is 96.80%, which is comparable to the best result before with the k-NN classifier.

Consider the polynomial kernel, the overall accuracy drops a little bit when the degree of polynomial goes higher, as we can seen in Figure 10. This phenomenon suggests that the overfitting is more likely to occur when the degree goes higher, and also, the accuracy is not necessarily related to the complexity of the kernel that are using. The best result of polynomial-kernel based SVM is 97.60%.

With the rbf kernel, the accuracy also change with different parameter σ, as can seen in Figure 12. With $\sigma = 4$, we reached an overall accuracy of 97.8%, which is the best among all the classification results of SVM classifiers.

5.3 CNN classifier

The overall accuracy of original CNN classifier on a sampled training set is 92.7%, which is below the overall accuracy of both k-NN classifiers with dimension reduction and SVM classifiers and after extending the training set to the full MNIST set, the accuracy increases

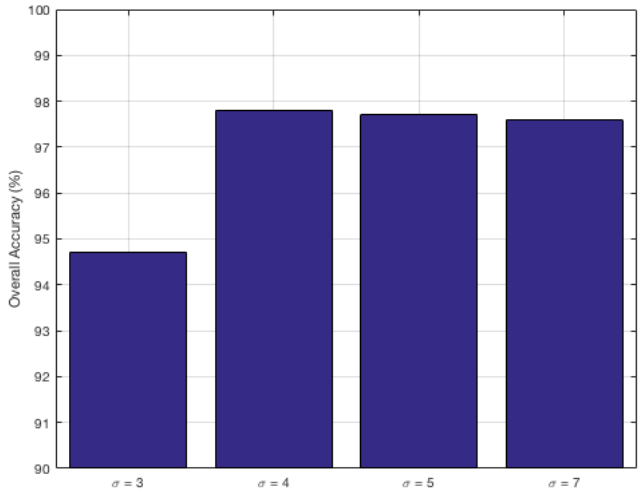


Figure 11: The classification accuracy of SVM with rbf kernels under different σ

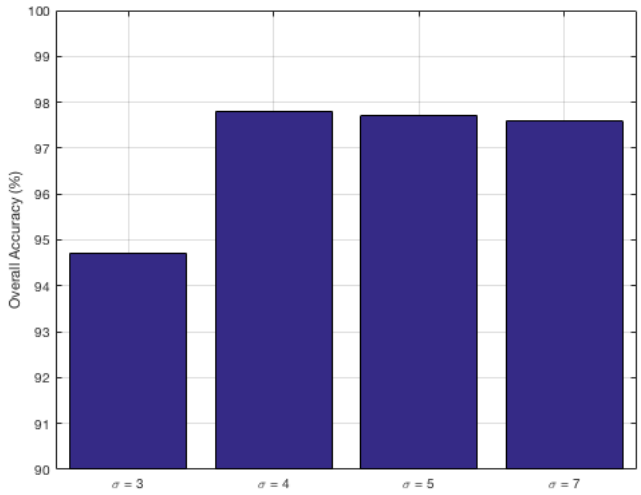


Figure 12: The classification accuracy of SVM with rbf kernels under different σ

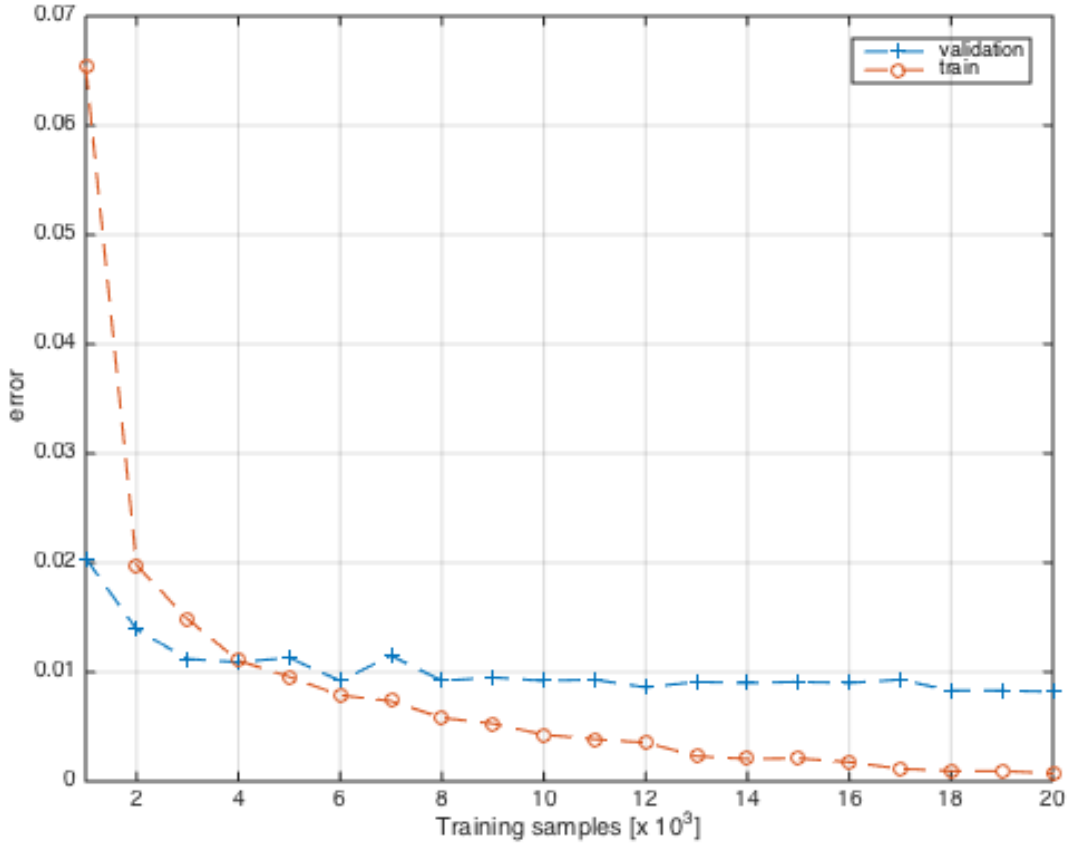


Figure 13: Error Rates of CNN classifier on Train and Validation Set with Growing Training Samples

to 95.4%, which is still below that of SVM classifier. However, this is expected because the CNN requires a huge size of training samples to determine the weights, and the modified architecture of CNN gives a better performance of 99.18%.

As shown in Figure 2, the training and validating error rates are both decreasing with the training samples, and the decreasing tendency is extremely large at the first several thousand of samples, which could partly explain the reason why the performance of subsampled CNN is worse than other types of classifiers.

6 Conclusion

- Preprocessing:
  - HOG feature: the HOG feature utilized the higher-order information of the feature vector (gradients), which added discrimination between the classes. However, the

HOG feature has a higher dimensionality than the original features, thus may increase the computational complexity of the classification step.

- SIFT feature: the SIFT descriptor is based on topology, and has a rotation invariant property, thus may not be suitable for digit recognition. Moreover, the number of descriptors found in the each image (feature vector) may not be the same, which complicated the further classification task, where the dimensionality of the feature varies.

- Classifiers:

  - Dimension reduction with k-NN: the dimension reduction methods explicitly transform the feature space. This method has a simple algorithm structure, but has a relatively high error rate. In the experimental comparison of the dimension reduction methods, Locally Linear Embedding (LLE) has the highest accuracy and relatively low computational complexity, thus is suitable for the digit recognition task.
  - Support Vector Machine (SVM): SVM has a fast training process and a relatively high accuracy. However, this classifier is sensitive to kernel choice, thus need prior knowledge of the dataset.
  - Convolutional Neural Network (CNN): the CNN renders the highest accuracy both in the literature and in our report. In the literature, the best single CNN has reached a 99.65% accuracy. In our experiments, the accuracy is 99.18%. Another strength of CNN is that is requires less preprocessing, where the convolution layers can be seen as feature extraction steps. CNN is a model-based method, thus once the coefficients are determined by the training process, the testing process will be a lot more time-efficient. However, the CNN tend to overfit the data, thus need a large training set, and the training process is time-consuming.

References

[1] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[2] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Lawrence D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition*, pages 77–77. IEEE Computer Society Press, 1994.

[3] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[4] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139. IEEE, 2011.

[5] 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.

[6] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[7] 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.

[8] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

[9] 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.

[10] 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.

[11] Yann LeCun, LD Jackel, L Bottou, A Brunot, C Cortes, JS Denker, H Drucker, I Guyon, UA Muller, E Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60, 1995.

[12] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[14] Patrice Y Simard, Dave Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE, 2003.

[15] Hasan Soltanzadeh and Mohammad Rahmati. Recognition of persian handwritten digits using image profiles of multiple orientations. *Pattern Recognition Letters*, 25(14):1569–1576, 2004.

[16] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244, 2001.

[17] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

Tiancheng's contribution

- Investigate the SIFT feature extraction techniques
- Investigate the SVM classifiers on digit recognition
- Investigate the RVM classifier on a sub-sampled dataset
- Extend the experiments before to applying on the whole dataset
- Help investigate the literature of CNN to build a new CNN architecture

Siyang's contribution

- Investigated the HOG, and discussed how it improved the performance of classification by detecting edges more accurately
- Ran the experimental comparison between the feature selected and none feature selected classification.
- Designed and trained the CNN classifier on both sampled and full MNIST set.
- Investigated and compared the performances on different architectures of CNN classifiers.