Frontiers in Signal Processing, Vol. 4, No. 4, October 2020
https://dx.doi.org/10.22606/fsp.2020.44005

Research on Tibetan Handwritten Numerals Recognition Based on TextCaps Model with Few Training Sample

Hongli Wei, Xiang Qiang*

College of Electrical Information Engineering, Southwest Minzu University, Chengdu, Sichuan, China
Email: xqiang_0426@163.com

Abstract. In recent years, machine learning has been widely used in image recognition while conventional models such as linear classifiers, K-nearest neighbors, non-linear classifiers, and Support Vector Machines (SVM) all require a large number of training samples to train the model to achieve good results. Previous researches on handwritten numerals recognition in Tibetan also need a large number of training samples to train the model. Therefore, this paper conducts training and recognition research on TextCaps model proposed by Vinoj Jayasundara et al., by using Tibetan handwritten numerals data set with a few training samples. In this paper, the training samples of 200 tags for each class are used to add random noise to the original image instantiation parameters and generate new image data sets through the Capsule Network (CapsNet) and its decoder network to achieve the purpose of data expansion. Finally, the character images are classified by the a new model.

Keywords: machine learning, image recognition, CapsNet, TextCaps

1 Introduction

Handwritten numerals in Tibetan are widely used in various fields of Tibetan informatization, such as Tibetan postal code system and Tibetan check data processing. These applications have high requirements on the accuracy of handwritten numerals recognition in Tibetan, and their recognition is directly related to local economic and information construction. First of all, this paper makes a small number of Tibetan handwritten digital picture data set. This data set consists of 10 classes of Tibetan handwritten digital pictures from characters 0 to 9, each class has 500 pictures, a total of 5,000 pictures. The data are from 250 students aged 8 to 15 in The Tibetan Autonomous Region of China, with a sex ratio of 1:1. Each student wrote each character class twice, then took a photo and pre-processed the data. The image was binarized and the pixel of the image was set as 28×28 to obtain the Tibetan handwritten digital image data set D_i.

Among the 500 images of each class, we randomly selected 200 images for training samples, which were added with the data set generated by TextCaps model as the new data set $D_{i,New}$, so as to achieve the effect of sample expansion. The main method to generate the data is to add noise to the instantiation parameter matrix C output from the CapsNet using the perturbation algorithm, and then obtain the reconstructed image through the decoder network in the model. In the process of image reconstruction, we find that the effect of image reconstruction using a single loss function is not as good as that using a combination of two loss functions (marginal loss and the reconstruction loss). We use a combination of marginal loss and the reconstruction loss for training as proposed in [4]. To avoid over fitting. We use cyclic learning rates for each 30 epochs, giving us 3 ensemble models with 90 epochs [5]. In order to verify the validity of the research on the recognition of Tibetan handwritten numerals by the model, our image test set does not include newly generated images，From the original D_i data set, we randomly selected 200 images for each class as the test set. Finally, we obtained an average recognition rate of 98.05%, which was slightly higher than the 97.85% [6] of the CNN model, and the number of training samples we needed was much smaller than that of the CNN model.

2 TextCaps Architecture

TextCaps model is based on the CapsNet. It consists of two parts: the capsule network for obtaining the instantiation parameters of the imageand the decoder network for image reconstruction.

2.1 Capsule Network

The architecture of Capsule Network (CapsNet) is shown in Figure 1. The first three layers are all convolution layers. Conv_1 has 64, 3×3 convolution kernels with a stride of 1, Conv_2 has 128, 3×3 convolution kernels with a stride of 1and the Conv_3 has 256, 3×3 convolution kernels with a stride of 2. The function of the first three convolutional layers is to converts pixel intensities to the activities of local feature detectors that are then used as inputs to the primary capsules. The fourth layer is a primary capsule layer with 32 channels of 8-dimensional capsules, with each primary capsule containing 8 convolutional units with a 9×9 kernel and a stride of 2. The fifth layer, termed as the character capsule layer, is a fully connected capsule layer with a 16 dimensional capsule per class, resulting in M capsules for a dataset with M number of classes. We use dynamic routing between the primary capsule layer and the character capsule layer, as proposed by [4].

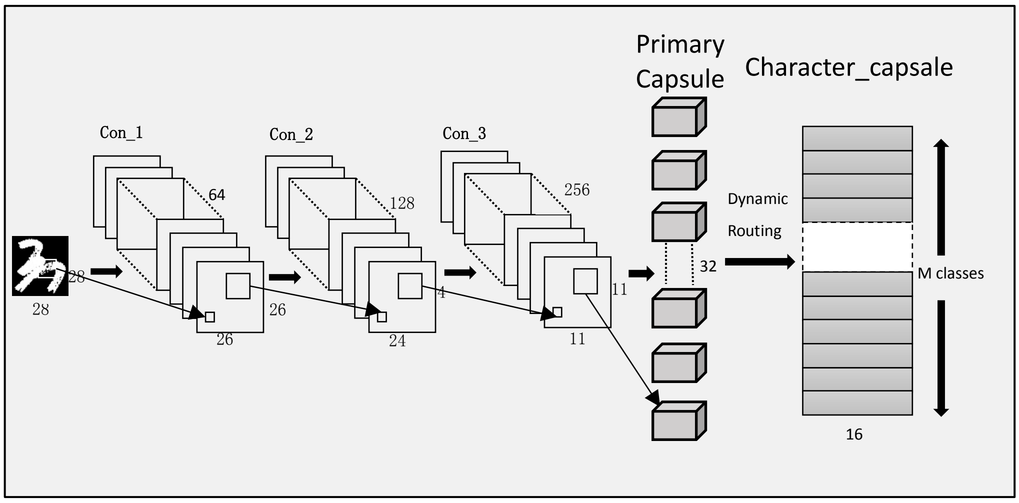

Figure 1. Schematic diagram of capsule network in TextCaps model

The input of the capsule network is a 28×28 image set J, and the output is a $J \times M \times 16$ dimensional tensor C, where C contains the instantiation parameters of the corresponding image. C_j, $j \in [J]$ is the instantiation parameter matrix of the j^{th} training sample. Before passing the instantiation parameter C as input to the decoder network. All activation vectors except for the correct activation vector of the digital capsule are masked with zero. So, the masked tensor $\hat{C}$ is still a $J \times M \times 16$ dimensional matrix, yet the $\hat{C}$ contains only the instantiation parameters of the real class. Then we input $\hat{C}$ into the decoder network.

2.2 TextCaps Decoder

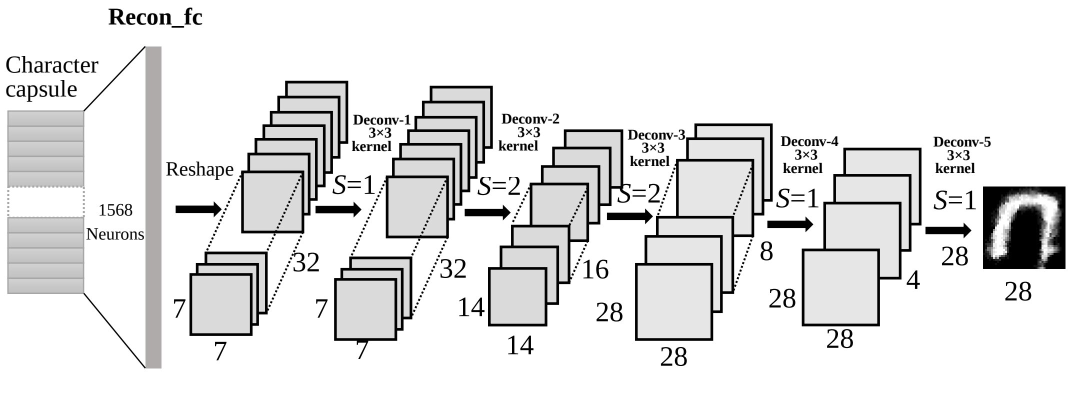

Figure 2. Decoder network for the character reconstruction

The architecture of TextCaps Decoder is shown in Figure 2. It is composed of one fully connected layer and five deconvolutional layers [7]. Its related parameters are shown in the figure, the letter S in the figure represents subsample. The input to the decoder is the masked matrix $\hat{C}$, and the output of the decoder is the set of reconstructed 28 ×28 images.

In the fully connected layer, we use the ReLU[8] activation function. In the deconvolution layers, the ReLU activation function is used in all layers except the Sigmoid activation function used in the last layer. We train the proposed model with a complete training set and evaluate its performance. Then, in order to solve the problem of lack of a large number of training samples in the identification study, we tried to use the same network with 200 training samples for each class to make the model achieve good results. According to the result analysis, in the case of insufficient training samples, the images reconstructed by the decoder network did not achieve satisfactory results. Therefore, it is necessary to expand the sample size. This paper expands the sample size by generating new image data.

3 How to Generate New Image Data

3.1 Generating Instantiation Parameters and Reconstructed Images

In the process of generating new data, the instantiation parameter of the image is needed. Therefore, it is necessary to using the original image data for pre training TextCaps model, and then capsule and decoder network separation in TextCaps model. As shown in figure 3, after separating the capsule network and the decoder network, the initial training samples were input into the trained capsule network model $M_{1,\text{caps}}$ to obtain the instantiation parameter matrix $\hat{C}_{\text{original}}$ of the original image. Prior to passing $\hat{C}_{\text{original}}$ as the input to the decoder network, the corresponding instantiation parameters should be masked with zeros for all the classes except the true class. Finally, we get the reconstructed image I_{recon}.

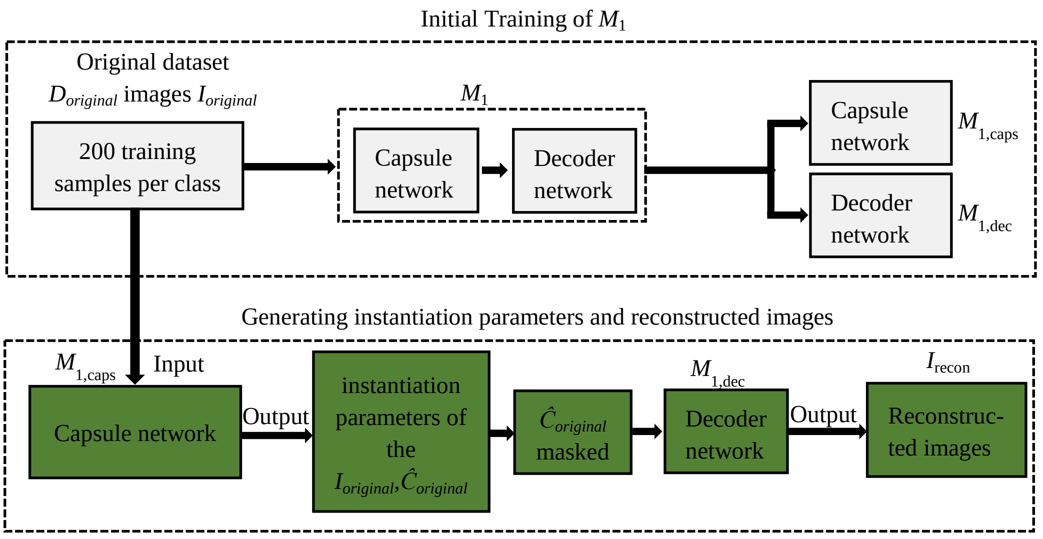
Initial Training of M_1
Original dataset
$D_{original}$ images $I_{original}$
200 training samples per class
M_1
Capsule network
Decoder network
Capsule network
$M_{1,caps}$
Decoder network
$M_{1,dec}$
Generating instantiation parameters and reconstructed images
$M_{1,caps}$
Input
Capsule network
Output
instantiation parameters of the $I_{original}$,$\hat{C}_{original}$
$\hat{C}_{original}$ masked
$M_{1,dec}$
Decoder network
Output
I_{recon}
Reconstructed images

Figure 3. Initial training of M1 and generating instantiation parameters

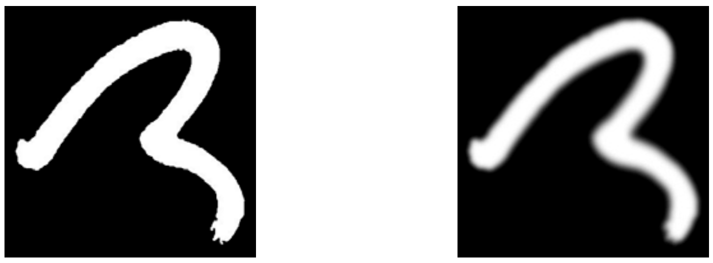

Figure 4. Original and reconstructed Image pairs with model M1

The reconstructed images obtained are shown in Figure 4. With such a small number of training samples, the reconstruction effect is poor. During reconstruction, the image becomes blurred. Therefore, we cannot directly create a new training sample from such an undertrained model. So next we need to solve the problem of blurred reconstructed images.

3.2 Generating New Image Data

In section 3.1, the reconstructed image we obtained is not clear enough, so before using the reconstructed image to generate a new image.We did the processing shown in Figure 5.

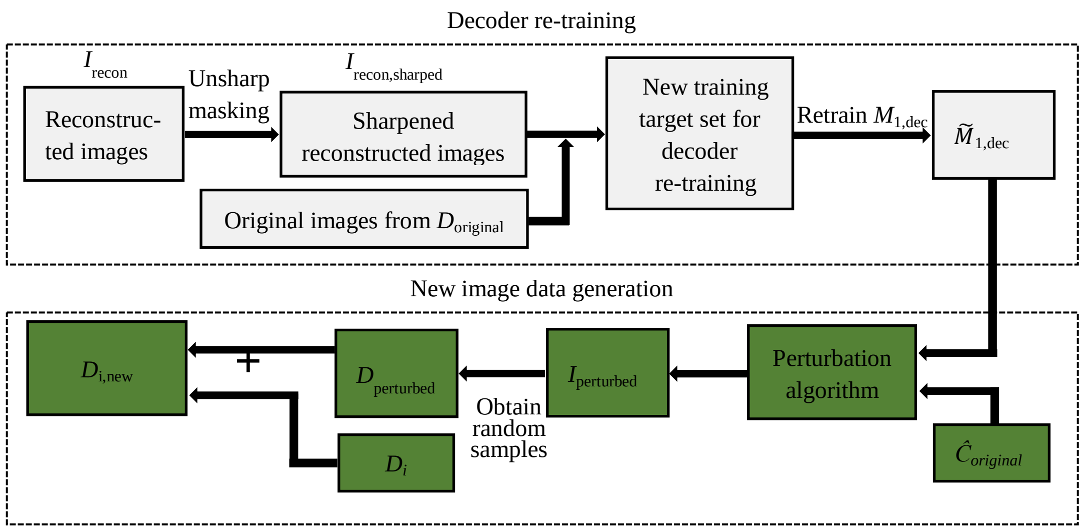
Decoder re-training
I_{recon}
Reconstruc-ted images
Unsharp masking
$I_{\text{recon,sharped}}$
Sharpened reconstructed images
Original images from D_{original}
New training target set for decoder re-training
Retrain $M_{1,\text{dec}}$
$\widetilde{M}_{1,\text{dec}}$
New image data generation
$D_{\text{i,new}}$
+
$D_{\text{perturbed}}$
D_i
Obtain random samples
$I_{\text{perturbed}}$
Perturbation algorithm
$\hat{C}_{original}$

Figure 5. Decoder re-training and new image data generation

For each Reostrued image in I_{recon},we perform unsharp masking[9] with *radius* = 1, *threshold* = 1 and *unsharp trength* = 10 *times*, which sharpens the reconstructed images. Then a new image data set is formed by combining the new sharpened image set and the original image set, and the obtained data set is used to train the decoder network of the model again. we re-train the decoder for 10 epochs , in order to obtain an improved decoder $\widetilde{M}_{1,\text{dec}}$ which provides sharper reconstructions than $M_{1,\text{dec}}$. Subsequently, we perform new data generation by perturbation,For non-zero instantiation parameters in $\hat{C}_{\text{original}}$, we add random controlled noise. In order to avoid distortion in the process of image reconstruction, only one parameter can be modified at a time when the instantiation parameter is modified.

For a given class, there exists a relationship between the variance of an instantiation parameter and the actual physical variations in the generated images, The higher the variance, the greater the change in the image, and vice versa.[4] Hence, for each instantiation parameter $k \in [0,15]$ in each class $(m \in M)$ we calculate the variance, $\sigma_{k,m}$, across all the training samples that belongs tom, so we have16 choices for the value of a, with a = 0 representing the instantiation parameter with the highest variance and a = 15 representing that with lowest variance. For this study, In order to obtain the reconstructed image with large variation, we generate two datasets with a = 0 and a = 1. The specific algorithm implementation process is perturbation algorithm in Table 1.

In perturbation algorithm, $\mathcal{T}_{m,k}$ is the maximum noise that can be added to each instantiation parameter k of the m^{th} class. And the average maximum noise that can be increased for the instantiation parameter k in all classes is represented by the character $\mathcal{T}_k$, Finally, two groups of new training samples were generated by using a = 0 and a = 1 through perturbation algorithm, a total of 500 new images are obtained for each class. Then, as shown in Figure 7, we train a new model M_2 with new data set $D_{\text{i,new}}$, and then retrain the decoder network to obtain the final character classification model.

Table 1. Perturbation algorithm

```
Algorithm 1 Image data generation using perturbation
Input: Instantiation parameters $\hat{C}, a^{th}$ highest variance,
Decoder Network model ($\widetilde{M}_{dec}$).
Output: Perturbed images $I_{\text{perturbed}}$
 1: Calculate class variance $\sigma_{k,m} = \text{var}_j(\hat{C}_{m,j,k})$.
 2: Get $\tilde{\sigma}_{m,k`} \leftarrow sort_k(\sigma_{k,m})$ descending.
 3: Get $\hat{k} = k$ corresponding to $k` = a$.
 4: $\mathcal{T}_{m,k} \leftarrow \frac{max(\hat{C}_{m,j,k})-min(\hat{C}_{m,j,k})}{2}$
 5: get $\mathcal{T}_k \leftarrow \text{avg}_i(\mathcal{T}_{m,k})$
 6: for each $\hat{j} \in [j]$ do
 7:    if $\hat{C}_{m,\hat{j},\hat{k}} > 0$ then
 8:       $\hat{C}_{m,\hat{j},\hat{k}} \leftarrow \hat{C}_{m,\hat{j},\hat{k}} + \min(\mathcal{T}_{m,\hat{k}}, \mathcal{T}_k)$
 9:    else
10:       $\hat{C}_{m,\hat{j},\hat{k}} \leftarrow \hat{C}_{m,\hat{j},\hat{k}} - \min(\mathcal{T}_{m,\hat{k}}, \mathcal{T}_k)$
11: $I_{\text{perturbed}} \leftarrow \widetilde{M}_{dec}(\hat{C})$
```

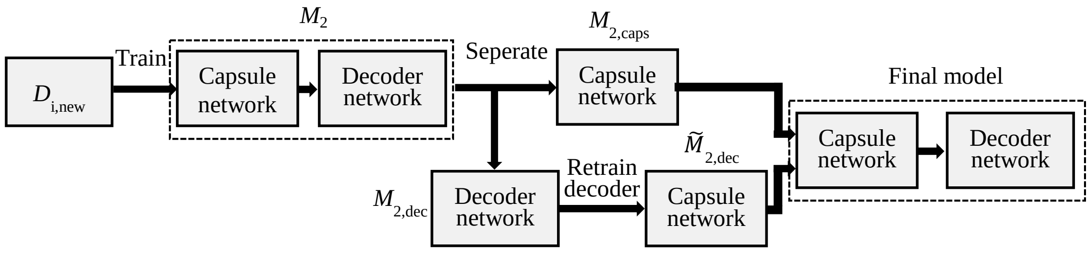

Figure 7. Final character classification model

4 Loss Functions and Results

4.1 Combinations of Loss Functions

As the number of image training samples used in the reconstruction of images is 200, according to the Suggestions of the paper [2], this paper uses the method of combining BCE loss function and SSIM loss function in the selection of the loss function of reconstructed images. The formulas for BCS and SSIDM are given as (1) and (3). In the process of combining the two loss functions, in order to get a better reconstructed image, the two loss functions are not linearly combining mathematically, but the decoder network is fine-tuned. We use two different loss functions corresponding to the two decoder networks to reconstruct the output. Finally, the absolute value of each pixel value difference between the two reconstructed outputs and the test image is compared, and the pixel value closer to the test image is assigned to the final reconstructed output.

Binary Cross-Entropy (BCE). BCE is often used as a measure to identify the difference between two distributions, which is defined by,

$$BCE = -\frac{1}{N}\sum_{i=1}^{n}\left[y(p)\log\left(x(p)\right) + \left(1-y(p)\right)\log\left(1-x(p)\right)\right] \tag{1}$$

DSSIM . We use *SSIM* proposed in [10] to capture spatial relationship between the input image and reconstructed image. *SSIM* for *x*, *y* and the loss function for *SSIM*, structural dissimilarity (*DSSIM*), are defined by,

$$SSIM(x,y) = \frac{(2\mu_x\mu_x + C_1)}{(\mu_x^2 + \mu_y^2 + C_1)} \cdot \frac{(2\sigma_{xy} + C_2)}{(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2}$$

$$DSSIM = \frac{1}{N}\sum_{i=1}^{n} 1 - SSIM(p) \tag{3}$$

In formula (2) and formula (3), μ_x, μ_x and σ_x^2, σ_y^2 are means and variances respectively. $\sigma_{x,y}^2$ is the covariance of reconstructed and true input pixel intensities. $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$ where L is the dynamic range of the pixel values (typically, $2^m - 1$, where m is the number of bits per pixel) and K_1, K_2 are small constants ($K_1 = 0.01$ and $K_2 = 0.03$).

4.2 Experiments and Results

This paper use a combination of marginal loss and the reconstruction loss for training as proposed in [4], and scale down this reconstruction loss by 0:0005 so that it does not dominate the margin loss during training, and further, the training procedure followed for every experiment in this paper is similar to [4].For each dataset, we use ensembling to improve our model accuracy, and to avoid over fitting. We use cyclic learning rates for each 30 epochs, giving us 3 ensemble models with 90 epochs [5]. Finally, the results of handwritten numerals recognition in Tibetan are shown in Table 2.

Table 2. Statistics for recognition results of Tibetan handwritten numerals

Character classes	Train samp/class (CNN)[5]	Recognition rate/% (CNN)[5]	Train samp/class (TextCaps)	Recognitionrate /% (TextCaps)
༠	-	98.5	200	98.0
༡	-	99.0	200	98.5
༢	-	95.5	200	97.5
༣	-	94.5	200	97.0
༤	-	98.0	200	97.0
༥	-	99.5	200	99.0
༦	-	99.5	200	98.5
༧	-	97.5	200	98.0
༨	-	99.5	200	98.5
༩	-	97.0	200	98.5
The total	13000	97.85	2000	98.05

As can be seen from Table 2, TEXTCAPS model achieves good recognition effect when the number of training samples is small. In our opinion, the main reason is that CNN will lose a large amount of image information in the pooling layer during the process of model feature extraction. In CapNet, detailed attitude information (such as the exact position of the object, rotation, thickness, inclination, size, etc.) is saved in the network and does not need to be lost before being recovered. Small changes in the input result in small changes in the output, and the information is saved.

Acknowledgments. The work of this paper is supported by the Southwest Minzu University Graduate Innovative Research Project (Master Program CX2020SZ83). A special acknowledgement should be given to Southwest Minzu University for its experimental conditions and technical support.

References

1. Y. Lee, "Handwritten digit recognition using k nearest-neighbor, radial-basis function, and back propagation neural networks". Neural Computation 3 (1991) 440–449.
2. V.Jayasundara, S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne,and R. Rodrigo ,"Text-Caps: Handwritten Character Recognition with Very Small Datasets", in 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 254-262). IEEE 2019.

3. G.E. Hinton, A. Krizhevsky,and S.D Wang, “Transforming auto-encoders”, in ICANN, Berlin, Heidelberg (2011) 44–51
4. S. Sabour, N. Frosst,, G.E. Hinton, “Dynamic routing between capsules”, in NIPS, Long Beach, CA (2017)3856–3866
5. L.N. Smith, “Cyclical learning rates for training neural networks” In: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, IEEE (2017) 464–472
6. X. Wuji, S. Chajia, Z. Ji, G. Cairang and H. Cairang,“Tibetan handwritten numeral recognition based on convolutional neural network”, Modern Electronics Technique, Vol. 42 No. 5, 10.16652/j.issn.1004-373x,2019.
7. M.D. Zeiler, D. Krishnan, G.W. Taylor,and R. Fergus, “Deconvolutional networks”, in CVPR, San Francisco, CA (2010) 2528–2535
8. G. Xavier , A. Bordes , and Y. Bengio, “Deep Sparse Rectifier Neural Networks”, Journal of Machine Learning Research 15(2011):315-323.
9. A. Polesel, A. Ramponi, V.J. Mathews, “Image enhancement via adaptive unsharp masking”, IEEE Transactions on Image Processing 9 (2000) 505–510
10. Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, “Image quality assessment”, From error visibility to structural similarity. Trans. Img. Proc. 13 (2004) 600–612