Convolutional Top-Down Mask Generator for Performance Improvement of Neural Network Classifiers

by

Yihe Tang

A senior thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science
(Computer Science)
in The University of Michigan
2015

Thesis Committee:

Professor Benjamin Kuipers, Advisor
Professor Jia Deng, Second Reader

ACKNOWLEDGEMENTS

I would like to first thank my advisor Professor Benjamin Kuipers for his patience and guidance, for his helpful advice, for encouraging me to challenge myself and think independently, and for spending hours and hours to train my communication skills. I am also grateful to Professor Jia Deng, for his useful suggestions and valuable comments in the development of this thesis.

I am appreciated with my friend Yee Zhang's help for providing me efficient functions of max pooling and padded convolution which I modified and applied in my max pooling layer's and padded convolutional layer's feed-forward part in the experiment. Without his help, I would spend a lot more time in training process. I also want to thank my friends Rui Zhang and Roy Luo for their kind help in debugging, and Roy Luo's assistance in proof reading. I am grateful to Andrew Ng, for writing UFLDL tutorial and creating very helpful programming exercises for me to practice almost all machine learning algorithms. I am also glad to meet Honglak Lee, as my instructor of EECS 445, Machine Learning, inspiring me the topic of the thesis.

I am also indebted to my roommates, Ji Wang and Roy Luo, for their kind support in the final two weeks of my thesis writing. They cooked and shopped for me, allowing me to spend more time on experiments and writing.

Lastly and most importantly, I want to thank my parents for supporting me in the past twenty two years. Without their encouragement and love, I would not be able to achieve anything significant. Thanks for giving me a wonderful life.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Convolutional Top-Down Mask Generator for Performance Improvement of Neural Network Classifiers

by

Yihe Tang

Advisor: Professor Benjamin Kuipers

This paper introduces a novel mask module which can be integrated into the existing neural network classifiers to improve the classification performance by generating a mask to filter the irrelevant background. The mask generator utilizes the top-down feedback to select useful features for the neural network classifier. It improves the classification performance of a given neural network algorithm especially when the input is corrupted by noise or background interference.

CHAPTER I

Introduction

While recent technology booming significantly improved the computational power available to the publics, one old topic is getting hot: how can computer see? Object recognition algorithms are the key to the answer. For decades, it has been an area of extensive research with the development of modern computer science.

Generally, there are two types of recognition: recognition of a special object and classification of a general category. For the first case, an instance of a special class is being identified, for example, your neighbor's car, Steve Job's face, and Apple's logo. For the second case, objects are classified into different categories, for example, numbers, letters, cars, cats, dogs, and airplanes. Both cases of recognition need to prepare some characteristic features to recognize a certain object or category.

The scientists started from various hand-crafted features, such as the Harris detector, the Hessian detector, the Laplacian-of-Gaussian (LoG) detector, the Hessian-of-Gaussian (HoG) detector and the SIFT detector [20]. These detectors utilize the deep insight of the characteristic features and proved to be effective. Many object recognizers were built based on these feature detectors, such as the SIFT face detector, the HoG person detector and the Bag-of-Words image classification [8]. Although they had competitive performance at the time they were invented, there is no such a

general way to improve the performance of the object recognition by hand-crafting a feature detector: the researcher does not have an automatic way to prove whether the detector is optimal with the current setting.

Machine learning provides a systematic way to select, generate and optimize features. The systematic generation enables researchers to obtain better high level features that are something even beyond their imagination. Neural network is one of the most popular models in machine learning with astounding flexibility on feature selection, generation and optimization. For example, sparse autoencoder is a typical neural network algorithm [**?**]. If trained on suitable datasets, it can automatically reconstruct the familiar hand-crafted corner features and the edge features in the different layer. See figure 1.1 for the edge features generated in the middle layer of the autoencoder. In the higher layer, it can even generate the eigen-object — the first few dominant orthogonal components of the whole object space (figure 1.2) which are hard to make by hands. The superb feature generation efficiency by sparse autoencoder shows the potential of utilizing neural networks for better feature selection, generation and optimization.

1.1 Motivation

When seeing an object in real world, the neural networks in human brains separate the object from the background, which is not easy for computer programs. In real world, the scene is usually complicated, since many background objects can have ambiguous colors and brightness compared to the foreground object. It takes humans a long time to learn the features of objects in their early life. These understanding is crucial for human's excellence in separating objects from the ambiguous backgrounds. Compared to the human beings, it is difficult for computers to summarize useful

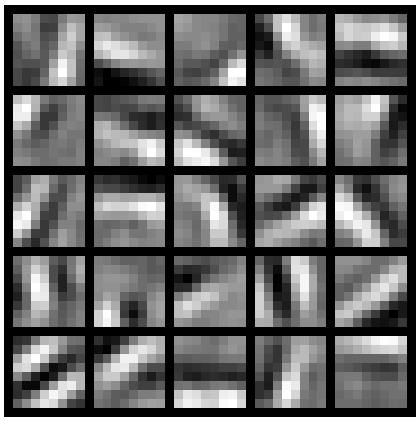

Figure 1.1: Visualization of the extracted edge features (middle features) of MNIST handwritten digits by my implementation of sparse autoencoder. The final accuracy classifed by softmax regression using the extracted eigen-stroke features (final features) is 98.18%. The experiment setting is different from the setting in chaper V and cannot compare to the results in chaper V.

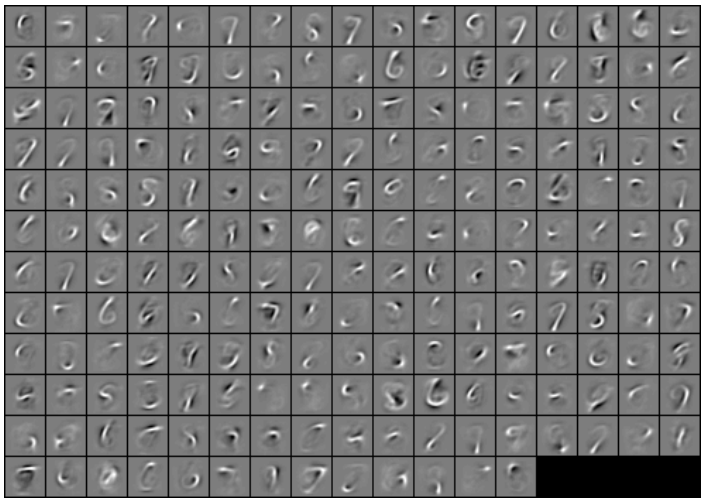

Figure 1.2: Visualization of the extracted eigen-strokes of MNIST handwritten digits by my implementation of sparse autoencoder. The final accuracy classifed by softmax regression using the extracted feature is 98.18%. The experiment setting is different from the setting in chaper V and cannot compare to the results in chaper V.

features in the chaotic backgrounds since there is no enough information for the easy separation of foreground and background. This is different from the humans' learning process: the additional information provided by three dimensional human vision helps the understanding of the relationship between foreground and background and thus to separate them. In the last decade, the researchers proposed lots of different neural network architectures. The growing recognition ratio under different settings makes it possible to put them into practical use. Meanwhile, those algorithms usually suffer a performance loss in the chaotic and noising real-world backgrounds.

The major motivation for this thesis is to improve the performance of existing neural network classifiers by applying a mask on one of their bottom layers for feature selection. By applying the mask, we can reduce noise and mitigate the interference from the unrelated background. The features improving the classification performance are expected to be retained in the selection process, while those useless or even harmful features will be eliminated. Such feature selection idea drives the design.

Another merit of adding mask is the pure differentiability of the masking operation (element-wise product), which makes it possible be trained with traditional back-propagation.

CHAPTER II

Related Work

Sequential multi-layer neural networks have achieved the state of the art in many classification tasks. For example, in the ImageNet Classification Challenges, some of them achieves the best performance among other algorithms [10, 22]. However, these algorithms usually suffer notable performance loss if the target object hides in the chaotic background [10].

Some segmentation algorithms built on deep neural networks have good performance [4, 14]. However, these segmentation algorithms are usually built on some other neural network classifiers. They are not intended to help classification but to focus on segmentation which is regarded as a harder challenge [14].

Wang, J. Zhang, Song and Z. Zhang have proposed an attentional neural network (ANN) which can select features by cognitive feedback [21]. The structure of the network is similar to the mask generator proposed in this thesis. The ANN [21] and the method proposed in this thesis both utilize top-down information and both tested the model on MNIST variations datasets. However, the ANN [21] focuses on cognitive bias. The bias itself is not generated from the model. Instead, it is a given input. It cannot solve the problem proposed in this thesis: improve the performance of a given model when the data is mixed with unrelated background. But our model,

the mask generator can generate the mask according to the given input to select the features automatically, which successfully solved our problem.

CHAPTER III

The Structure of the Mask Generator

3.1 Overview

The mask generator is a separate module existed alongside with the original neural network classifier (see figure 3.1 and 3.2). Depending on the depth of the given neural network algorithm, the mask is applied to different layers. For shallow neural network, the mask should be applied on the input layer of the network, since the upper layers are mostly occupied with high level features. Blocking any of these features is unnecessary, because almost all of the features will contribute to the classification (they are close to the output layer), which is further explained in section 4.3. Figure 3.1 illustrates the general structure of the mask module working with shallow neural networks.

For deep neural network, the mask may better apply to lower feature layers. Applying the mask to the input layer directly is usually computationally inefficient. Since the feature layers usually have lower dimensions than the input layer, applying the mask to the preliminary features generated in the bottom of the network reduces the amount of computational work and benefits from the translation invariance provided by lower layers. Figure 3.2 shows the general structure of the mask module applied to deep neural networks.

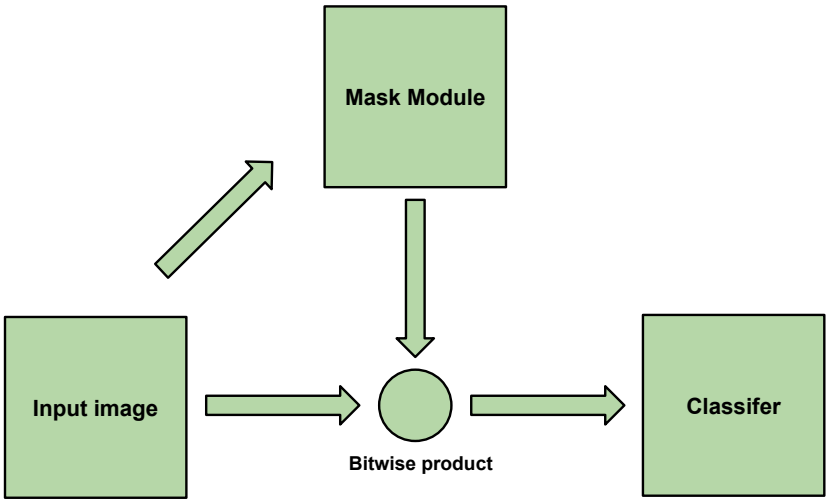

Figure 3.1: Mask module applies to shallow neural networks.

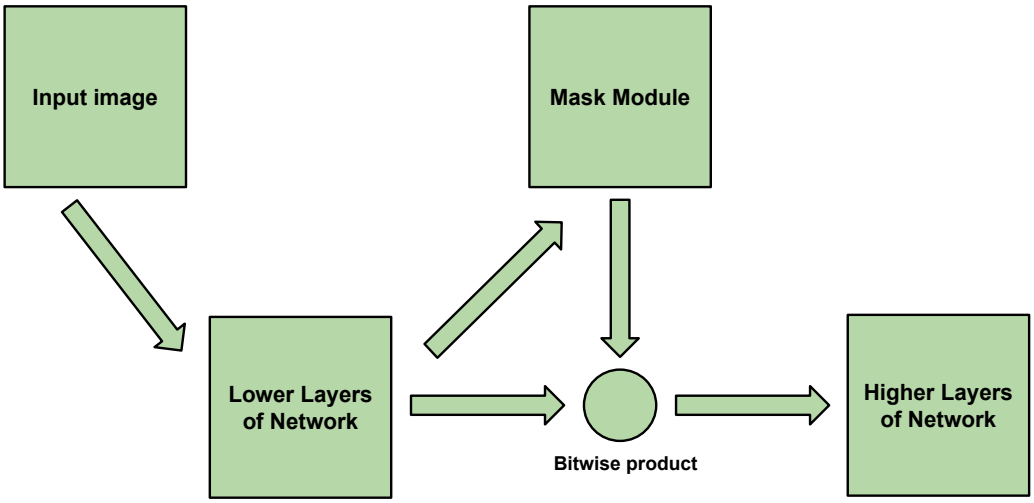

Figure 3.2: Mask module applies to deep neural networks.

The inner structure of the mask generator is shown in figure 3.3. The mask generator consists of several feedback units. For example, the mask generator shown in figure 3.3 has two feedback units. The number of the units in the mask generator is determined by the complexity of the original neural network, and specifically the layers between the masked layer and the label (output) layer. The detail setting of the mask generator such as its number of the feedback units, the number of the feature maps for each convolutional layers and the size of pooling block need to be determined case by case.

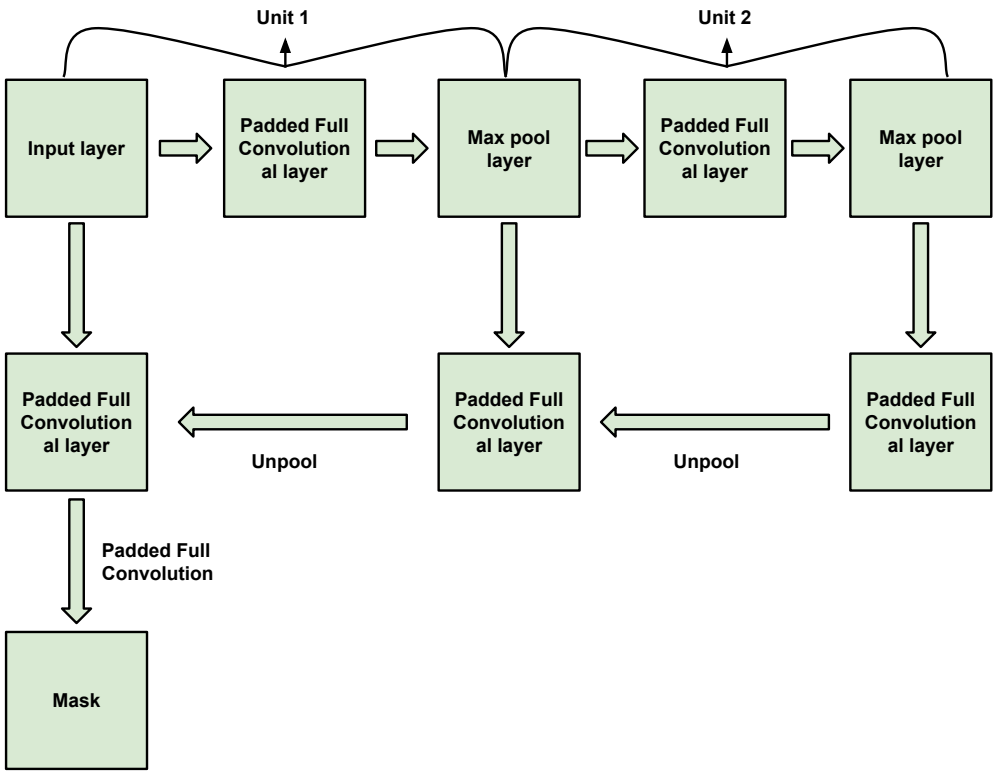

Figure 3.3: The structure of the mask generator with two units.

3.2 Convolutional Layer

The motivation of convolutional neural network comes from monkey's visual cortex.[5, 9]. The cells on the cortex are arranged in a special way, so that it is only sensitive to a small contiguous block of the visual field, called receptive field[5]. These small contiguous blocks together cover the whole visual field[5]. Since each of them is only sensitive to a small region on the visual field, they can be trained to exploit the locally spatial correlations [5].

The definition of the convolution is given as follows:

Definition III.1. Denote matrix W as the weight map, matrix I as the input layer. Then the (Padded Full) Convolution of I and W is defined as

$$C = I * W$$

where

$$C_{m,n} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I_{i,j} W_{m-i,n-j}$$

If the operation only returns the part of the convolution that can be computed without assuming I as zero-padded, it is called convolution. If zero-padded is assumed, the operation is called the padded full convolution.

Each convolutional layer has a number of shared weight maps. The output of a convolutional layer is a set of feature maps, which are the convolution of each weight map W^i and the input layer I. The shared weight maps is also called the hidden units. The neural network mainly consists of convolutional layers (and corresponding pooling layers) are called convolutional neural networks (CNN).

Compared to fully connected networks, instead of connecting to all the pixels (features) of the previous layer, CNN is locally connected. The hidden units only

connect to a small contiguous block of pixels (features) as in the definition of the convolution operation. This enables CNN to find out locally spatial correlations like the animal's visual cortex does.

By the definition of convolutional neural network, the weight map W^i is shared throughout the whole input layer. This allows the network to discover translation invariance, which means that finding out features regardless of their position in the input layer. The shared weight also reduces the total number of the free parameters, which increases the learning efficiency [5]. The structure of convolutional layer is shown in figure 3.4 [1].

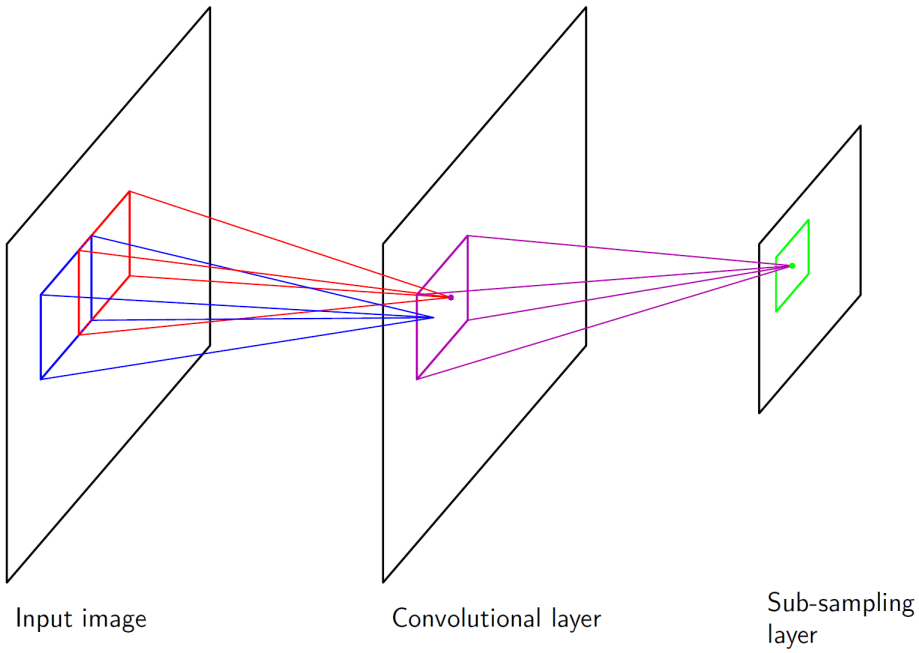

Figure 3.4: Illustration of a convolutional layer and its corresponding pooling (sub-sample) layer. (Bishop [1].)

3.3 ReLU Nonlinearity

ReLU stands for rectified linear unit, which is a neuron in the neural network using rectifier for nonlinearity. The rectifier function is defined as

$$f(x) = \max(0, x)$$

Krizhevshky, Sutskever and Hinton claims that compared to using traditional nonlinear unit sigmoid function ($f(x) = (1 + e^{-x})^{-1}$) and $\tanh(x)$, ReLU is significantly faster in the training process [10]. My own verification is given in section 4.4.

3.4 Pooling

Directly utilizing the extracted features from convolutional layer can be challenging. Suppose that we have learned n of $m \times m$ feature maps. The total number of the features is $n \times m \times m$. For example in our experiment, at C_1 layer, there are $28 \times 28 \times 18 = 14112$ features, which is relatively large. On the one hand, it can be computationally expensive to use this information directly. On the other hand, large number of features sometimes may be prone to overfitting.

We use pooling to solve this problem. Pooling can reduce the number of features and also emphasize significant features by statistical information. Statistics is good at describing large number of data with a short summary. The summary is in much lower dimensions, and describes the local information effectively. For example, we can apply mean or max function to each of the non-overlapping small blocks in the output feature maps of a convolutional layer, to find out a local statistical representation of the block. (We do not consider min function, since zero does not represent anything special). In the project, we set the non-overlapping block as 2×2 and use max function for the statistical representation.

The mechanism of pooling is shown in figure 3.5 [15]. Pooling process divides the input layer into non-overlapping fixed-size blocks. Then it selects the largest number in each block and omits other values in the block. The selected values make up the new output layer.

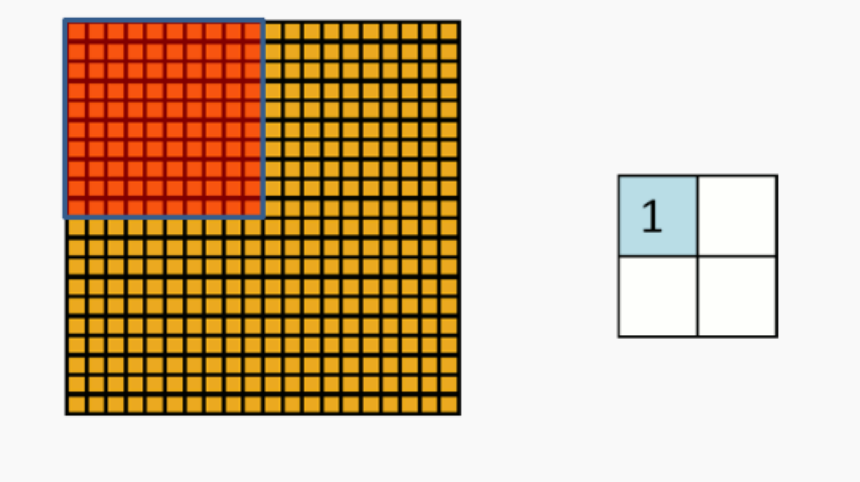

Figure 3.5: Ilustration of pooling [15]. 1 is the maximum value in the red area.

Besides, pooling also provides a form of translation invariance [15]. Assuming the block size is 2×2 and the image input is translated by one pixels, there are eight possibilities for such translation. After pooling, three out of the eight possible configurations will produce the exactly same output, which leads to better feature robustness. Choosing a larger block size enables stronger translation invariance, but the block size cannot be too large, since there will be fewer details at the same time.

3.5 Unpooling

In order to generate a mask for noise reduction and background separation, higher level information is needed since the mask is generated in the lower layer of the network. In the traditional models, the architecture is usually "sequential", in other words, it does not contain a "loop". In these models, the lower layers are unable

to get insight from higher level features. Similar to Zeiler, Matthew D and Taylor's (max) unpooling [23], we define (full) unpooling as adding the value of current pixel to all the corresponding (mapped) pixels in the "unpooled" layer. It can gather higher level abstraction into lower level mask generation. In the rest of the thesis, unpooling will mean (full) unpooling instead of (max) unpooling proposed in [23].

The unpooling operation is essential to the proposed mask generation module. It just adds the value of a given pixel in the higher layer to all the pixels that map to it from the lower layer. Unpooling can be viewed as the inverse operation of traditional pooling, which recovers original input from the pooled features by utilizing the top-down feedback in mask generation.

3.6 Training Method

The training of the network uses the same back-propagation algorithm of multi-layer neural network. Notice that the mask operation is compatible with the back-propagation, as shown below:

Let M be the mask matrix, H be the activation value before masking and A be the masked activation value. Let J be the cost function, M_{ij}, H_{ij} and A_{ij} denote the elements in M, H, A respectively. Then we have

$$A_{ij} = H_{ij} M_{ij}$$
$$\frac{\partial J}{\partial M_{ij}} = \frac{\partial J}{\partial A_{ij}} \frac{\partial A_{ij}}{\partial M_{ij}} = \frac{\partial J}{\partial A_{ij}} H_{ij}$$

which can be use in the back-propagation. In addition, we use sigmoid as the activation function for the mask matrix so that they are constrained within the interval $(0, 1)$.

However, with the addition of the musk module, the training procedure becomes crucial. Each iteration of the back-propagation can be divided into four steps, as

shown in figure 3.6.

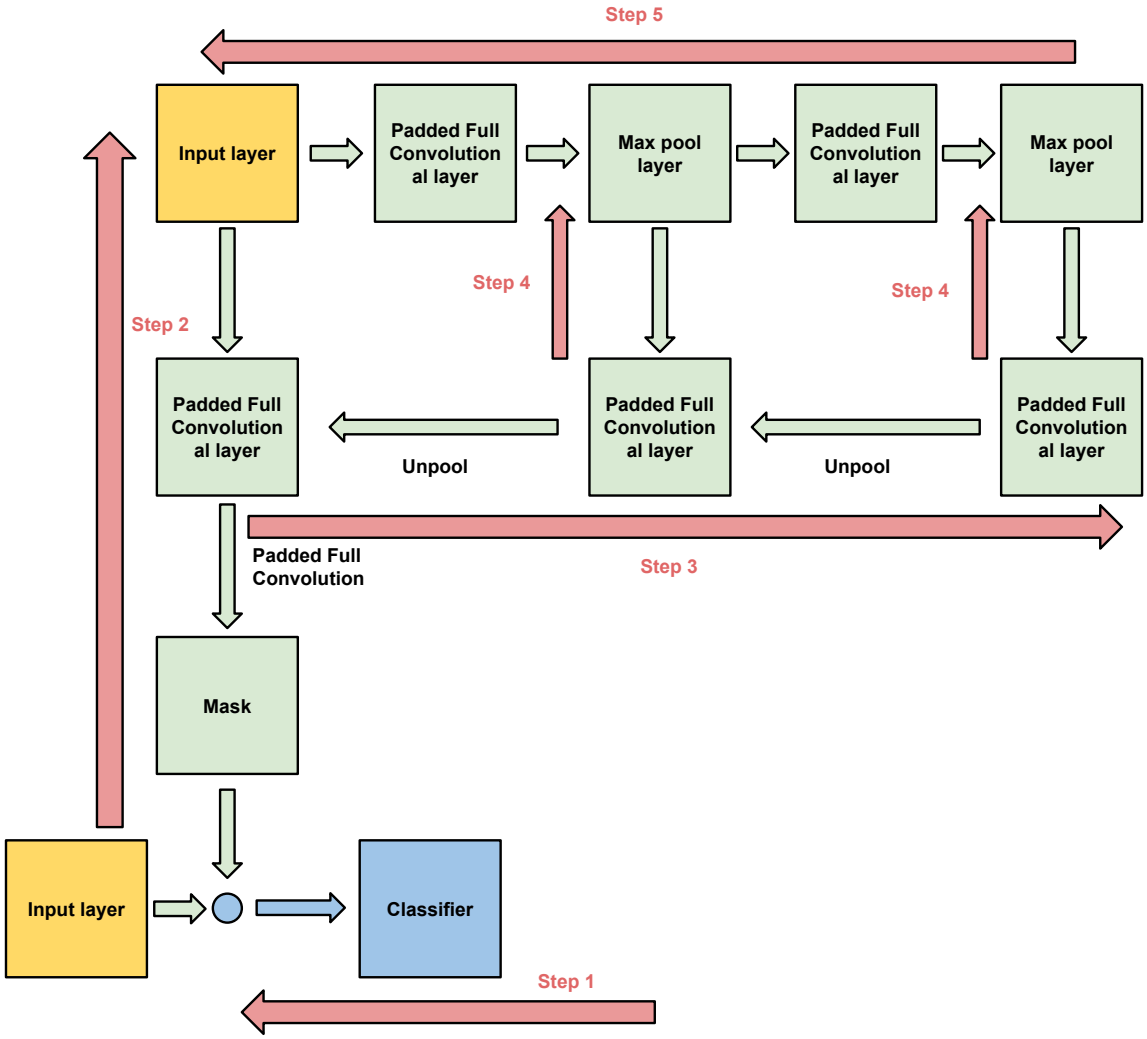

Figure 3.6: Illustration of the training steps in one iteration of training.

CHAPTER IV

Evaluation of the Mask Generator

To evaluate the mask generator, we conducted an experiments with details described in the sub-sections.

4.1 Dataset

The selected datasets for the evaluation should satisfy three criteria:

- The selected data for the experiment needs to be trained without intensive computation. Without access to strong GPU-based computational power, the selected data for performance benchmark needs to be trainable in the CAEN lab. Specifically, the CAEN lab desktop should be able to train the data within several hours.
- The data used in the experiment should be capable to illustrate how the mask module improves the performance of the original neural network in background interference and random noise.
- The data should be widely used among other researchers, which makes it possible to compare with some other modern object recognition algorithms.

Based on these criteria, I chose a published variation of the MNIST handwritten digits database [13]. MNIST stands for Mixed National Institute of Standards and

Technology, a well-known database of handwritten digits. It contains 60000 training images and 10000 testing images. Each of the images is pre-processed to fit into a 28×28 bounding box. They are also anti-aliased, grayscaled and centered by mass of the pixels.

MNIST Variations are a set of databases generated by MNIST. It contains five databases and each of them is modified as follows (Erhan [6]):

mnist-basic each of the digits were rotated by 0°, 90°, 180°, 270° uniformly.

mnist-rot each of the digits were rotated by 0° to 180 ° uniformly.

mnist-back-rand each of the digits were put on a random background. The pixels on the background was uniformly generated between 0 and 255.

mnist-back-image each of the digits were randomly put on a patch of grayscaled image.

mnist-rot-back-image combine the perturbations used in *mnist-rot* and *mnist-back-image*, namely, the images have random grayscaled background and also rotated from 0° to 180 ° uniformly.

Another advantage for this dataset is that it uses much less data in the training phase and much more data in testing compared to original MNIST database. It uses 12000 for training (2000 for validation) and 50000 for testing. For a machine learning algorithm, most of the computational time is spent on training. Smaller size of training data can save time for the researchers who do not have strong computational power. The large number of testing images also decreases the variance in the testing, making the testing benchmark more reliable.

4.2 Selected Neural Network

In order to test the efficiency of adding a mask module to an existing neural network classifier, we selected a simple but efficient neural network classifier. The structure of this convolutional neural network is shown in figure 4.1. The network consists of two sets of convolutional and pooling units, then follows by a softmax classifier. See table 4.1 for detailed settings.

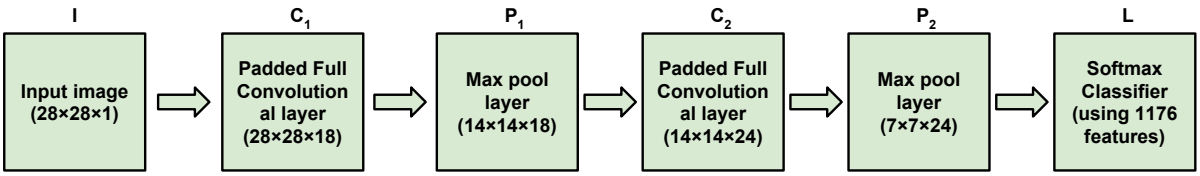

Figure 4.1: The structure of the selected neural network.

Layer	*Feature Maps Dimensions*	*Number of Feature Maps*	*Convolutional Patch Size*
I	28×28	-	-
C_1	28×28	18	9
P_1	14×14	18	-
C_2	14×14	24	5
P_2	7×7	24	-
L	1×1	1176	-

Table 4.1: Details of the selected neural network.

4.3 Details of the Mask Generator

The structure of the mask generator for the selected neural network is shown in figure 4.2. Since the selected convolutional neural network only has two convolutional and pooling units and the dimensions of the input layer is only 28×28, two feedback

units are used in the mask generator.

According to our experiment, applying the mask at the second layer of the original CNN can lead to worse performance (achieving 0.6534 in *mnist-rot-back-image*, a worse performance compared to the result in table 4.3). This follows the discussion in section 3.1.

The detailed setting of the layers is given in table 4.2.

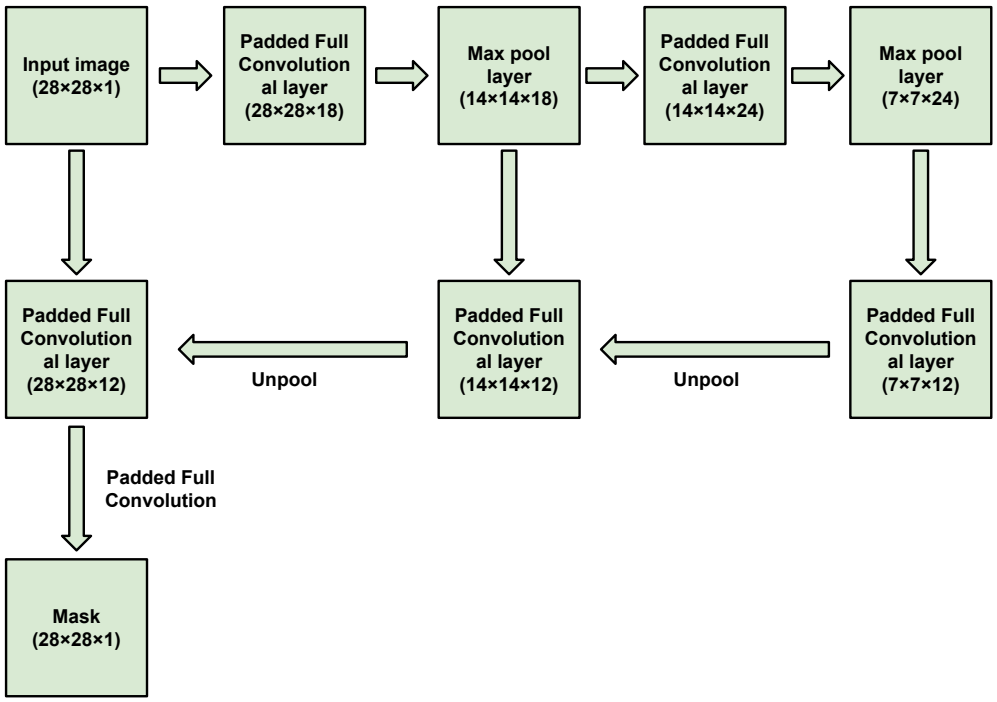

Figure 4.2: The structure of the mask generator applied to the selected neural network.

4.4 Selection between ReLU and Sigmoid

In order to verify the conclusion given in section 3.3, we need to test the practical performance of ReLU and sigmoid function. Choose the selected two level convolutional neural network introduced in section 4.2 and test the performance between

Layer	*Feature Maps Dimension*	*Feature Maps Number*	*Convolutional Patch Size*
I	28×28	/	/
C_1	28×28	18	9
P_1	14×14	18	/
C_2	14×14	24	5
P_2	7×7	24	/
C_3	28×28	12	9
C_4	14×14	12	5
C_5	28×28	12	3
M	28×28	1	1

Table 4.2: Detail setting of the mask generator.

rectifier and sigmoid function as the non-linearity in the neural units. The result is shown in figure 4.3. The result indicates that in our setting, ReLU learns faster than

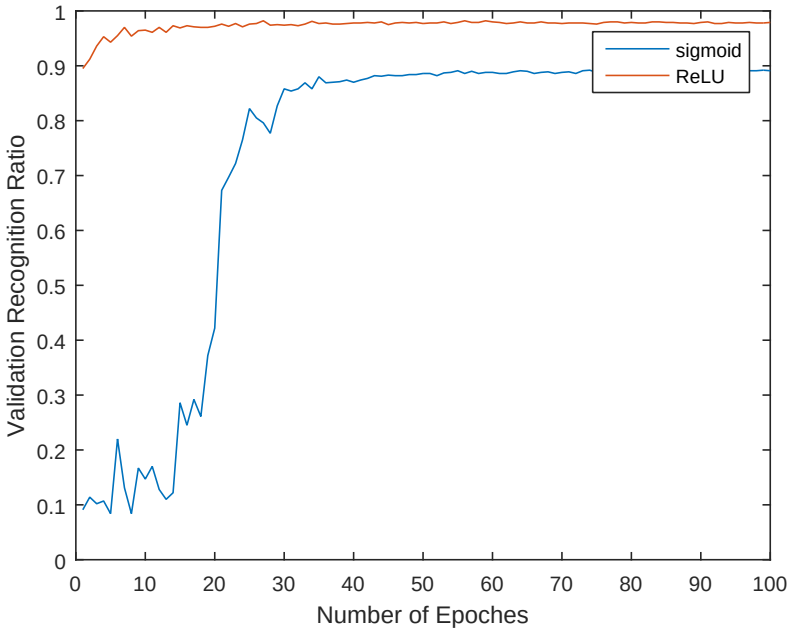

Figure 4.3: Training a two layer convolutional neural network using sigmoid function and rectifier through a hundred epochs.

traditional sigmoid neural units. For further applications of the mask generator, using ReLU can achieve better performance influence for large models trained on large datasets.

4.5 Result of the Mask Generation

Figure 4.6, 4.7, 4.8, 4.9 and 4.10 show the result of the mask generation on different corrupted MNIST datasets. Table 4.3 is the benchmark summary of the evaluation.

Datasets	*CNN without Mask*	*CNN with Mask*	*Recognition Ratio Improvement*
mnist-basic	97.51	97.98	0.47
mnist-rot	87.34	89.86	2.52
mnist-back-rand	93.22	94.07	0.85
mnist-back-image	89.51	92.52	3.01
mnist-rot-back-image	61.24	70.92	9.68

Table 4.3: Performance comparison between the existence of the mask module.

4.5.1 mnist_basic and mnist_rotation

For datasets *mnist_basic* and *mnist_rotation*, the generated masks are the identical mask (all one), which seems to be overfitted. However, this is within the expectation. For the datasets *mnist_basic* and *mnist_rotation*, no random noise or unrelated background is added, because MNIST database already pre-processed images to normalize them. The images in the database have already been centred, anti-aliased, grayscaled and unified. The only difference between MNIST variations *mnist_basic*, *mnist_rotation* and the original MNIST database is the orientation of the digits.

Therefore, the mask in this case should not block anything. We see the results on *mnist_basic*, and *mnist_rotation* do follow the expectation. The only non-one regions lie in the corners. This can be explained. The L_2 regularization in the training process punishes large values in the mask. The mask can only be all ones if staying all ones is considerably beneficial, which is impossible in this case. Meanwhile, blocking the corners will not influence the classification since the digits are centred. This explains the almost-all-one mask with few non-ones in the corners derived from the mask generator.

In table 4.3, as expected, the performance difference between the existence of the

mask is very small for *mnist-basic*. However, the performance difference is pretty large for *mnist-rot*. The joint training process complicates the whole network. The increasing complexity may provide the network an opportunity to avoid staying in the current local maximum and to move forward. There is a notable phenomenon that at the beginning stage of the training, the mask looks very similar to the mask in *mnist-back-rand*, *mnist-back-image* and *mnist-rot-back-images*. The outline of the digits can be seen on the mask as seen in figure 4.4. Compare the validation recognition ratio in the training process (see figure 4.5), we see the masked version is trained better at the beginning. This supports our assumption that although the mask in these two cases does not function at the final stage, they are still useful in the early stages.

4.5.2 mnist-back-rand

For *mnist-back-rand* dataset, the generated mask reduced most of the random noise. Shown in figure 4.8, compared to the original image, the masked image has far less random noise than before. However, in table 4.3, we notice the small performance difference between the masked CNN and unmasked CNN. Why an effective denoising process does not improve the ratio of recognition? We think the locally spatial correlation of CNN homogenizes the random noise in the training process by twice convolutions and poolings. Although the recognition ratio does not increase significantly, the mask does good job for denoising the input as shown in figure 4.8.

4.5.3 mnist-back-image and mnist-rot-back-image

For *mnist-back-image* and *mnist-rot-back-image*, the mask generator does a good job on separating the unrelated background as seen in the figure 4.9 and figure 4.10. The successful separation of the foreground digits and the background random images enhances the recognition ratio, especially for *mnist-rot-back-image* as seen in

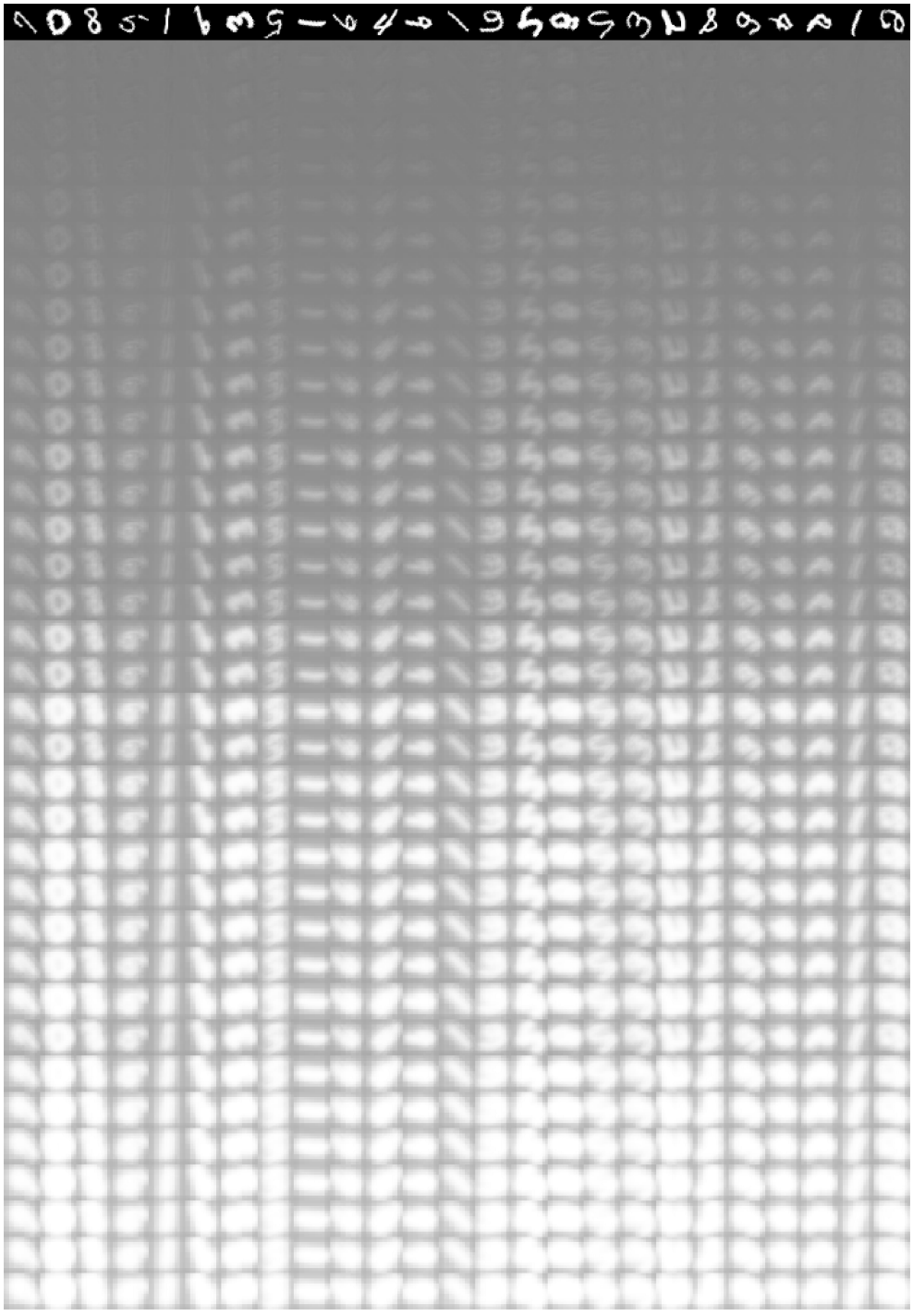

Figure 4.4: Visualization of the generated mask on *mnist-rot* dataset throught the training process. We can see the generated mask starts from almost 0 at everywhere. Then we can figure out a vague outline of the digits. Finally it turns out to be all pass (all one) mask.

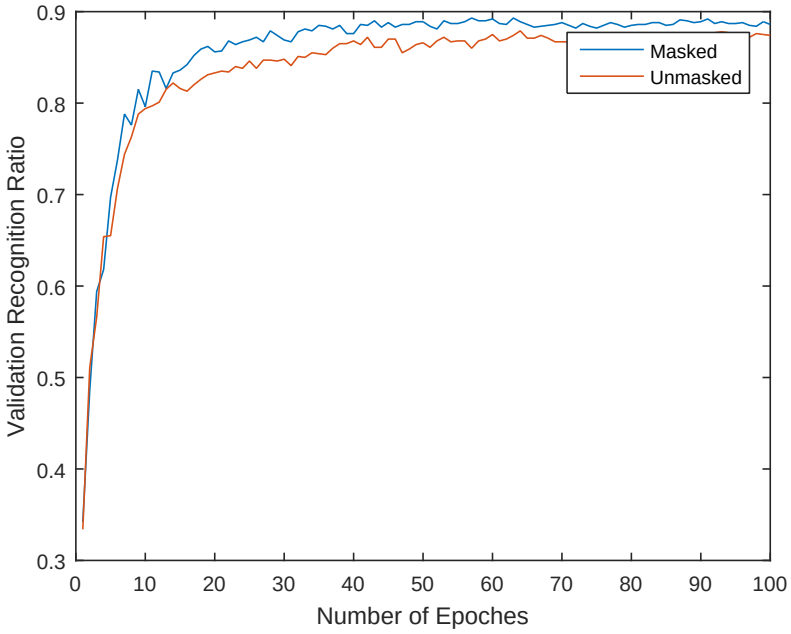

Figure 4.5: Comparision between the validation recognition ratio in the training process of masked and unmasked CNN-2 on *mnist-rot* dataset.

Figure 4.6: Visualization of the generated mask on *mnist-basic* dataset. Note that the image on the left, in the middle and on the right are the orignal image, the generated mask by our mask generator, and the justified image by appling the mask to the original image respectively.

Figure 4.7: Visualization of the generated mask on *mnist-rot* dataset. Note that the image on the left, in the middle and on the right are the orignal image, the generated mask by our mask generator, and the justified image by appling the mask to the original image respectively

table 4.3.

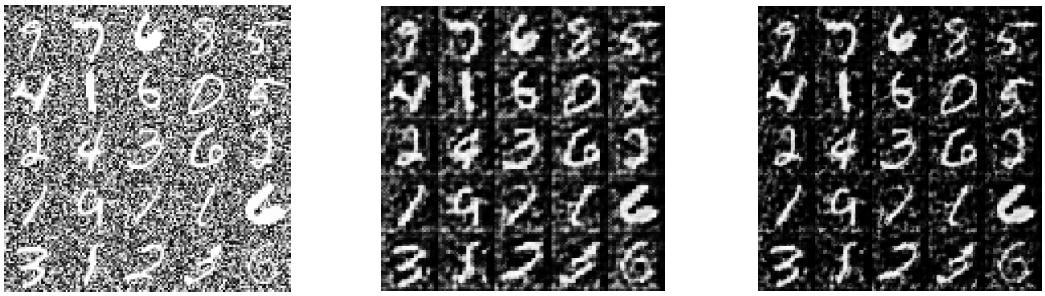

Figure 4.8: Visualization of the generated mask on *mnist-back-rand* dataset. Note that the image on the left, in the middle and on the right are the orignal image, the generated mask by our mask generator, and the justified image by appling the mask to the original image respectively

4.5.4 Comparison to the State-of-the-Art Classification Result of MNIST Variations

According to the survey paper [12] and all the reachable papers since 2014 related to MNIST variations classification [19, 11, 25, 2], we compare our result to other models. We exclude the models with different training and testing settings, such

Figure 4.9: Visualization of the generated mask on *mnist-back-image* dataset. Note that the image on the left, in the middle and on the right are the orignal image, the generated mask by our mask generator, and the justified image by appling the mask to the original image respectively

Figure 4.10: Visualization of the generated mask on *mnist-rot-back-image* dataset. Note that the image on the left, in the middle and on the right are the orignal image, the generated mask by our mask generator, and the justified image by appling the mask to the original image respectively

as [24]. As shown in figure 4.4, our simple model with the mask generator outperforms the-state-of-the-art result in *mnist-back-rand* and *mnist-back-image*, and also wins the second place in *mnist-rot-back-image*, showing the promise to improve the recognition ratio of a given network under noisy and chaotic backgrounds.

Model	*basic*	*rot*	*back-rand*	*back-image*	*rot-back-image*
Masked CNN-2	2.02	10.14	**5.93**	**7.48**	29.08
SVM_{rbf}[12]	3.03	10.38	14.58	22.61	32.62
SVM_{poly}[12]	3.69	13.61	16.62	24.01	37.59
NNet[12]	4.69	17.62	20.04	27.41	42.17
DBN-1[12]	3.94	12.11	9.80	16.15	31.84
SAA-3[12]	3.46	11.43	11.28	23.00	**24.09**
DBN-3[12]	3.11	12.30	6.73	16.31	28.51
CR-SAE [11]	-	12.86	10.98	18.59	47.68
SAE[11]	-	14.23	13.60	19.97	53.56
CAE-1[11]	-	11.59	13.57	16.70	48.10
CAE-2[11]	-	9.66	10.90	15.50	45.23
SdA-3[11]	-	10.29	10.38	16.68	44.49
RBM [19]	-	-	11.39	15.42	49.89
imRBM [19]	-	-	10.46	16.35	51.03
discRBM [19]	-	-	10.29	15.56	48.34
RBM-FS [19]	-	-	11.42	15.20	49.65
PGBM [19]	-	-	7.27	13.33	45.45
Supervised PGBM [19]	-	-	6.87	12.85	44.67
PGBM+DN-1 [19]	-	-	6.08	12.25	36.76
DAE-2J [25]	2.44	8.40	8.98	17.11	46.94
CAE-3J [25]	2.78	7.91	13.67	17.24	47.19
TIRBM [2]	-	**4.20**	-	-	35.50
ScatNet-2 [2]	1.27	7.48	12.30	18.40	50.48
RandNet-1 [2]	1.86	14.25	18.81	15.97	51.82
RandNet-2 [2]	1.25	8.47	13.47	11.65	43.69
PCANet-1 [2]	1.44	10.55	6.77	11.11	42.03
PCANet-2 [2]	1.06	7.37	6.19	10.95	35.48
LDANet-1[2]	1.61	11.40	7.16	13.03	43.86
LDANet-2 [2]	**1.05**	7.52	6.81	12.42	38.54

Table 4.4: Performance comparison between the masked CNN and the-state-of-the-art result.

CHAPTER V

Future Work

5.1 Further Refinement of the Structure

Due to the limitation of the computing resource, we chose a very simple structure to evaluate our mask generator. Adding more layers in each feedback units can significantly increase the training time which stops us from making improvements and debugging efficiently. Therefore we sticked to this simple structure. However, it is promising to increase the effectiveness of the feature selection by improving the structure of the mask, specially adding some full connected layers after max-pooling layers. This may be able to extract higher level features in the mask generating module, so that the unpooling process can do better feedback on the mask layer.

The way for feedback from higher level of the network to the lower level can also be refined. The unpooling operation simply adds the higher level abstractions to corresponding lower layer regions. Although this is effective enough as shown in our evaluation and the locally spatial correlation is exploited in the process, however, the higher abstraction is also restrained by the locally spatial correlation. What if the higher abstraction is not local? The current feedback system cannot handle this well.

5.2 Weakly Supervised Learning on Large Datasets

Convolutional neural networks construct successive feature vectors that progressively describe the properties of larger and larger image areas [7, 10, 16]. Recent works [17, 18, 3] train convolutional feature extractors on a large supervised image classification task, such as ImageNet, and transfer the trained feature extractors to other object recognition tasks, such as the Pascal VOC tasks.

Oquab, Bottou, Laptev and Sivic's weakly supervised CNN architecture [16] has successfully reached the state-of-art result on Pascal VOC 2012 without using bounding boxes. However, although the error ratio of recognition is comparatively low, the boundary of the detected objects are not clear enough. Combining this successful deep CNN structure with the novel feature selection ideas proposed in this thesis, we may modify [16] and design a mask generating module to generate feature masks for immediate features. This further feature selection may block some harmful features and provide better segmentation for the detected objects.

5.3 Foreground and Background Separation

The results in section 4.5.2 and 4.5.3 show effective mask generators for noisy digits and digits with unrelated background images. The trained mask generators can not only be applied to their corresponding neural network classifiers where they are trained, but they can also be used independently after the training. Given an input image, the mask generator can produce a mask to filter out the background. In section 4.5.2 and 4.5.3, we see the separation is impressive. The independently used mask can be used in many fields of computer vision, such as segmentation. Further improvement on the structure of the mask module and also training method of the current model may build a more reliable foreground background separator for

general propose with better capability of separation under complex settings.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

[2] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *arXiv preprint arXiv:1404.3606*, 2014.

[3] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[4] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. *arXiv preprint arXiv:1412.1283*, 2014.

[5] Deeplearning.net. Convolutional neural networks (lenet). *DeepLearning 0.1 documentation [Online]. Available: http://deeplearning.net/tutorial/lenet.html*, 2015.

[6] Dumitru Erhan. Mnist variations handwritten digit database. *Google Lab [Online]. Available: http://www.iro.umontreal.ca/ lisa/twiki/bin/view.cgi/Public/MnistVariations*, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[8] Kristen Grauman and Bastian Leibe. *Visual object recognition*. Morgan & Claypool Publishers, 2010.

[9] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[10] 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.

[11] Martin Längkvist and Amy Loutfi. Learning feature representations with a cost-relevant sparse autoencoder. *International journal of neural systems*, 25(01), 2015.

[12] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[13] Yann LeCun and Corinna Cortes. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2015.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014.

[15] Andrew Ng. Ufldl. *Deep Learning Tutorial [Online]. Available: http://deeplearning.net/tutorial/lenet.html*, 2015.

[16] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Weakly supervised object recognition with convolutional neural networks. 2014.

[17] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[18] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[19] Kihyuk Sohn, Guanyu Zhou, Chansoo Lee, and Honglak Lee. Learning and selecting features jointly with point-wise gated {B} oltzmann machines. In *Proceedings of The 30th International Conference on Machine Learning*, pages 217–225, 2013.

[20] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[21] Qian Wang, Jiaxing Zhang, Sen Song, and Zheng Zhang. Attentional neural network: Feature selection using cognitive feedback. In *Advances in Neural Information Processing Systems*, pages 2033–2041, 2014.

[22] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[23] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.

[24] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1461, 2012.

[25] Yingbo Zhou. Learning deep autoencoders without layer-wise training. *arXiv preprint arXiv:1405.1380*, 2014.