Improving Quality of Training Samples Through Exhaustless Generation and Effective Selection for Deep Convolutional Neural Networks

Takayoshi Yamashita[1], Taro Watasue[2], Yuji Yamauchi[1] and Hironobu Fujiyoshi[1]
[1]*Chubu University, 1200, Matsumoto-cho, Kasugai, Aichi, Japan*
[2]*Tome R&D, kyoto, Japan*
yamashita@cs.chubu.ac.jp

Keywords: Deep Learning, CNN, Data Augmentation, Data Selection, MINIST.

Abstract: Deep convolutional neural networks require a huge amount of data samples to train efficient networks. Although many benchmarks manage to create abundant samples to be used for training, they lack efficiency when trying to train convolutional neural networks up to their full potential. The data augmentation is one of the solutions to this problem, but it does not consider the quality of samples, i.e. whether the augmented samples are actually suitable for training or not. In this paper, we propose a method that will allow us to select effective samples from an augmented sample set. The achievements of our method were 1) to be able to generate a large amount of augmented samples from images with labeled data and multiple background images; and 2) to be able to select effective samples from the additionally augmented ones through iterations of parameter updating during the training process. We utilized exhaustless sample generation and effective sample selection in order to perform recognition and segmentation tasks. It obtained the best performance in both tasks when compared to other methods using, or not, sample generation and/or selection.

1 INTRODUCTION

Deep learning methods have been applied to many challenging tasks including speech recognition, bioinformatics and object recognition. Among those, Deep Convolutional Neural Networks (ConvNets) are used in various benchmark tests for handwritten characters(Y. LeCun ad L. Bottou and P.Haffner, 1998), house numbers(Sermanet et al., 2012), traffic signs(Ciresan et al., 2012) and ImageNet(Sutskever and Hinton, 2012). ConvNets lead to significantly advanced performance results with these dataset. Krizhevsky et al. obtained impressive classification performances using a large ConvNets and won the ImageNet 2012(Sutskever and Hinton, 2012). Zeiler et al. also utilized a ConvNets and applied it to not only the recognition task but also the localization challenge, winning the last ImageNet 2013(Zeiler and Fergus, 2013). Because of these achievements, deep learning including ConvNets is becoming a more and more popular learning method.

The advantage of ConvNets is that it is able to extract complex and suitable features for the task. It can reduce the burden of designing the features since the entire system is trained from raw pixels.

Although many researchers have proposed methods to address the disadvantage of ConvNets, but it remains that they require a huge amount of labeled samples. Although many benchmark methods can prepare an abundant number of samples for training, this is often not just enough to fulfill all the learning potential that ConvNets could actually achieve. Creating additional samples through data augmentation method is one of the solutions to this problem as it enables us to control at will the amount of samples to be used. However, simple data augmentation does not take account of the quality of the augmented samples, i.e. whether the additional samples will actually be useful or not for training. While the amount of samples is important for an effective training method, so is their quality.

In this paper, we propose a method to select useful samples from the ones generated through data augmentation. First, we utilized an asynchronous sample generation process. This data augmentation method did not only use translation, scaling and rotation, but also deformed the samples through elastic distortion. Then, we selected the useful samples from all the augmented ones through iterations of the parameter update process. We applied this method to a ConvNets and evaluated its performance for several tasks, including recognition and segmentation. For the recognition task, we tested the efficiency of our sample selection method in MNIST for hand posture recognition with complex figures in shape variation. Furthermore, we applied our proposed method to a segmentation task in order to show its wide application range.

The rest of this paper is organized as follows: The related works and Convolutional Neural Networks are briefly reviewed in Section 2 and 3. Our proposed method is described in Section 4 and our experiment results are given in Section 5. Finally, we present our conclusions in Section 6.

2 RELATED WORKS

Since the early 1990's, many researchers have proposed methods that use ConvNets(Vaillant et al., 1994)(Y. LeCun ad L. Bottou and P.Haffner, 1998). More recently, ConvNets have demonstrated state of the art performance for text detection(Delakisand and Garcia, 2008), image recognition(Ciresan et al., 2012), and pedestrian detection(Sermanet et al., 2013)(Ouyang and Wang, 2013b)(Ouyang and Wang, 2013a). After being utilized by Krizhevsky et al. to win the object recognition competition ImageNet, ConvNets moved into the limelight and their use spread in numerous computer vision research fields(Sutskever and Hinton, 2012). ConvNets achieved a breakthrough on the 1000-class ImageNet task, and while they are mainly used for object recognition, they have a tremendous potential of applications, like pedestrian detection as proposed by Sermanet(Sermanet et al., 2013). ConvNets combine multi-stage features, skipped layers connections and unsupervised pre-training with sparse coding. They integrate both global shape information and local distinctive motif information. The unsupervised pre-training initializes the filters at each layer and obtains efficient filters in a fine tuning process. Ouyang et al. proposed a pedestrian detection method using a special ConvNets that obtained the filter output from local regions(Ouyang and Wang, 2013b). Each local region corresponded to a body part, such as the head, upper body, left or right half body, and all responses were joined together in the final layer. Multi-task applications are also a field of usage where ConvNets are able to achieve great results. Osadchy et al. utilized ConvNets for simultaneous face detection and pose estimation by directly predicting the parameters of face location and head pose(Osadchy et al., 2007). Face location and head pose were represented by high-dimensional points on a 3D manifold, and the feature space was trained by ConvNets to output the face location and head pose. ConvNets based segmentation can also be used to perform object localization. For example, training was done to classify the central pixel of the window with an objectfs category label to represent the said object with a bounding contour instead of a traditional bounding box(Jain et al., 2007).

ConvNets, however, still have several disadvantages that restrain their full potential. Although ConvNets represent high-dimentional features due to having a lot of filter banks, the large number of parameters also may cause overfitting problems. Many researchers proposed various approaches to avoid overfitting. Here are the two main ones: Applying a micro architecture to the ConvNets. (Scherer et al., 2010)(Boureau et al., 2010)(Goodfellow et al., 2013); and Using data augmentation. Data augmentation is known as the easiest and most common method to reduce overfitting on training data. Augmented data is obtained by artificially enlarging the data set but preserving their labels. Put simply, the amount of data is augmented by cropping different regions from the same image and mirroring them(Sutskever and Hinton, 2012)(Hinton et al., 2012). Additional transformations such as scaling or rotation can also be applied (Wan et al., 2013). Simard proposed a highly advanced deformation method called elastic distortion, effective for handwritten character recognition(Simard et al., 2003). We found that elastic distortion is also useful in hand posture recognition (see Section 5). However, training a huge amount of parameters using augmented data takes a lot of time. In order to reduce learning time, vast parallel computing with GPU is often used. Fortunately, some GPU computing software packages that can train ConvNets are nowadays available(Sermanet et al., 2013). Commonly, GPU parallelization is used when updating parameters. Even though GPU helps updating the parameters faster, it requires complex data augmentation that can be a bottleneck to the training process. Krizhevsky et al. implemented simple data augmentation in Python on CPU while parameter updating is computed on GPU(Sutskever and Hinton, 2012). Nevertheless, when a highly effective and highly sophisticated data augmentation process is required, if both the data generation and the parameters refinement are run on the same process, the whole takes quite some time to complete the training. However, if data augmentation and parameter updating are separate process and those two can asynchronously access the data, then the program is kept simple and training time can be reduced. Yet, approaches like these only focus on the mass-generation of training data and do not provide solutions to the issues of the effectiveness of the samples to train or the selection of such high performing samples. That is, there must be some effective sample selection method used to achieve high performance while using relatively less samples. We also focus on these issues.

3 CONVOLUTIONAL NEURAL NETWORK

ConvNets (Y. LeCun ad L. Bottou and P.Haffner, 1998) contain an alternate succession of convolutional layers and subsampling layers, based on the notion of local receptive fields discovered by Hubel (Hubel and Wiesel, 1962). There are several types of layers, including input layers, convolutional layers, pooling layers and classification layers. Each input layer also has, besides the raw data, edge and normalized data as input. The convolutional layer has M kernels with size $(Kx \times Ky)$ and filters them in order to input data. The filtered responses from all the input data are then subsampled in the pooling layer. Scherer (Scherer et al., 2010) found that max pooling can lead to faster convergence and improved generalization, while Boureau (Boureau et al., 2010) analyzed theoretical feature pooling. Max pooling can output the maximum value in certain regions such as a 2×2 pixel. The convolutional layer and pooling layer are laid alternatively in order to create the deep network architecture. Finally, the output feature vectors from the last pooling layer are used in the classification layer. The classification layer will output the probability of each class through a soft max connection of all the nodes with weights in the previous layer. Unlike Belief neural networks, ConvNets assume supervised learning where filters are randomly initialized and updated through backpropagation (Rumelhart et al., 1986) (Duffer and Garcia, 2007).

3.1 Training of Convolutional Neural Networks

ConvNets are trained by backpropagation just as general neural networks. Backpropagation uses the function shown in Eq.(1) to estimate the connected weights with minimized E by gradient descent in Eq.(2).

$$E = \frac{1}{2}\sum_{p=1}^{P} E_p \tag{1}$$

$$w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} + \Delta w_{ji}^{(l)} = w_{ji}^{(l)} - \lambda \frac{\partial E_p}{\partial w_{ji}^{(l)}} \tag{2}$$

Note that $\{p|1,...,P\}$ is the training sample, $\boldsymbol{o}_p$ is the corresponding value of training sample p in output layer and $\boldsymbol{t}_p$ is the label data of p. λ is the training ratio, $w_{ji}^{(l)}$ is the weight that connects from the node i in layer l to the node j in the next layer. The error for each training sample E_p is the sum of the differences between the output value and the label. $\Delta w_{ji}^{(l)}$ is represented as following Eq.(3).

$$\Delta w_{ji}^{(l)} = -\lambda \delta_k^{(l)} y_j^{(l-1)} \tag{3}$$

$$\delta_k^{(l)} = e_k \phi(V_k^{(l)}) \tag{4}$$

$$V_k^{(l)} = \sum_j w_{kj}^{(l)} * y_j^{(l-1)} \tag{5}$$

$y_j^{(l-1)}$ is the output of the node j in the $(l-1)$th layer and e_k is the error of node kD $V_k^{(l-1)}$ is the accumulated value connected to node k from all nodes in the $(l-1)$th layer. The local gradient descent is obtained by Eq.(4). The activation function ϕ has variation such as sigmoid, hyperbolic tangent and ReLu(Sutskever and Hinton, 2012). The connected weights in the entire network are updated concurrently for a predetermined number of times, or until reaching the satisfying convergence condition.

Backpropagation has multiple ways to calculate the error E, including *full-batch*, *online* and *Mini-batch*. *Full-batch* gives all training data at once. As such, it requires few iterations but is weak for convergence due to the increasing gradient descent. *Online* gives the training data iteratively. As such, it obtains optimized results from its small gradient descent, but requires lengthy processing time due to its numerous iterations. *Mini-batch* is a commonly used middle approach that updates the weight with small subsets of training data. It is able to effectively update connected weights with a huge amount of training data in a realistic time length.

4 PROPOSED METHOD

The conventional ConvNets method trains from a huge yet limited amount of training samples. In most cases, the training process divides the training samples into subsets of mini-batch size and uses them to update the parameters. When the training process finishes using all the subsets, it starts receiving data again from first subset through a process called *epoch*. While this enables a continuous updating of the parameters, it has distinctive limitations in the variation of its samples. We can assume that variation in the data to be continuously given to the network is significantly important when trying to obtain the best parameters. To demonstrate this assumption

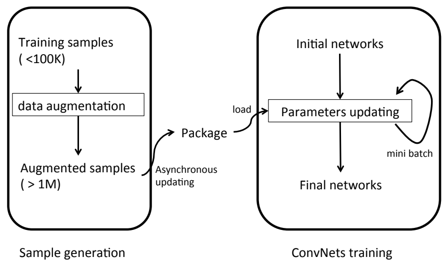


Figure 1: Entire training system. It consists of two parts, sample generation and ConvNets training. The sample generation make the augmentation samples through data augmentation from basis training samples. The package that contain certain samples is utilized parameter updating of networks. The package are asynchronously updated and reloaded to training process.

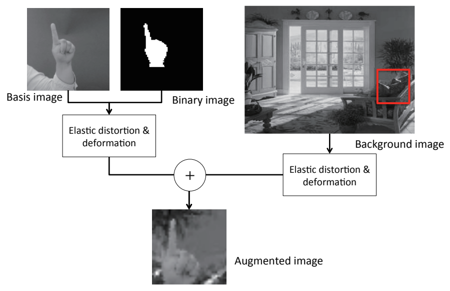

Basis image
Binary image
Elastic distortion & deformation
Background image
Elastic distortion & deformation
+
Augmented image

Figure 2: Sample generation process. The augmented image is made by elastic distortion and deformation of scaling, rotation and translation from basis image and binary image. It also synthesizes object image and background image.

and achieve better results than the ones that can be obtained by concrete training samples, we propose here a method for an asynchronous exhaustless sample generation coupled with an effective sample selection during parameter updating. We illustrated the entire training system in Fig.1. The system consists of two processes, data augmentation and ConvNets training. The data augmentation continuously creates a package that includes a certain amount of training samples, while the ConvNets training updates the parameters using selected effective samples from the package. The package is loaded to ConvNets and used for training, and when all training samples in the package are used, the package is reloaded. We will describe details of the sample generation and selection in the following sections.

4.1 Exhaustless Sample Generation

ConvNets training requires samples with broad variation including non-controlled backgrounds if it is to be used for real-world applications. Each sample we used is generated through data augmentation. We start with binary labeled data and multiple background images to create a vast combination of samples through elastic distortion, as shown in Fig.2. The elastic distortion arranges the corresponding new target position x^* from the original position of every pixel. The new target position $\boldsymbol{x}^*$ is derived from random displacement and is applied a scaling factor α as follows,

Table 1: Deformation range of each factor.

deformation factor	range
translation	± 3
scaling	$\pm$ 5%
rotation	$\pm$ 5
brightness	$\pm$ 10%

$$\boldsymbol{x}^* = \boldsymbol{x} + \alpha \Delta \boldsymbol{x}. \tag{6}$$

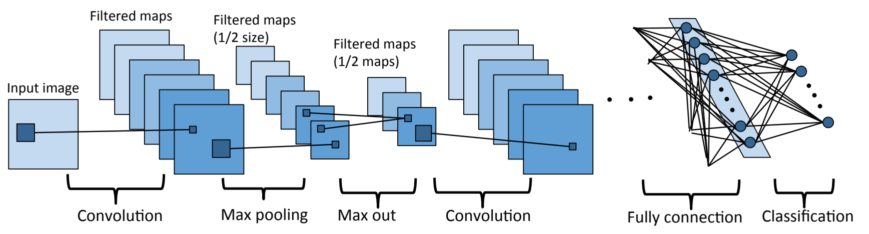
$\Delta \boldsymbol{x}$ is a random real value between -1 and 1 with uniform distribution. Like the method used by Simard, $\Delta \boldsymbol{x}$ is convolved with Gaussian of standard deviation σ to obtain smoothed displacements(Simard et al., 2003). The value at position $\boldsymbol{x}^*$ is computed through the neighboring pixels using bilinear interpolation.

After applying elastic distortion on both the sample images and the binary labeled data, the synthetic images are then synthesized with background images while applying scaling, rotation and translation. We define the deformation range in Table.1. The background regions are randomly cropped from the original images.

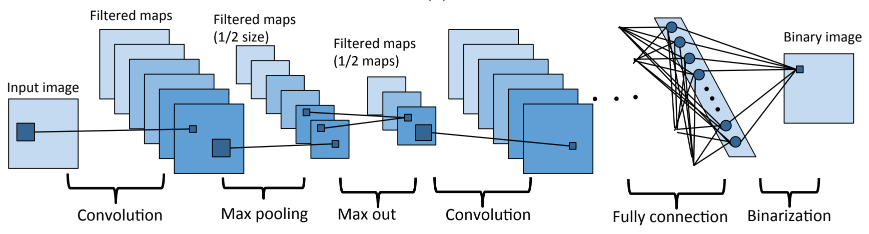
Sample generation results in a package that includes a set amount of synthetic images. That package is then updated asynchronously to avoid the recycling of the same samples during the training process.

4.2 Effective Sample Selection

We employ a sample selection of data to obtain effective training samples from the package, as the data augmentation process itself does not take in factor the quality of the additional samples generated. Variation within training samples data is quite important when training sophisticated networks. Using only elementary samples repeatedly for training does not tend to provide great results. The current training networks require an increasing difficulty in their samples in order achieve their expected performances in real environments and avoid over fitting. For our effective sample selection, we evaluate all the samples in the package using current networks and sort them by worse error order. Then, a set amount of samples M from the worse ones are used for training while the rest of the samples that were recognized correctly are discarded.



(a)



(b)

Figure 3: Architecture of our networks. (a) is the networks for recognition task. It has convolution, max pooling, maxout, fully connected and classification layers. (b) is the networks for segmentation task. It has the binarization layer instead of classification layer.

Table 2: Architecture of networks for each dataset.

	Recognition task (Fig.3(a))				segmentation task (Fig.3(b))	
	MNIST		Hand Shape		Hand Shape	
layer	type	size, # of kernels	type	size, # of kernels	type	size,# of kernels
input	grayscale	28×28	grayscale	40×40	grayscale	40×40
1st	convolution	5×5, 40	convolution	5×5, 40	convolution	5×5, 40
2nd	max pooling	2×2	max pooling	2×2	max pooling	2×2
3rd	maxout	2	maxout	2	maxout	2
4th	convolution	5×5, 400	convolution	5×5, 40	convolution	5×5,40
5th	max pooling	2×2	max pooling	2×2	max pooling	2×2
6th	maxout	2	maxout	2	maxout	2
7th	fully connected	100	fully connected	100	fully connected	400
output	soft max	10	soft max	6	L2 norm	1600

4.3 Architecture of Networks

The network architecture for our method is shown in Fig.3. It consists of six layer types, i.e. convolutional, max pooling, maxout, fully connected, binarization and classification layers. The architectures are set up individually for each task. As shown in Fig.3(a), the recognition task utilizes the convolutional, max pooling, maxout, fully connected and classification layers, but not the binarization layer. However, the binarization layer comes in handy for the segmentation task, as illustrated in Fig.3(b). Each parameter, such as the kernel size in the convolutional layer and the number of nodes in the fully connected layer, are predefined for the experiment.

Max pooling can help the network achieve faster convergence and improves generalization. Unlike Krizhevsky and Schmidhuber methods, we employ maxout following max pooling as same as (Goodfellow et al., 2013). While max pooling selects the maximum output from the same map produced by a kernel and subsamples the map size to half, maxout selects the maximum value from several maps and reduces their number. The fully connected layer receives high dimensional feature vectors by flattening the output of previous layer. It then trains with dropout and output specific dimensional feature vectors.

For the segmentation task, the feature vectors are input in the binarization layer in order to obtain binary image. This layer output the probability of each pixel of being part of an object or not. The classification layer produces a distribution of each class labels using soft max. When training the network, all the parameters are optimized through backpropagation.

4.4 Input Layer

The network is trained with a large amount of data to reduce over fitting. However, as it is difficult to collect a huge amount of labelled data, the elastic distortions described in 4.2 is employed to increase the amount of data from the initial limited dataset. This data augmentation method is applied to both the original image and the binary labeled data in the segmentation task.

4.5 Pooling Layer

We employ maxout to avoid the drawbacks of a traditional activation function design (Goodfellow et al., 2013). This results in a higher representation compared to a ReLU nonlinearity (Sutskever and Hinton, 2012). The common activation function h_i in $\boldsymbol{h} = (h_1, ...h_i, i \in I)$ is defined as in Eq. (7):

$$h_i = \sigma(x^T W_i + b_i). \quad (7)$$

$\sigma(\cdot)$ is the sigmoid function, x is the input vector, W_i is the weight and b_i is the bias. The maxout selects the maximum value from several filtered output kernels, as Eq.(8):

$$h_i = \max_{j \in [1,k]} z_{ij}, \quad (8)$$

$$z_{ij} = x^T W_{ij} + b_{ij} \quad (9)$$

Unlike max pooling, maxout selects the maximum output from several maps.

4.6 Fully Connected Layer

Dropout is an efficient method to reduce overfitting and improve generalization (Hinton et al., 2012). It randomly samples hidden units with a probability of 50%. The features of the selected units are then used for optimization during each iteration of the training process. Dropout is also used for training in the fully connected layer.

4.7 Binarization Layer

For the segmentation task, we utilize a binarization layer based on a fully connected layer. The output node of the binarization layer receives the value of all the input nodes that are the response of the previous layer. Each node output the probability of corresponding to a pixel of the sample image. As such, the number of output nodes is the same as the number of pixels of the sample image.

4.8 Network Training

All $\boldsymbol{W}$ parameters, such as the kernel elements, the weights between the units and the bias, are randomly initialized. We use backpropagation to update the parameters $\boldsymbol{W}_{t+1}$ in iteration $t+1$ as defined in Eq.(10):

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \varepsilon_{t+1} \frac{\partial E}{\partial \boldsymbol{W}} \quad (10)$$

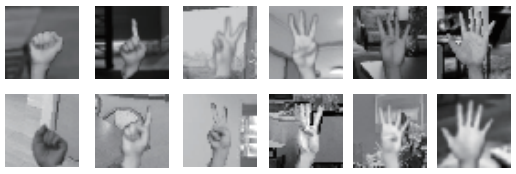


Figure 4: Example of evaluation dataset including 6 classes based on the number of upped fingers. Each hand posture class is identified by the number of up fingers displayed, with ”0” for a closed fist and ”5” for an open palm.

Soft max is used for the loss function E in the classification layer as follows:

$$E_i = \frac{\exp(h_i)}{\sum_{j=1}^{M} \exp(h_j)}, \quad (11)$$

$$E = -log(E_i). \quad (12)$$

E is derived from E_i, the loss function of node i with label y of the training sample, and M is the number of total output nodes.

Meanwhile, we utilize the L2 norm for the loss function E in the binarization layer as described in Eq.(13):

$$E = ||\boldsymbol{h} - \boldsymbol{y}||^2 \quad (13)$$

The update efficiency will decrease depending on the update time t as defined in Eq.(14):

$$\varepsilon_{t+1} = \frac{\varepsilon_0 \tau}{\max(t, \tau)} \quad (14)$$

The ε_0 is initially update-efficient and τ is the defined parameter.

5 EXPERIMENT

We perform an experiment to demonstrate the effect of our proposed exhaustless sample generation and effective selection method for several tasks. First, we evaluate it in MNIST, including ten classes for handwritten digits. Although MNIST is an appropriate dataset for a recognition task, its background is quite simple and it is easy to obtain a high recognition performance when using it with state of the art methods. Because of this, we prepare a hand posture dataset that is more complicated, including object variation like handwritten digits under cluttered backgrounds, as shown in Fig.4. In addition, the dataset has binary images in order to perform the segmentation task. It includes six hand shape classes based on the number of upward fingers and each class has scale, position and rotation variations. We evaluate the recognition performance with both MNIST and our hand posture dataset. We also evaluate the segmentation performance with the hand posture dataset.

Table 3: The classification error depends on the presence or absence of sample generation and selection.

methods	MNIST	Hand Shape
generation & selection	0.57%	9.0%
no generation & selection	0.77%	14.4%
generation & no selection	0.71%	10.6%
no generation & no selection	0.89%	15.6%

The network architectures for each individual dataset are as described in Table 2. In the sample selection, training samples are sorted based on Eqns.(12)(13) in descending order. The first 50% samples are selected and randomly shuffled to reduce the bias in mini batch.

5.1 MNIST

MNIST contains 50000 images of each class for training and 10000 images for testing. The baseline, in which no data generation and data selection is used, is trained only with the original training samples. The method using only sample selection uses the effective samples from the original ones. The method using only sample generation creates and uses the augmented samples from original ones. The method using both generation and selection uses the effective samples from the augmented samples. All methods perform update of the network parameters through 200000 iterations using a mini-batch that included 10 samples from above ones. The methods with sample generation renew the package that contains the augmented samples. The updating parameters τ and ε_0 are set to 20000 and 0.006 respectively. We implemented the methods using Theano library and train the network on a NVIDIA GT640 2GB GPU. The error for each method is displayed in Table 3. The method using both sample generation and selection achieves the best performance. There are just slight differences between the results of the other methods due to the dataset being composed of images with a simple black background.

5.2 Recognition of Hand Shape

To evaluate the network in a more challenging environment, e.g. like under cluttered background, we collect images presenting shape variations in the same class and illumination changes. The basic dataset contains 1600 original grayscale and binary images for each class. Our proposed method synthesizes augmented images from the basic dataset with the background images, as described in previous section. We stored a set amount of augmented samples in the package, here 1000 images. For a mini-batch size of

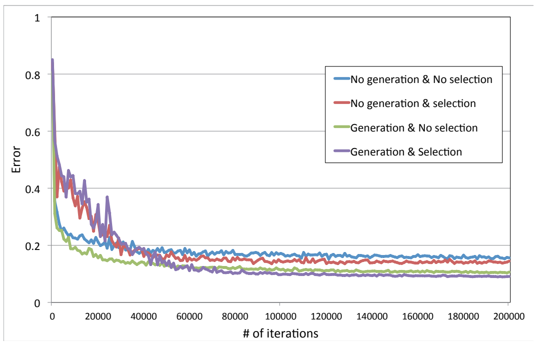


Figure 5: Test error rate in each iteration.

Table 4: The binarization precision-recall rate depends on the presence or absence of data generation and selection.

methods	precision	recall	f value
generation & selection	0.9023	0.9298	0.9158
no generation & selection	0.8957	0.8909	0.8933
generation & no selection	0.9036	0.9249	0.9141
no generation & no selection	0.8958	0.9008	0.8983

10, the number of iterations for the package is 100. The package is updated asynchronously until all the iterations are completed. After the network parameters are updated with 200000 iterations, the total augmented sample number amounted to 2 million individual images. The comparison results are show in Fig. 5 and Table 3. The method with sample generation and selection is best performance in the dataset. We can conclude that this is an efficient method for sample generation and selection.

5.3 Segmentation of Hand Shape

We also evaluate the efficiency of our proposed approach for the segmentation task. We prepare a dataset of binary images of hand shapes. Through data generation, we create the augmented images with their corresponding labeled images. We compare the methods by precision-recall rate as shown in Table 4. From comparison results, the sample generation and selection methods are efficient for segmentation task. The output of the binarization layer with 130000 iterations is shown in Fig.6. Even with a grayscale image input, the network successfully extracts the hand region under a cluttered background. Furthermore, it is robust to not only the background but also to illumination changes.

6 CONCLUSION

We proposed a sample generation and selection approach for ConvNets. Sample generation is per-

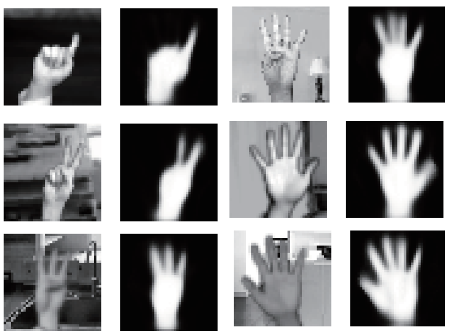


Figure 6: Output of binarization layer: the 1st and 3rd columns are original images and the 2nd and 4th are output images.

formed asynchronously to create a package including augmented samples in order to reduce the processing time. The effective samples are selected from that package and used to update the parameters. The exhaustless sample generation and effective sample selection methods are used in an efficient combination in order to train the network using a huge amount of samples, all the while keeping the sample quality. After comparison, our proposed approach is not only useful for the recognition task, but also for the segmentation task. While this is still in its primitive state for sample generation and selection, we will continue our research by applying this proposed approach to other datasets and more challenging recognition tasks.

REFERENCES

Boureau, Y., Bach, F., LeCun, Y., and j. Ponce (2010). Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR2010)*.

Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR2012)*.

Delakisand, M. and Garcia, C. (2008). Text detection with convolutional neural networks. In *InternationalConference on Computer Vision Theory and Applications (VISAPP 2008)*.

Duffer, S. and Garcia, C. (2007). An online backpropagation algorithm with validation error-based adaptive learning rate. In *In International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 958–962.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *arXiv preprint arXiv:1302.4389*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv preprint arXiv:1207.0580*.

Hubel, D. and Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154.

Jain, V., Murray, J., Roth, F., and Turaga, S. (2007). Supervised learning of image restoration with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV2007)*.

Osadchy, M., LeCun, Y., and Mille, M. (2007). Synergistic face detection and pose estimation with energy-based models. In *Journal of Machine Learning Research*, number 1197-1215, page 8.

Ouyang, W. and Wang, X. (2013a). Joint deep learning for pedestrian detection. In *IEEE International Conference on Computer Vision (ICCV2013)*.

Ouyang, W. and Wang, X. (2013b). Single-pedestrian detection aided by multi-pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR2013)*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing:Explorations in the Microstructures of Cognition*, 1:318–362.

Scherer, D., Muller, A., and S.Behnke (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks(ICANN2010)*.

Sermanet, P., Chintala, S., and LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR 2012)*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *arXiv preprint arXiv:1312.6229*.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *In International Conference on Document Analysis and Recognition*, volume 2, pages 958–962.

Sutskever, A. K. I. and Hinton, G. (2012). magenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25(NIPS2012)*.

Vaillant, R., Monrocq, C., and LeCun, Y. (1994). Original approach for the localisation of objects in images. volume 4, pages 245–250.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *In International Conference on Machine Learning (ICML2013)*.

Y. LeCun ad L. Bottou, Y. B. and P.Haffner (1998). Gradient-basedlearning applied to document recognition. *In Proceedings of the IEEE*, 86(11):2278–2324.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. In *arXiv preprint arXiv:1311.2901*.