Dissertation on Super-Resolution and Noise Removal of Textual Images for Quality Enhancement: A Representation Learning Based Approach

Thesis submitted towards partial fulfillment of the requirements for the degree of

Master in Multimedia Development

Submitted by
Subarno Pal

EXAMINATION ROLL NO.: M4MMD19007
UNIVERSITY REGISTRATION NO.:141018 of 2017-18

Under the guidance of
Dr. Ranjan Parekh

School of Education Technology
Jadavpur University

Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India

2019

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **"**Super-Resolution and Noise Removal of Textual Images for Quality Enhancement: A Representation Learning Based Approach**"** is a bonafide work carried out by Subarno Pal under our supervision and guidance for partial fulfillment of the requirements for the degree of Master in Multimedia Development in School of Education Technology,during the academic session 2018-2019.

SUPERVISOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata-700 032

DEAN -FISLM
Jadavpur University,
Kolkata-700 032

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination for evaluation of Thesis

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **Master in Multimedia Development** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

NAME: SUBARNO PAL

EXAMINATION ROLL NUMBER: M4MMD19007
THESIS TITLE: Super-Resolution and Noise Removal of Textual Images for Quality Enhancement: A Representation Learning Based Approach.

SIGNATURE:
DATE:

Acknowledgement

I feel fortunate while presenting this dissertation at School of Education Technology, Jadavpur University, Kolkata, in the partial fulfillment of the requirement for the degree of Master in Multimedia Development.

I hereby take this opportunity to express warm thanks from the core of my heart to my guide and mentor Dr. Ranjan Parekh for being so kind and supportive throughout the course period. His advice and guidance have paved the path of success of this thesis. He has encouraged me and has guided me with all possible suggestions and constructive criticism. I will remain grateful to him, for my entire life.

I would like to express my warm thanks to Prof. Matangini Chattopadhyay, Director of School of Education Technology for her support and advices.

I would like to thank Mr. Susovan Jana for his support during my thesis work.

I would also like to thank Mrs. Saswati Mukherjee and Mr. Joydeep Mukherjee for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from M.Tech IT (Courseware Engineering) and MMD. I do wish to thank all the departmental support staffs and everyone else who has different contributions to this dissertation.

Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieve this height.

Date:

Place: KOLKATA

Subarno Pal

Master in Multimedia Development
School of Education Technology
Jadavpur University
Kolkata-700032

Super-Resolution and Noise Removal of Textual Images for Quality Enhancement: A Representation Learning Based Approach

Contents

List of Figures

List of Tables

Executive Summary

Optical Character Recognition (OCR) systems depend on quality of their Region of Interest (ROI) for detection of characters from the image. Detection of texts from an image such as license plate image or house number images largely depends on the resolution and noise present in the RIO of the image. Very Low-Resolution Recognition (VLRR) problem occurs when the ROI of the image is smaller than 16x16, and it becomes difficult to recognize even by human experts. Most of the efficient algorithms require minimum 16x16 resolution of ROI of a single character in the image. Images containing lower dimension of ROI gets very hard to be detected by any system trained on normal images. Slightest of noise present in natural images degrades the performance of OCR engines. But both of them are inevitable cases in natural images. Super-Resolution (SR) of images is the process of increasing the resolution of an image for better understanding. For a single character image a deep neural network based model is for upscaling and detection of textual images that show a significant increase in performance on standard datasets. The proposed model consists of a customized autoencoder network for 4x upscaling of an image so that an automated system like the convolutional neural network can detect the upscaled images accurately. Results show a significant increase in the Structural Similarity Index (SSIM) and the Peak Signal to Noise Ratio (PSNR) measures for using the proposed model. Most importantly, the proposed approach achieves 96.44% identification accuracy for the Super-Resolution (SR) images. Fpr multiple character image set we extended this worked and proposed a model for bot super-resolution and denoising. This work focuses on increasing the resolution of the image by four times, and removal of noise from the image. The autoencoder learns the representation of the pixels in lower dimension and noisy images, and gives out clear 4x higher dimensional image. Two different networks are present in the model, responsible for super-resolution and denoising respectively. Quality of the result is measured with Peak Signal to Noise Ratio (PSNR) and Structural Similarity Measure Index (SSIM) values compared with the ground truth (GT). SSIM for super resolution is set to 0.81 and PSNR 14.74. For denoising the values for SSIM and PSNR is 0.85 and 17.77 respectively. These results clearly suggest that the quality of the images have been firmly increased with the proposed model.

1. Introduction

1.1. Overview

Identification of texts from the images depends largely on the size and quality of the Region of Interest (ROI) of the text in the image. Optical Character Recognition (OCR) systems assume a minimum particular resolution for detection of texts from the image. Both single digit and multi-digit identification systems are bound to these criteria. Most of the textual identification systems require a minimum of 16x16 of single character size for proper identification of that character in the image. Very Low Resolution Recognition (VLRR) [1] problem is a matter of serious concern for text identification systems. High amount of noise present in the image also disrupts the performance of the system. These kinds of problems mainly arise due to lack of efficient systems used during the image capture. Image from old sources are also susceptible to these issues. Closed-Circuit television (CCTV) cameras used for surveillance also face this problem a lot. Super-Resolution (SR) is the process of increasing the size of the image for better understanding of the image. Noise removal from noise is also a crucial part of the process because images taken from natural environments are very much likely to be present with noise. Thus quality enhancement of the textual images consists of mainly two parts image super-resolution and noise removal.

1.2. Applications

Quality enhancement of textual images is mainly done for the purpose of retrieval of degraded quality textual image in natural environment. Some of the main areas this system can be installed are:

- *License plate images:* License plate images of motor vehicles obtained from CCTV footage are often very small in dimensions and dusty. Super-Resolution and noise removal are applied for proper identification of the characters and hence recognition of the license plate.
- *Hand writing images recognition:* Old scripts scanned copies are mostly of very small size, the characters present in the images gets hazy and unclear. This system applied on handwritten textual images can enhance the quality the texts in the image.
- *House number recognition:* Autonomous vehicles and drones use cameras for detection of house numbers. SR systems applied in those cameras helps to identify texts very easily.
- *Natural environment OCR:* Different natural environment OCRs are now widely used in practice, their performance can be increased by using this SR and denoising systems.

The proposed model can be used in any type textual images irrespective of language, font or color. Hence, it’s not restricted to only certain domains that are discussed above.

1.3. Challenges

The main challenge of Super-Resolution is to maintain the structural integrity of the image. The image produced by the system has to be sharp and prominent for the OCR systems to identify. Denoising systems are more concerned with removing the noise from the image but also maintaining the structural integrity. Challenges in general can be summed up as:

- Structural integrity: The image before and after processing should have the same structural integrity. If structure gets disrupted then the OCR would confuse the character with any other character. Hence keeping the structural integrity is one of the key factors of the process.
- Signal to noise ratio: Natural environment noises are most unpredictable. Reduction of noise from the images is a key challenge to quality enhancement.
- Shape: The shape of texts in the images varies randomly and can't be assured. The height width aspect ratio also changes accordingly.
- Style: Text fonts vary in style. So, the upscaling method that is proposed has to be independent of the style of font.
- Color: The main challenge with colored images is the contrast of the text with its background. In some of the cases the background is darker and in other it's vice versa.

By studying the existing techniques in this field some of the challenges we would like to address are as follows:

- Interpolation techniques used are very fast and context independent still the quality of the image produced by them is blurry. The structural integrity is hard to find in these cases for more than 2x upscaling cases.
- Deep adversarial networks are hard to train, resource and time consuming techniques. Training these networks frequently on a small dataset is also impossible.
- Noise removal techniques vary with context of different noise present and amount of noise in the image. We tried to propose a unique method for noise removal that can remove a noise occurring to any particular type of image.

1.4. Problem Statement

Quality enhancement of textual images by super-resolution and denoising for proper character recognition purpose.

1.5. Objectives

The objectives of the thesis can be summarized as follows:

- To study the existing methods for super-resolution.
- To study different filter based methods for denoising an image.
- Propose a model for 4x super-resolution for single character input image.
- Propose a combined model for 4x super-resolution for multiple character input image and denoising of it.

1.6. Organization of the Thesis

Section 2 includes a study of existing approaches, section 3 outlines the proposed approach, section 4 tabulates experimental results, section 5 consists of an analysis of the current work side by side other works, and section 6 brings up the conclusion and future scopes.

2. Literature Survey

Single image Super-Resolution is a very highly studied field in computer vision for quite some times. From context independent interpolation methods to highly trained deep learning models, SR has been widely studied through various aspects. Noise removal from images is also a widely studied field. Different filter based methods has been studied here which are applicable to this context. In this segment some notable works relating to our study has been discussed.

A very convenient and context independent approach used in SR is cubic interpolations. Hou et al. [2] in a very remarkable work proposed the use of cubic spilnes for signal upscaling. Their idea was not restricted to image data only but was one of the breakthrough works in the field of super-resolution. The cubic 8-spline being piecewise analytic and spanning five knots is very flexible. Furthermore, it has the advantage of offering good approximation of the function values and its first and second derivatives at the knots. For most engineering interpolation problems, this is a good approximation. Thus they proposed a cubic spilne system. Their experimentation on images was mainly based on binary images. They calculated every column of data using B-splines formula and got the resultant image. Bicubic images showed a great improvement over replication images and bilinear images. The resultant images on zooming showed logical increase in pixels colors. This work was used as a base paper by numerous researchers in later years.

Ruangsang et al. [3] proposed a method for smooth Bicubic Interpolation super-resolution of CCTV images. This work is on efficient SR algorithm using overlapping bicubic interpolation is presented. This scheme is suitable for real-time implementation on many small camera systems with limited processing capability which makes it attractive in the applications of real scenes and Internet of Things.Their idea of super-resolution was to mainly reduce the processing time of the image with minimum hardware used. Bicubic resolution is about considering a cubic spilnes over neighboring pixel. Their method was based on overlapping of previous consecutive frames in the video. A low resolution frame was upscaled and matched with previous high resolution frame to obtain a decent result. They upscaled the low resolution (LR) image by two times. The experimentation and results show improvement than normal Bicubic upscaled image but near to sparse representation results. Their main achievement was they received a high SSIM result with very low computation time and resource.

Zhou et al. [4] proposed a two dimensional wavelet method for single frame super-resolution. They showed that the Bicubic interpolation of high frequency components decomposed by wavelet introduces noise, it will affect reconstruction effect. An improved algorithm using Fourier transform and zero-padding re-sampling instead of bicubic interpolation is proposed by them. The advantage of frequency domain interpolation is obtained by using Fourier transform and zero-padding re-sampling. And high frequency components obtained by wavelet decomposition of the original image can be interpolated optimally without introducing noise, which makes the high frequency details more precise in the reconstruction process. Their experimental results show that the improved algorithm is superior to the traditional two dimensional wavelet reconstruction algorithms, which can be applied to the single-frame remote sensing image super-resolution reconstruction.

Wu et al. [5] proposed an advancement of super-resolution over nearest neighbor embedding. They used a weighted average over the main process to improve their resultant output. Their work was mostly focused to retain back the texture and color of the image. The importance levels of the characteristics are different; they calculated the importance weight value and then embed the value into the SRNE. Each feature of all image patches is represented with a feature vector. Through selecting the feature vector, they restored the information from the high and low frequencies information of the image as much as possible. In this algorithm they choose the Gaussian intensity vector and first-order gradient vector. The experiment results show that the improved algorithm has a good effect in raising the image resolution and eliminating noise. Results clearly show improvement in texture restoration and color.

Elad et al. [6] proposed two iterative algorithms, the R-SD and the R-LMS, to generate the desired image sequence. These algorithms assume the knowledge of the blur, the down-sampling, the sequences motion, and the measurements noise characteristics, and apply a sequential reconstruction process. It has been shown that the computational complexity of these two algorithms makes both of them practically applicable. In this paper, they rederive these algorithms as approximations of the Kalman filter and then carry out a thorough analysis of their performance. For each algorithm, we calculate a bound on its deviation from the Kalman filter performance. We also show that the propagated information matrix within the R-SD algorithm remains sparse in time, thus ensuring the applicability of this algorithm. To support these analytical results some computer simulations on synthetic sequences is shown, which also show the computational feasibility of these algorithms.

Duchon et al. [7] worked on another ground breaking work of super-resolution using Lanczos filtering. This work described enhancement of both one and two dimensional data. Its principle is about using "sigma factors" that significantly reduce Gibbs oscillation in single dimension. A pair of graph was designed to show the magnitude of Gibbs oscillation and the width of the pass band after Lanczos filtering. The weight function is easily modified to get high pass and low pass filtering. Extension of this method to two dimension is used to increases the size of the image.

SRGAN proposed by Ledig et al. [8] introduced adversarial training into this field. A generative network was constructed with Residual network by them. They proposed a perceptual loss function which consists of an adversarial loss and a content loss. The adversarial loss pushes our solution to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images. In addition, we use a content loss motivated by perceptual similarity instead of similarity in pixel space. The deep residual network is able to recover photo-realistic textures from heavily down-sampled images on public benchmarks. An extensive mean-opinion-score (MOS) test shows hugely significant gains in perceptual quality using SRGAN. The MOS scores obtained with SRGAN are closer to those of the original high-resolution images than to those obtained with any state-of-the-art method. This work is notable for texture synthesis of 4x upscaled images. The photo-realistic effect of their resultant images was class apart.

To improve the visual quality of the low-resolution license plate image in the video surveillance system, Yang et al. [9] proposed a new method of super-resolution, namely multi-scale super-resolution convolutional neural network (MSRCNN), it was inspired by Inception architecture of GoogLeNet. The proposed method uses different sizes of filters for parallel convolution to obtain different features of the low-resolution license plate image, and then the features can be used by the layer of concatenation. Finally, the high-resolution images can be reconstructed through non-linear mapping. Experimental results prove that the proposed method can improve the quality of low-resolution license plate image obviously.

Ghosh et al. [10] proposed a method for to construct higher order Gaussian Derivatives as a linear combination of lower order derivatives at different scales for image noise reduction. Based on this, a new kernel which simulates non-classical receptive fields with extended inhibitory surrounds had been proposed. It is easy to implement and finds justification from an old psychophysical angle too. The proposed kernel has been shown to perform better than the well-known Laplacian kernel, which models the classical excitatory-inhibitory receptive fields.

George et al. [11] showed a comparative study on different types of median filters for noise reduction. The methods they suggested first detected the noise and then applied proper median filter for that type of noise. They compared between various types of median filters such as recursive median filter, iterative median filter, directional median filter, weighted median filter, adaptive median filter progressive switching median filter and threshold median filter. From the results, they concluded that threshold median filters are better method for removal of noise from images, because it's the only process the noisy pixel and thus helps to retain the features and edges of an image. Most of the median filters discussed above either couldn't remove the noise and PSNT didn't decrease or couldn't restore the features in image. It was clear from their work threshold median filter met the two criteria better than the rest.

To overcome these problems discussed in various techniques auto-encoder model is proposed that is responsible for SR and noise reduction from the images.

3. Proposed Methodology

3.1. Overview

Optical Character Recognition (OCR) systems highly depend on the quality of the Region of Interest of the image. The main challenge that any OCR system faces is the size of the image. Different types of filtering techniques are present that removes the noise at a certain extent but size of the image is a major concern. The proposed methodology mainly consists of two parts, one for the single character and other for the multi-character system. We proposed an autoencoder in both the cases for super-resolution and denoising of the image. The details of the proposed model and the training procedure are described in the following sections.

3.2. Motivation

Autoencoders are an unsupervised learning technique in which neural are leveraged into networks for the task of representation learning. Specifically, neural network is a designed architecture such that a bottleneck is imposed in the network which forces a compressed knowledge representation of the original input. If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task. However, if some sort of structure exists in the data, i.e. correlations between input features, this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck.

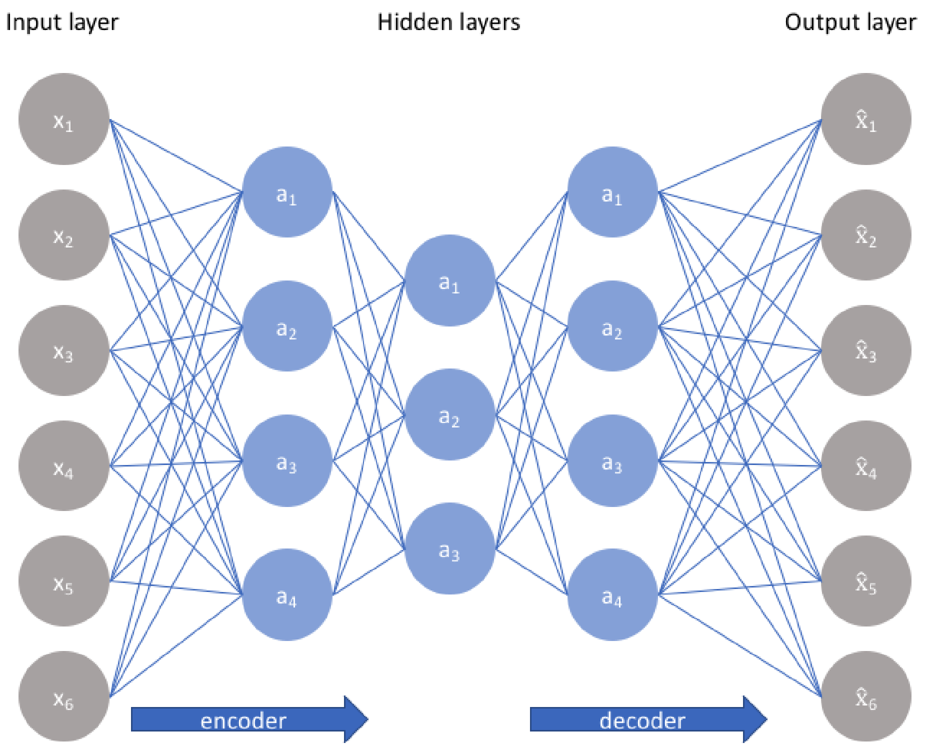


Fig. 1. Autoencoder Network

In general an autoencoder consists of mainly two parts an encoder and a decoder network. Encoder networks encodes the data to a compressed format, decoder network decodes the data to its original form ideally. In fig. 1 we show a very basic form of autoencoder formed by dense nodes. This network forms a bottleneck in the middle hence exact reconstruction of data is not possible.

A convolutional autoencoder is an autoencoder formed by mainly convolutional hidden layers. The three layered RGB image is compressed into latent space representation by the convolution and maxpooling layers. The latent space representation is then converted back into RGB image with convolution and upsampling.

The Convolution layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which *max pooling* is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in CNN architecture. The pooling operation provides another form of translation invariance.

Upsampling layers or decovolution layers are mainly used to increase the size of the layer in the network. Learnable deconvolution layer is also referred to as transpose convolution. Upsampling layers helps to increase the size of the latent vector in two dimensional space.

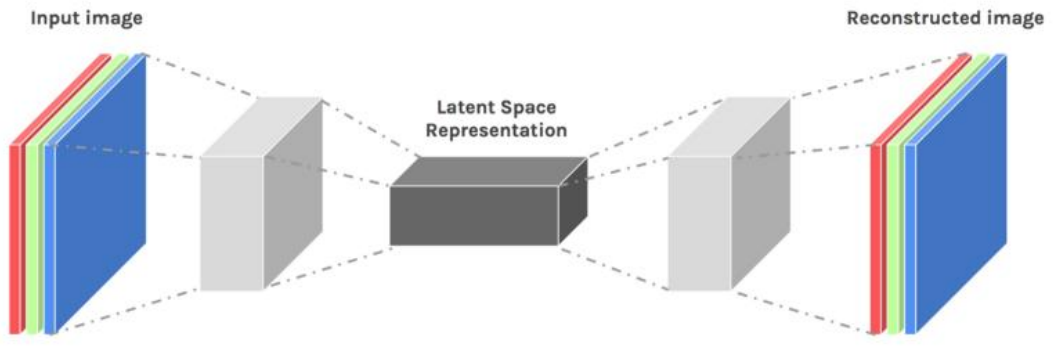


Fig. 2. Basic Convolutional Autoencoder

In fig. 2 we show a very basic convolutional autoencoder where the input image is transferred into latent space and brought back to reconstructed image. The RGB sequence of the output image is same as that of input image.
Our idea of super-resolution and denoising is mainly based on this convolution autoencoder network. In standard models input and output for training are exactly same where as the in our case it's varied. In case of SR model we used low resolution images as input and original high resolution images as output. In case of denoising systems the noisy images are input and clean images are output. The details of the networks designed are described in the following upcoming sections.

3.2. Autoencoder in the light of Representation Learning

The performance of machine learning methods is heavily dependent on the choice of data representation (or features) on which they are applied. For that reason, much of the actual effort in deploying machine learning algorithms goes into the design of preprocessing pipelines and data transformations that result in a representation of the data that can support effective machine learning. Such feature engineering is important but labor-intensive and highlights the weakness of current learning algorithms: their inability to extract and organize the discriminative information from the data. For cases like super-resolution and noise removal such feature engineering processes are not feasible or are less generalized.

In order to expand the scope and ease of applicability of machine learning, it would be highly desirable to make learning algorithms less dependent on feature engineering, so that novel applications could be constructed faster, and more importantly, to make progress towards Artificial Intelligence (AI). Learning representations of the data makes it easier to extract useful information when building classifiers or other predictors. In the case of probabilistic models, a good representation is often one that captures the posterior distribution of the underlying explanatory factors for the observed input. A good representation is also one that is useful as input to a supervised predictor. Hence super-resolution and denoising models are extremely important for prediction of text form textual image inputs. The model proposed for Super-Resolution learns the representation of data from the low resolution images and constructs the high resolution images. The encoder network in the autoencoder is mainly responsible for this learning procedure. The latent space representation is formed based on this representation learning algorithm.

The denoising autoencoder is an improved version of Principle Component Analysis (PCA) that learns the representation of the pixels from the image. The noisy pixels in the images are eliminated by this PCA technique which is just another form of representation learning.

Representation learning is a form of unsupervised learning. There is no labeled data for training and testing. Downscaled and noisy images are set as input to the autoencoders. The autoencoder tries to learn the useful data representation in the data and reconstruct the actual data in upscaled size.

3.3. Single Character Super-Resolution System

Single character Super-Resolution system consists of a single image as input which contains a single character. Low resolution image enters the autoencoder and four times (4x) SR image comes out from the output end. The SR image is thus passed into a normal OCR system such a convolutional network and the prediction class is obtained. The SR autoencoder enhances the input of the convolution neural network hence helping it to increase the accuracy of the system as a whole.

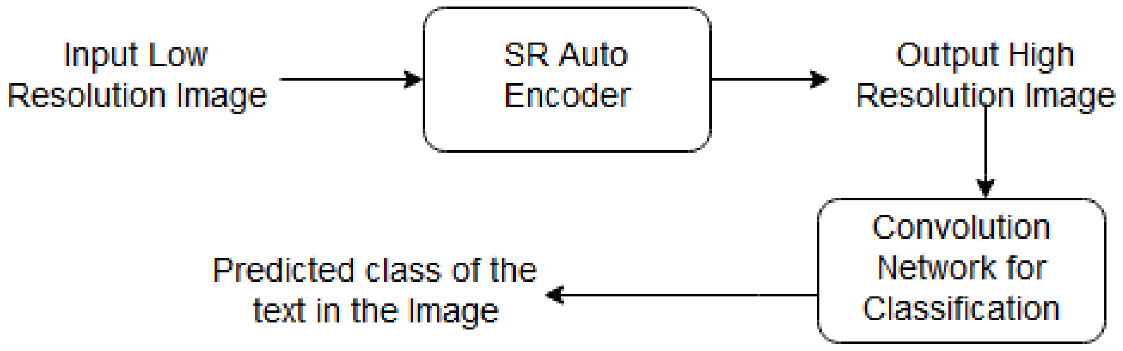

Input Low Resolution Image
SR Auto Encoder
Output High Resolution Image
Convolution Network for Classification
Predicted class of the text in the Image

Fig. 3. Flowchart of SR and Identification of single character image.

A modified convolutional autoencoder model is proposed for image super-resolution favorable for text detection. The autoencoder network consists of mainly convolution blocks, which were fitted one after another. It takes input a low-resolution image and gives a higher resolution version of it as output. A low-resolution image is taken and converted into grayscale. This single-channel grayscale image is the input to the system.

Formally, given a set of $Y = \lfloor y_1, y_2, ..., y_n \rfloor$, where $y_i \in R^d$, autoencoder is trained to minimize the reconstruction error to properly modeled data. The loss function ‘l’ is defined in eqn. (1).

$$l = \sum_i \| y_i - \hat{y}_i \|^2 \qquad (1)$$

y_i is the actual HR image and $\hat{y}_i$ is the reconstructed SR form of the LR image. The hidden layers of the network are mainly involved in encoding and decoding of the data.

$$h_i = f(Wy_i + b) \qquad (2)$$

$$\hat{y}_i = f(W'h_i + b') \qquad (3)$$

h_i denotes the encoded compact representation of data in the network. Refer to eqn. (2). $\hat{y}_i$ is the reconstructed representation of the image. Refer to eqn. (3). Activation functions are at the core of the network in every layer and it is denoted by *f(z)*.
Different activation functions are used at different stages of the network viz. ‘sigmoid’ and ‘ReLU’ mentioned in eqn. (4) and eqn. (5) respectively.

$$f(z) = \frac{1}{1+e^{-z}} \qquad (4)$$

$$f(z) = \max(0, z) \qquad (5)$$

Single layer encoder decoder is not very efficient in learning complex varied datasets representations. To address this problem, a deep layered architecture is constructed for increasing the nonlinearity of the system.

A greyscale version of the image is sent into the input layer of the network. The single channeled image forms the input to the network. Followed by the input layer the autoencoder consist of two more layers for encoding followed by four decoding layers. The encoder part of the network encodes the input to low dimensions. The encoder network consists of two convolution blocks 3x3 kernel size which results to two encoded layers. The convolution layers are followed by an upsampling layer which forms the compressed data or latent representation vector of the distribution. The encoded data is now decoded using a combination of 3x3 kernel size convolution layer followed by an upsampling layer with the 2x2 upsampling factor. This combination is used thrice in the network which is finally followed by a 3x3 convolution layer. The activation function of all the inner layers is 'ReLU', only the final layer has ‘sigmoid’. Fig. 4 visualizes the structure of the autoencoder network. The output of the network is a single channel higher resolution version of the input image which is used for text detection in the next subsection.

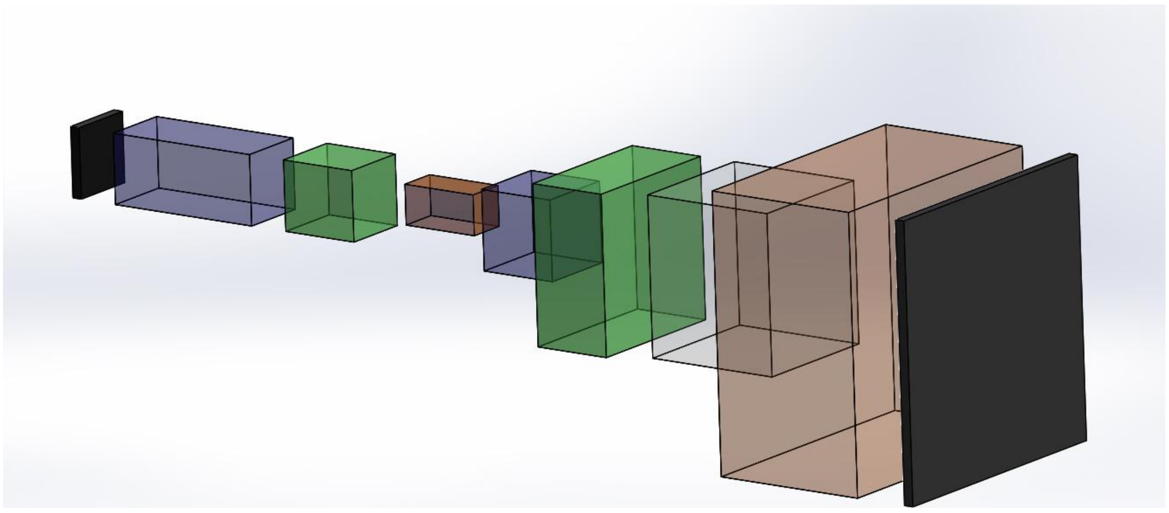


Fig. 4. Detailed Architecture of single image SR autoencoder network.

The text detection from the image is done using a simple convolutional neural network. The detection convolutional network consists of two convolution layers followed by a max-pooling layer. The max-pooling layer is connected to two successive dense layers for classification. ‘ReLu’ activation function described is used in the layers to add nonlinearity in the inner layers. Only the outer layer has a softmax function. Refer to eqn. (6). Here, z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z) and again, j indexes the output units as j = 1, 2, ..., K.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad (6)$$

Any type of convolution network can be used for identification purpose. Here, this configuration of the convolution neural network is used to validate the results of the autoencoder.
The detailed architecture shows how the SR model is constructed. We take this network as the basis of constructing the next SR network for multiple characters set.

3.4. Multiple Character Super-Resolution and Denoising System

We propose a convolutional autoencoder network for super-resolution and denoising of the image through subsequent phases. Our proposed model of autoencoder has two distinctive parts one for upsampling and another for denoising trained separately. The combined network can be used together and distinctively according to the user's wish.

Autoencoders are a type of neural network that are known for reproducing its input, i.e. the target output is the input. The eqn. 7 depicts the mapping of a classical autoencoder, where h and r depicts the transition of encoder and decoder respectively.

$$h : X \to F$$

$$r : F \to X$$

$$h, r = argmin_{h.r} ||X - (r \circ h)X||^2 \qquad (7)$$

Convolution autoencoders are convolutional and maxpooling layers forming the layers of the network. The convolutional model learns the optimal filters that reduce the reconstruction error. These type of autoencoders are known for copying the input to output, but in our case we modify this autoencoder in some relatable ways. The smaller dimension image x' of x is passed into encoder function of $h = f(x')$ and decoder produces a reconstruction of $r = g(h)=g(f((x'))$. In simple words the lose function can be defined as $L(x,\ g(f((x')))$.

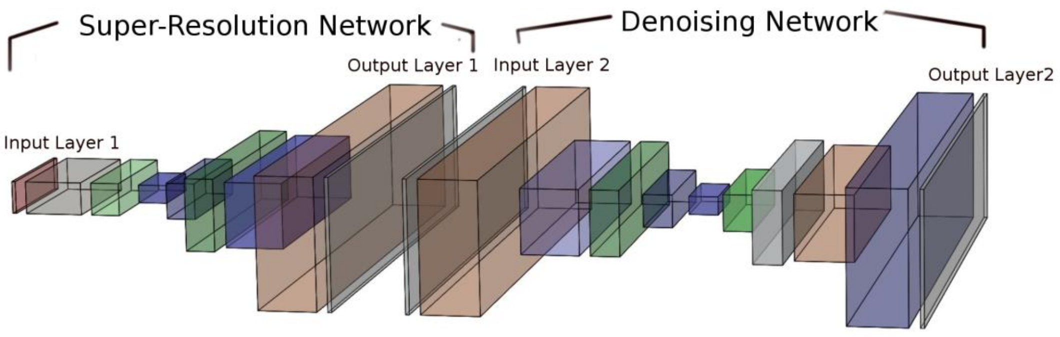


Fig. 5. Quality Enhancement Autoencoder Network for multiple character images.

The autoencoder we use in this case is convolutional autoencoder mainly consisting of the convolutional layers. Normal autoencoders can learn any distribution function whereas convolutional autoencoders are specialized for image data. The input to the network is grayscale image of shape (9, 37, 1). We upscale this image in our model by four times to (36, 148, 1). The input layer is followed by a convolution layer of stride (1, 1) with activation function ReLU defined in eqn. 8. The convolution layer is followed by max pooling layer of stride (1, 1). This combination of convolution and max pooling is again used but with strides (3, 3) and (2, 2) which completes the encoder. The encoder encodes the lower dimensional data to be passed into the decoder. The decoder network consists of three combinations of convolution layer and upsampling layer. The convolution layers has stride (3, 3) with ReLU activation discussed in eqn 2. The stride value for upsampling layers is (2, 2). The final layer of the decoder consists of a convolution layer of stride (3, 3) and sigmoid activation function eqn. 9. Adadelta optimizer is used to train this autoencoder network that updates the learning rates according to moving window of gradient updates. The loss function used is binary cross-entropy described in eqn. 10.

The denoising network described in the model consists of an encoder and decoder part. The input to this network is noisy image and output from the network is cleaned up image. The network takes the necessary pixels that are not responsible for noise and presents a clean version of the image. The encoder network consists of three pairs of combination of convolution and maxpooling layers of strides (3, 3) and (2, 2) respectively. The ReLU activation function is used in the convolution layers. The decoder network used in the network consists of combinations of convolution network and upsampling network of stride (3, 3) and (2, 2) respectively. The final layer of the decoder network is again a convolution layer of stride (3, 3) and sigmoid activation function. This model is trained with adadelta optimizer and loss function is set to binary cross-entropy.
Eqn. 3 describes the ReLU activation function g(x).

$$g(x) = max\{0, x\} \qquad (8)$$

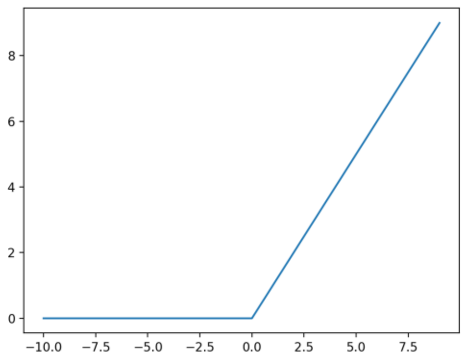


Fig. 6. ReLU Activation function

Eqn. 8 describes the sigmoid activation function g(x).

$$g(x) = \frac{1}{1+e^{-x}} \qquad (9)$$

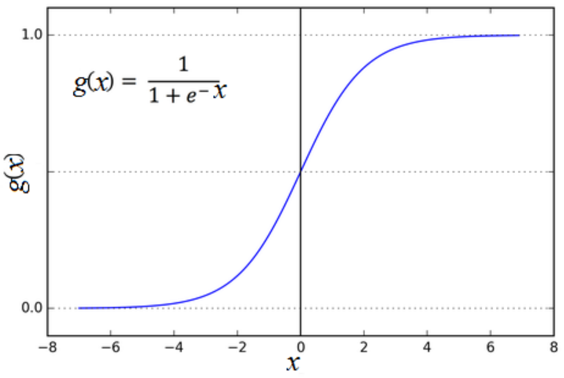


Fig. 7. Sigmoid Activation function

The binary cross entropy loss is defined as

$$loss = -\sum_{c=1}^{2} y_{o,c} \log(p_{o,c}) \qquad (10)$$

Where c is the number of classes, y is binary indicator (1 if correct, 0 if incorrect) and p is predicted probability observation o of class c.
Fig. 5 demonstrates the structure of the model in detail, where the blocks depict the input and output

of the convolution operations. The low-scale input image is passed from the left of the network as shown which reaches its 4x resolutions by output1. The second network cleans the image from noise present after super-resolution. Hence the Output Layer2 is the final output of the model.

4. Experimentation

4.1. Overview

The experimental process in a nutshell can be described as taking a dataset, decreasing the quality of the image with different techniques, enhancing the quality of those images with the proposed model and finally comparing the enhanced image with original high resolution (HR) image. In fig. 8 the detailed data flow diagram of the experimental procedure is shown.

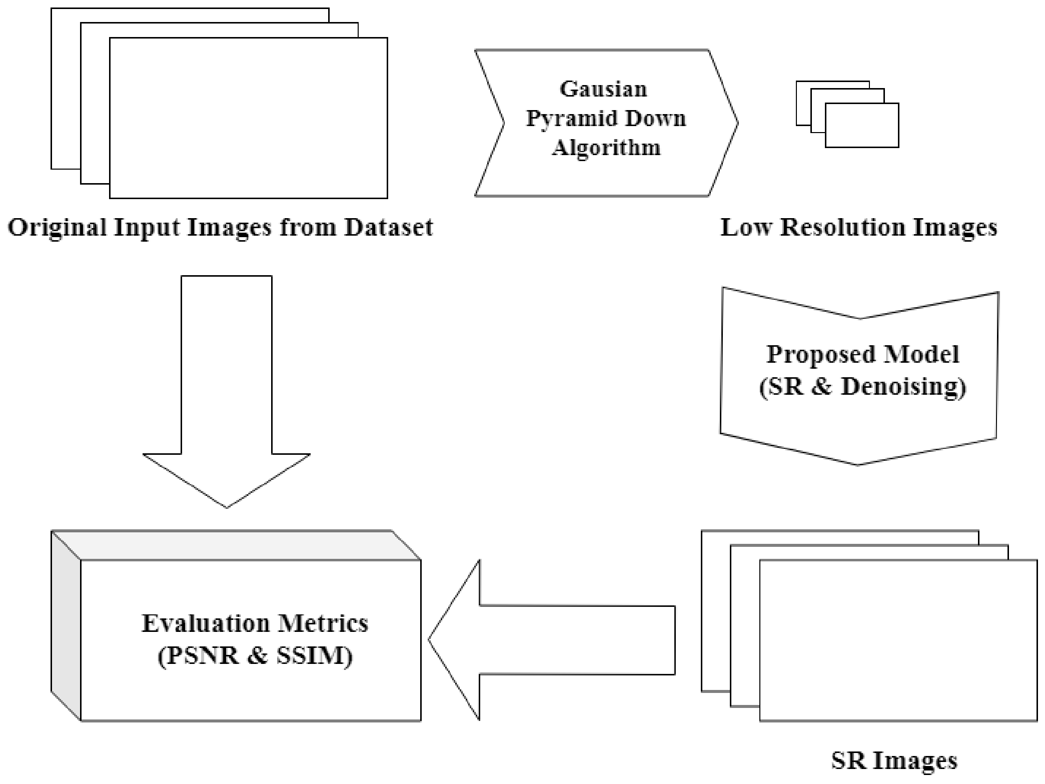

Gausian Pyramid Down Algorithm
Original Input Images from Dataset
Low Resolution Images
Proposed Model (SR & Denoising)
Evaluation Metrics (PSNR & SSIM)
SR Images

Fig. 8. Dataflow diagram of Experimentation procedure

4.1. Single Character Image Super-Resolution Autoencoder Network

The single image SR autoencoder was trained and tested on two different datasets. In this phase, the main focus is given on textual image super-resolution for proper text identification. The experimentation was done on two datasets, which were widely used for text recognition purpose (a) MNIST - consisting of handwritten images and (b) SVHN - street view house number dataset. The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly hard, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images. The cropped digit format of the image set is used for experimentation.

4.1. 1. Preprocessing

The images in MNIST dataset contains uniform background and hence doesn't require any pre-processing. The SVHN dataset contains RGB images of highly varied texture and color. So the dataset was preprocessed to bring uniformity in the distribution. Firstly, the RGB images were converted to grayscale images. Next, the dataset consisted of two main types of images, i.e. one with 'lighter background' and other with 'darker background'. Otsu's binarization was applied to the grayscale image. Left corner of the binary image was analyzed to find which of them has a lighter background.
The images with lighter backgrounds were inverted and further converted into images with 'darker background'.

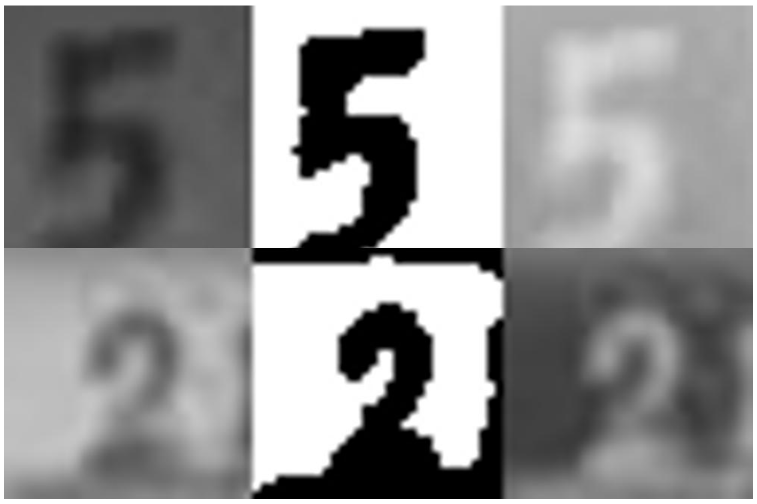


Fig. 9. Preprocessing of image data in SVHN dataset
Right-most column showing final output, i.e. all images with darker background.

Fig. 9 shows how the preprocessing is done on the images. The preprocessing shows that the images with lighter backgrounds are identified and inverted. Inversion of those images transforms them into images with darker background as shown in the 3rd column.

4.1.2. Training

At first, the images are converted into a low-resolution image by 4x by the pyramid-down algorithm. Then, the lower resolution image is reconstructed into 4x higher resolution by our model. The content loss (*l*) of the image is calculated by measuring the reconstruction activation loss. The network is trained with back-propagation throughout the system. The network is trained approximately 100 epochs on the training set to obtain the output.

4.2. Multiple Character Image Super-Resolution Autoencoder Network

The single image SR autoencoder was trained and tested on two different datasets. The model is trained with a set of 10,821. The training set is divided into training and test set in 3:1 cross fold ratio to be fed into the network.

We train the super-resolution network with a set of license plate image along with their low resolution version. We took a dataset consisting of normal license plate image of dimensions (36, 148, 1) downscaled those images 4x to (9, 37, 1). The downscaled images are obtained by Gaussian pyramid down algorithm that removes one of consecutive rows and columns to obtain an image half the size of the original image. The formula is shown in eqn. 6. $w(m,n)$ is the weighting function which is used to generate the pyramid at each level. For simplicity the weighted function is taken to $w(m,n) = w(m) * w(n)$. Two successive iterations of this operation leave us with 4x downscaled images. This is basically a semi-supervised learning algorithm where the network learns from low resolution image to form the high resolution image.

$$g_l(i,j) = \sum_{n=-2}^{2} \sum_{m=-2}^{2} w(m,n) g_{l-1}(2i+m, 1j+n) \quad (11)$$

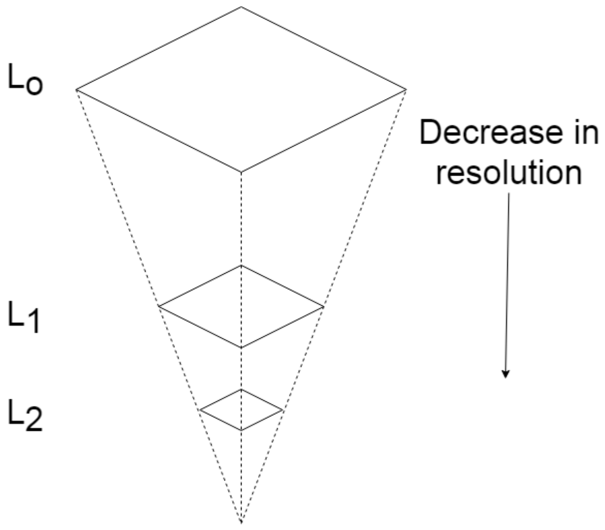


Fig. 10.Gaussian Pyramid Algorithm

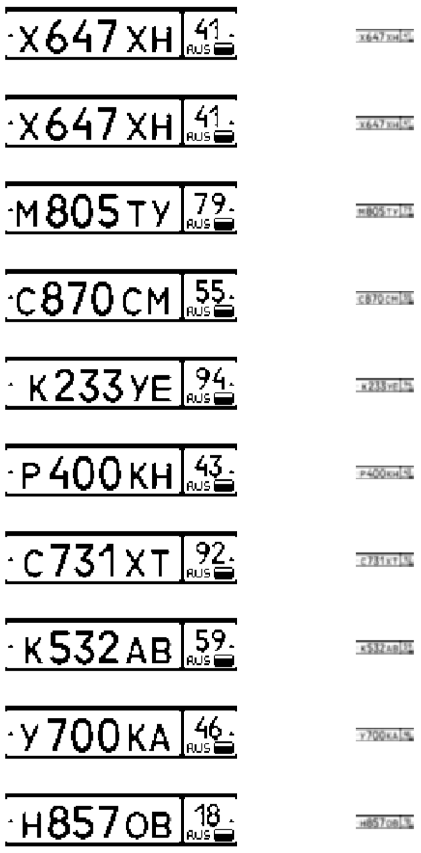


Fig. 11. Downscaled samples (i) Original HR image (ii) downscaled version of them

Fig. 10 shows the Gaussian pyramid algorithm in structural view. L_0 is the primary level representing the original HR image. L_1 is the 2x downscaled version of the image and L_2 is the 4x downscaled version. Our task is to deal with the L_2 version of the image and bring it back to L_0 by upscaling.
In fig. 11 we show some examples of license plate image from the dataset and their 4x downscaled version.
For the donising part we add random noise to the existing fresh dataset elements. Random noise looks very familiar to Gaussian noise or normal noise but does not follow any particular pattern. Hence random noise becomes much more difficult to be removed. In eqn. 12 $rand(x, y)$ generates an image of size (x, y) containing random grayscale values. So, the eqn. describes the formation of noisy image from the original set of images for training and testing the network.

$$I_{x,y}^{Noise} = rand(x, y)$$

$$I_{x,y}^{new} = I_{x,y}^{original} + I_{x,y}^{noise} \qquad (12)$$

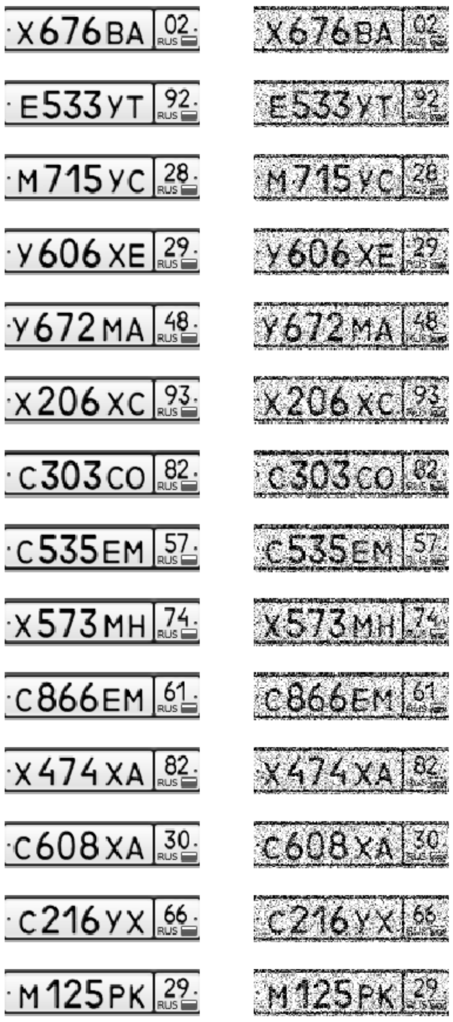


Fig. 12. Sample of Noisy Image with their original HR

Fig. 12 represents some examples of images from the dataset and their noise added versions.

4.2.1. Training

The denoisisng network is trained with noisy images and there corresponding clean ones. Clean set of noisy images are taken and random noise is added to it on a scale of 0.3. On an experience basis it has been seen the noise present in license plate images are much alike to Gaussian noise or normal noise. The noisy images are taken as input to the denoising network and cleaner ones are expected as output. Our dataset consists of 10,821 training images and 561for validation. The training data is split in 3:1 ratio for training and testing the network. The validation data is kept separated while training the network.

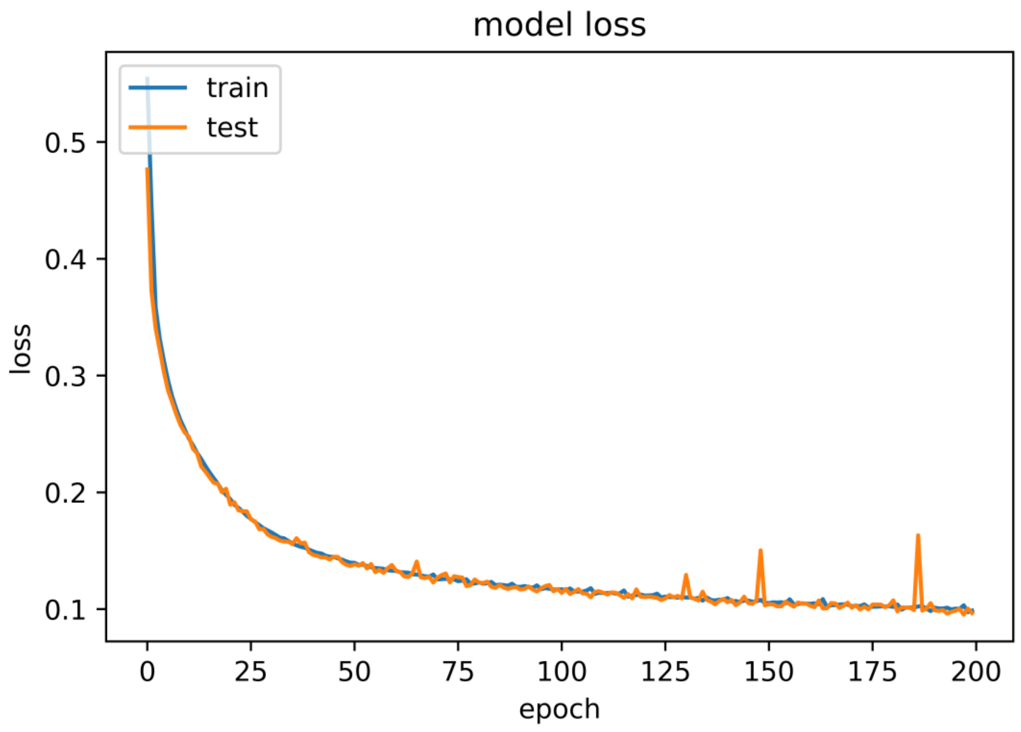


Fig. 13. . Training loss of SR Autoencoder.

The super-resolution network and denoisisng network is trained separately. The super-resolution network is trained separately for 200 epochs where the loss converges to minimum. The denoising network trains for 300 epochs. The batch size is set to 128 for both the case and data is shuffled during the training session. In fig 5 and 6 we plot the training loss of both the networks. The plot suggests a stable reduced loss is achieved in both the case with successive epochs of training.

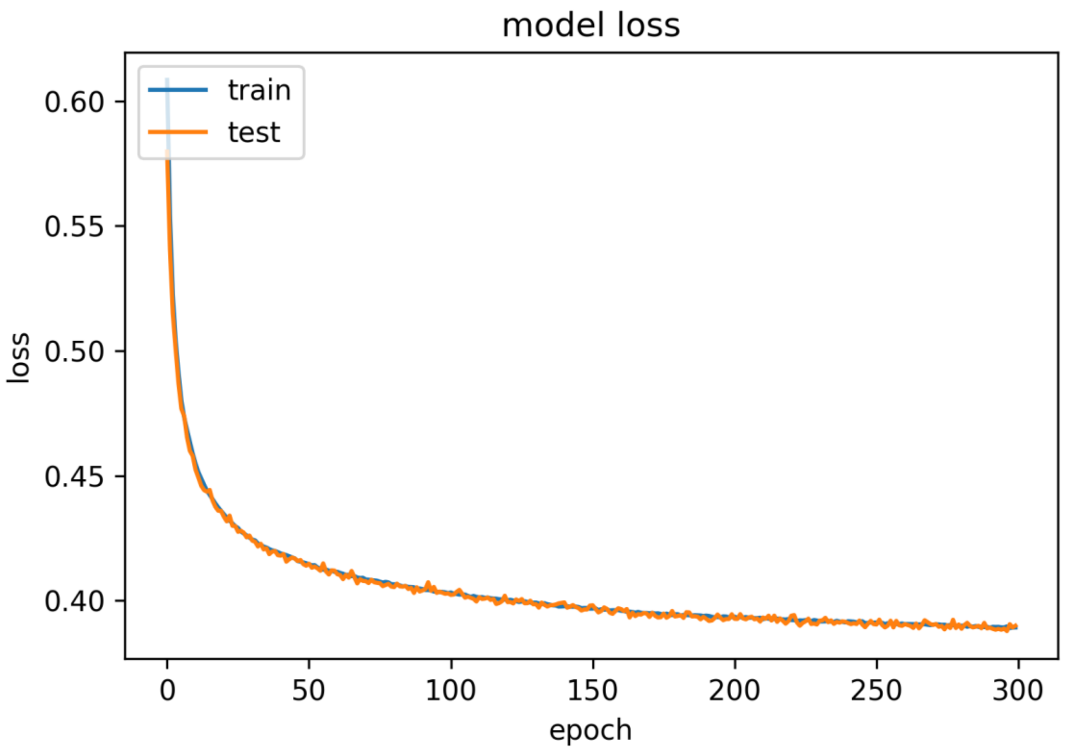


Fig. 14. Training loss of Denoising Network.

Fig 13 and 14 shows stable training loss and test loss. Thus we can confirm that the model is trained properly and there is no chance of overfitting or underfitting. This confirms the selection of proper hyper-parameters for training the model.

5. Result and Analysis

5.1. Analysis Measures

The quantitative analysis presents the metrical overview of our work based on PSNR and SSIM metrics. The qualitative analysis shows how the resultant image has been visually improved. And finally we discuss the efficiency of the model, in context of its training time, test time and hardware requirements.

Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) are the main two metrics for analyzing the model.
PSNR [12] is the ratio of maximum possible power of a signal to the power of the corrupting noise. The signal in this case is the original data and the noise is the error due to reconstruction.

$$PSNR = 10.log_{10}\left(\frac{255^2 * N}{||f_R - f_{Ref}||^2}\right) \qquad (13)$$

In the eqn. 13 N is the total number of pixel in the image, f_R is the reconstructed image and f_{Ref} is the referential image. Higher the PSNR better the reconstructed image.

SSIM [13] is the measure of similarity between two images. SSIM is calculated for two similar window x and y of size NxN.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \qquad (14)$$

In the above eqn. 14 $\boldsymbol{\mu_x}$ is the average of x, $\boldsymbol{\mu_y}$ is the average of y,$\boldsymbol{\sigma_x^2}$,$\boldsymbol{\sigma_y^2}$is variance of x and y respectively, $\boldsymbol{c_1}$and $\boldsymbol{c_2}$ two variables to stabilize the division with weak denominator.

5.2. Comparative Analysis Study

In this part of the thesis we compare our model with the existing models that were used previously. PSNR and SSIM are the two main criteria for the comparative study.

5.2.1. Single Character Image Super-Resolution Autoencoder Network

The test data was down-sampled by 4x using pyramid down algorithm and it was used as test set of our model. The SR image obtained from our network was compared using PSNR and SSIM values with the original High Resolution of the image.
We evaluate the accuracy of the system by PSNR and SSIM methods. Finally these images are used for identification and we show a high increase in accuracy of the system for our model. The PSNR and SSIM measures are computed using SciPy [19] package for authentic verification. The PSNR and SSIM values are detailed in the following table:

TABLE I. MNIST SR COMPARISON WITH VARIOUS METHODS

SR Method	PSNR	SSIM
Nearest Neighbor Interpolation	12.15	0.18
Lanczos Interpolation	12.42	0.21
Bicubic Interpolation	12.40	0.21
Autoencoder (Proposed model)	**19.79**	**0.85**

TABLE II. SVHN SR COMPARISON WITH VARIOUS METHODS

SR Method	PSNR	SSIM
Nearest Neighbor Interpolation	21.63	0.46
Lanczos Interpolation	22.08	0.52
Bicubic Interpolation	22.02	0.51
Autoencoder (Proposed model)	**25.85**	**0.73**

Thus we can see clearly from table I and II that the PSNR and SSIM value remarkably increase for using our Autoencoder network in both the datasets. The test results obtained are calculated after training the autoencoder for 100 epochs. Increase in training time might increase the accuracy of the system even further.

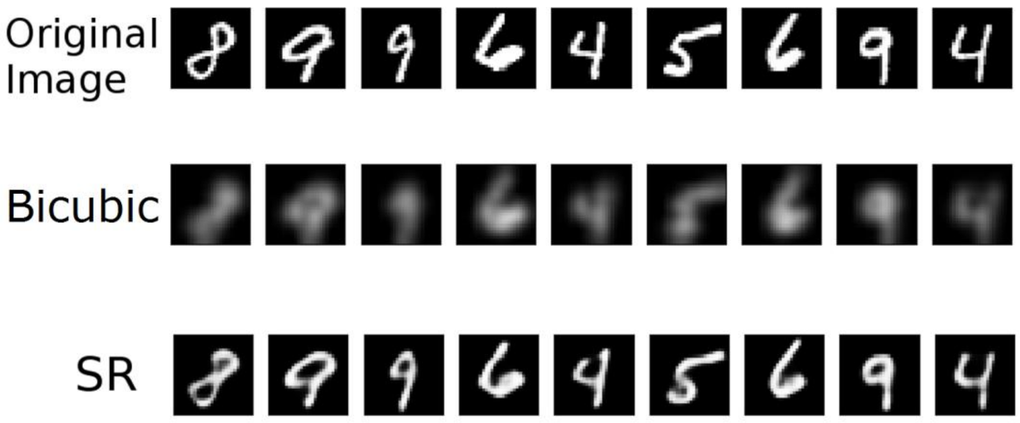


Fig. 15. Comparison on MNIST data of Bicubic (MIDDLE) and Autoencoder Model (Bottom) with Original Image (TOP).

Thus we can clearly see in fig. 15 the reconstructed images from SR Autoencoder are more prominent and distinct, hence can be easily distinguished by automatic systems.

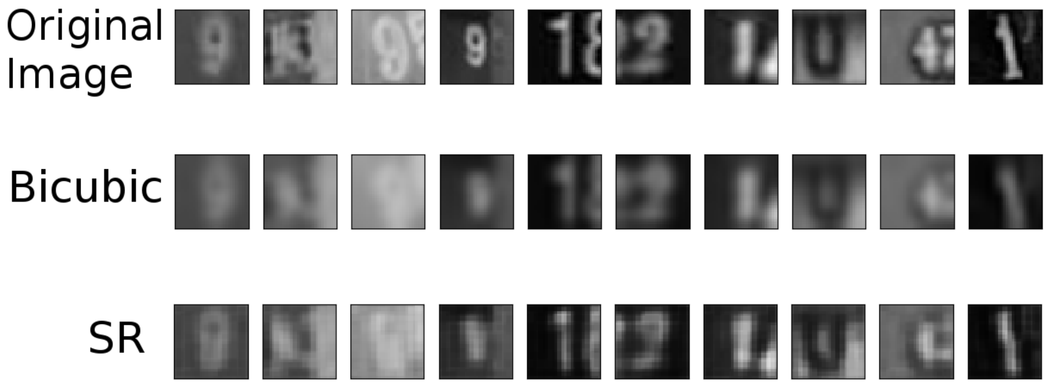


Fig. 16. Comparison on SVHN data of Bicubic (MIDDLE) and Autoencoder Model (Bottom) with Original Image (TOP).

From Fig. 16 it can be clearly seen that the Autoencoder results are clearer and distinguishable compared to Bicubic reconstructions.

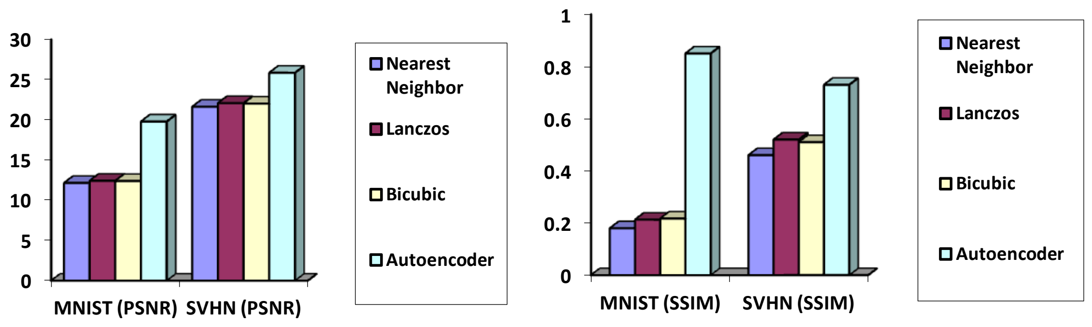


Fig. 17. PSNR and SSIM comparison of different approaches in different datasets

Fig. 17 compares the PSNR and the SSIM values obtained from the other interpolation methods and proposed autoencoder model for both the dataset. The graphical plot clearly shows improvement on image quality for using the deep neural network.

To further verify the efficiency of the system we used the test SR image for identification purpose in a convolutional neural network. The network has of 99% test accuracy on MNIST dataset. The automated system on MNIST data showed a huge lack of performance for Interpolation processes, whereas the Autoencoder SR images showed decent performance. MNIST has 10 classes of handwritten digits 0-9, accuracy measured in terms of correctly identification of the digits by the automated system.

$$Acc = \frac{N_{Correct}}{N_{Total}} \times 100\% \tag{10}$$

Formula 6 depicts the calculation of accuracy of the system. Acc is the measured value of accuracy which is calculated using $N_{Correct}$ = Total number of correctly predicted digit images, N_{Total} = Total number of test digit images.

TABLE III. IDENTIFICATION ACCURACY OF MNIST IN CONVOLUTION NETWORK

SR Method	**Accuracy (%)**
Nearest Neighbor Interpolation	52.67
Lanczos Interpolation	38.64
Bicubic Interpolation	36.30
Autoencoder	**96.44**

Thus it is clearly seen from Table III, the autoencoder output is visually more attractive and prominent. So, the detection becomes more easy and accurate for the convolutional neural network or any other automatic systems.

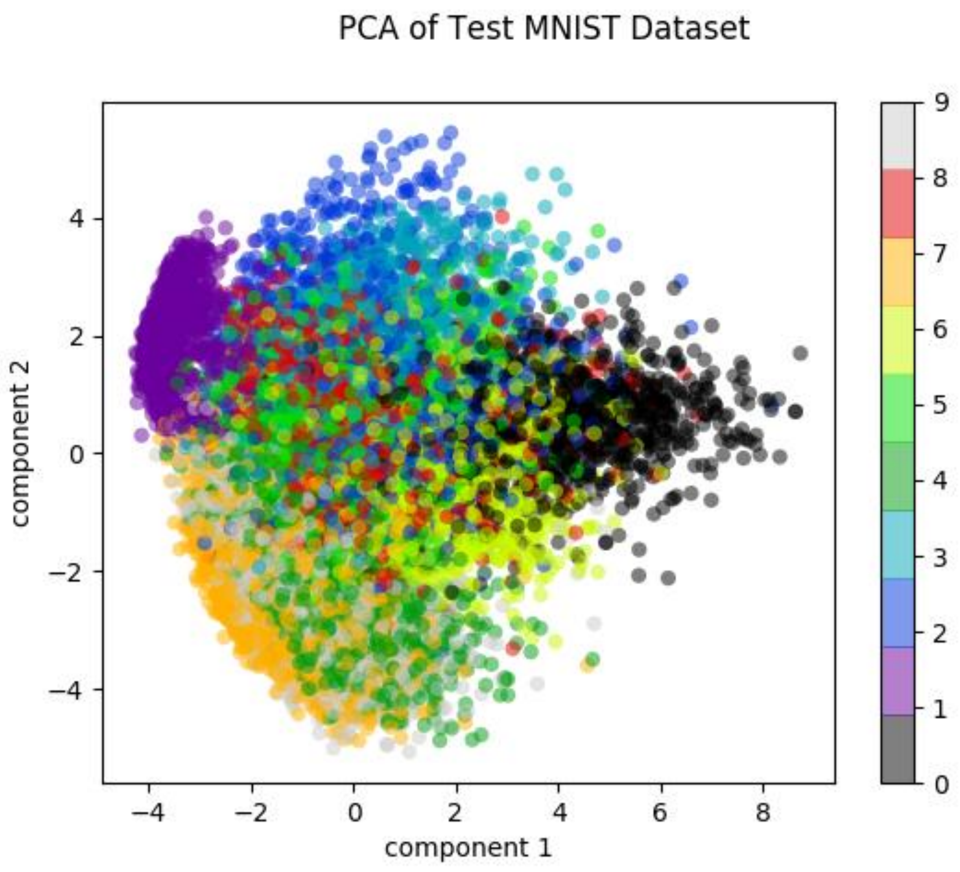


Fig. 18. PCA Analysis of SR Autoencoder output.

This section of the thesis analyzes the output of this proposed system in details. The MNIST dataset consisted of 10 classes of digits from 0-9. For proper identification of these classes the SR data must be separable from one another, i.e. they form separable clusters in the data-space. To show our SR images are clustered into different data spaces we perform a Principal Component Analysis (PCA) over the 256 (28x28) pixel values of the image with two main components. Refer to Fig. 18. The SR images that proposed model has produced can be clearly clustered into different spaces by the PCA. Fig. 6 is showing that the images can be clearly identified by the automated systems. The reconstructed images are sharper and clearer for increasing the confidence of identification.

5.2.2. Multi Character Image Super-Resolution Autoencoder Network

Results shown in Table IV and V are on a validation set of 561 images which were not previously used. The model is compared with interpolation results on because the present SR techniques can only provide a upscaling of 1.75-2x but our main objective was to scale the image by 4x. Denoising results are compared with Gaussian filter and median filters two most generalized filters used for noise removal.

TABLE IV. SUPER-RESOLUTION QUANTITATIVE ANALYSIS

METHODOLOGY	***PSNR***	***SSIM***
Bicubic Interpolation	7.4178	0.1094
Bilinear Interpolation	7.4353	0.0964
Lanczos Interpolation	7.4111	0.1096
SR Autoencoder (proposed model)	**14.7846**	**0.8139**

TABLE V. DENOISING AUTOENCODER QUANTITATIVE ANALYSIS

METHODOLOGY	***PSNR***	***SSIM***
Gaussian Filter	14.9113	0.3659
Median Filter	11.6815	0.7310
Denoising Autoencoder (proposed model)	**17.7756**	**0.8568**

The results present here clearly show here that restoration of by super-resolution is quite firmly done. For SR autoencoder PSNR increases by almost double amount from interpolation techniques and SSIM has been restored up to 81%. The denoising autoencoder performs far better than the filters compared with as measured by PSNR. The SSIM of the denoising autoencoder is also not hampered in the process of noise removal.

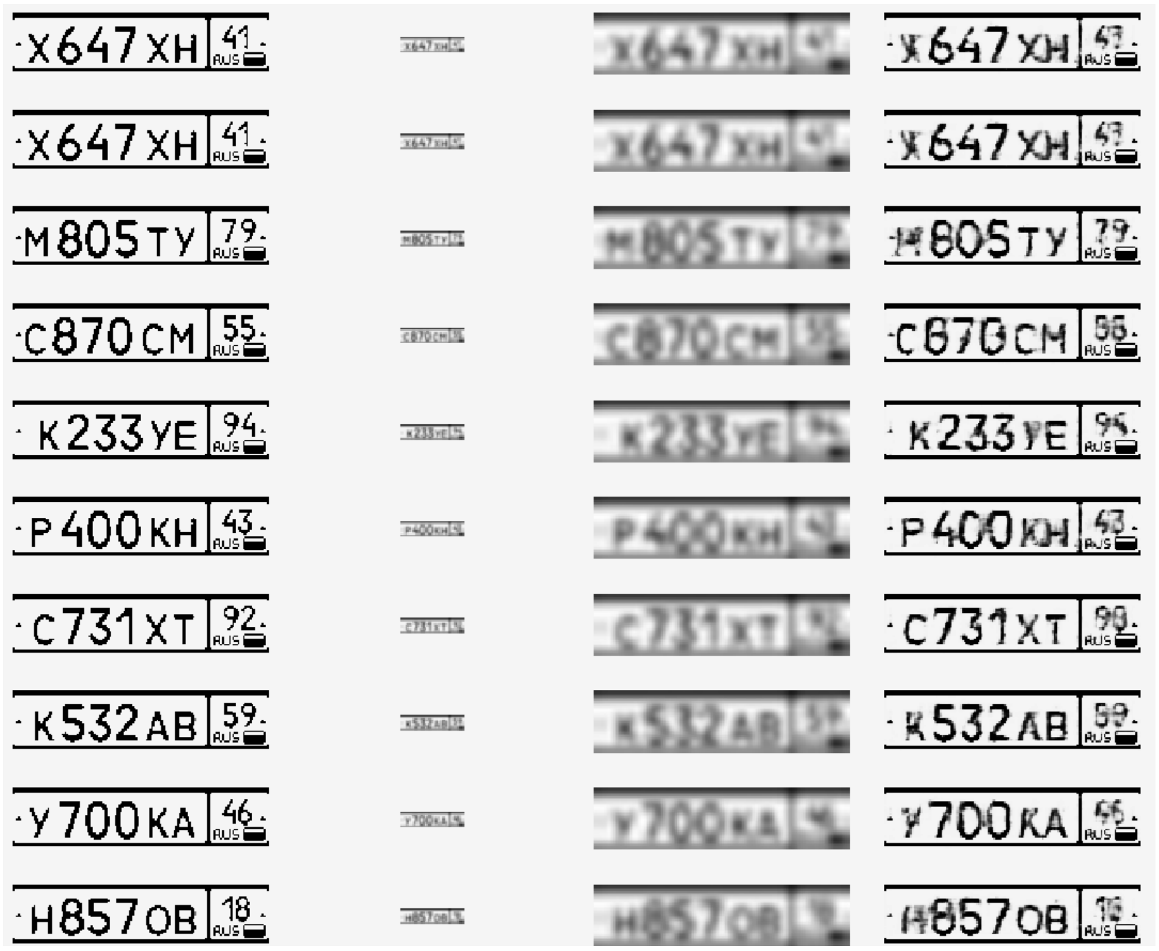


Fig. 19. Super-resolution outputs study I (i) Ground Truth (GT) (ii) 4x lowered dimension image (iii) Bicubic interpolation of lowered dimension (iv)Autoencoder result (proposed model) (by column)

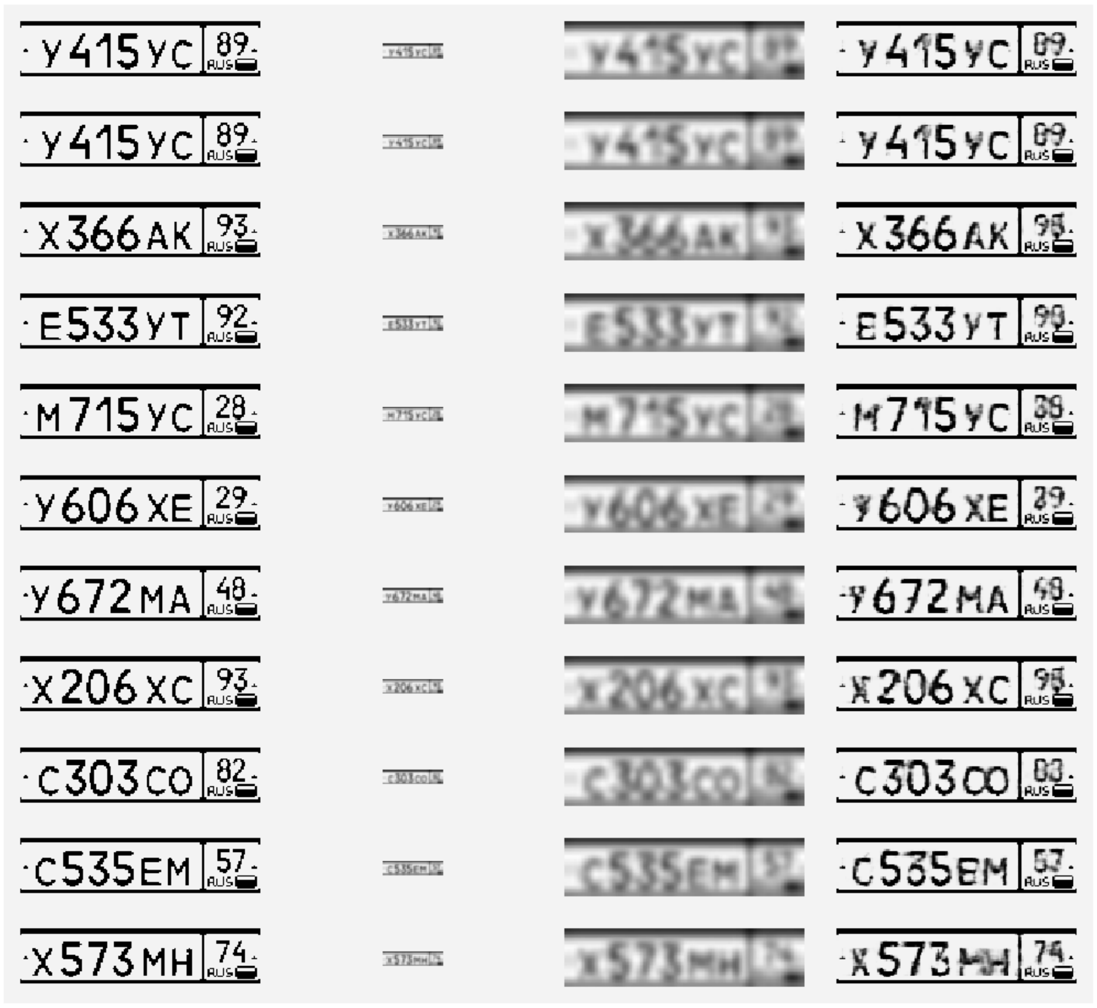


Fig. 20. Super-resolution outputs study II (i) Ground Truth (GT) (ii) 4x lowered dimension image (iii) Bicubic interpolation of lowered dimension (iv)Autoencoder result (proposed model) (by column

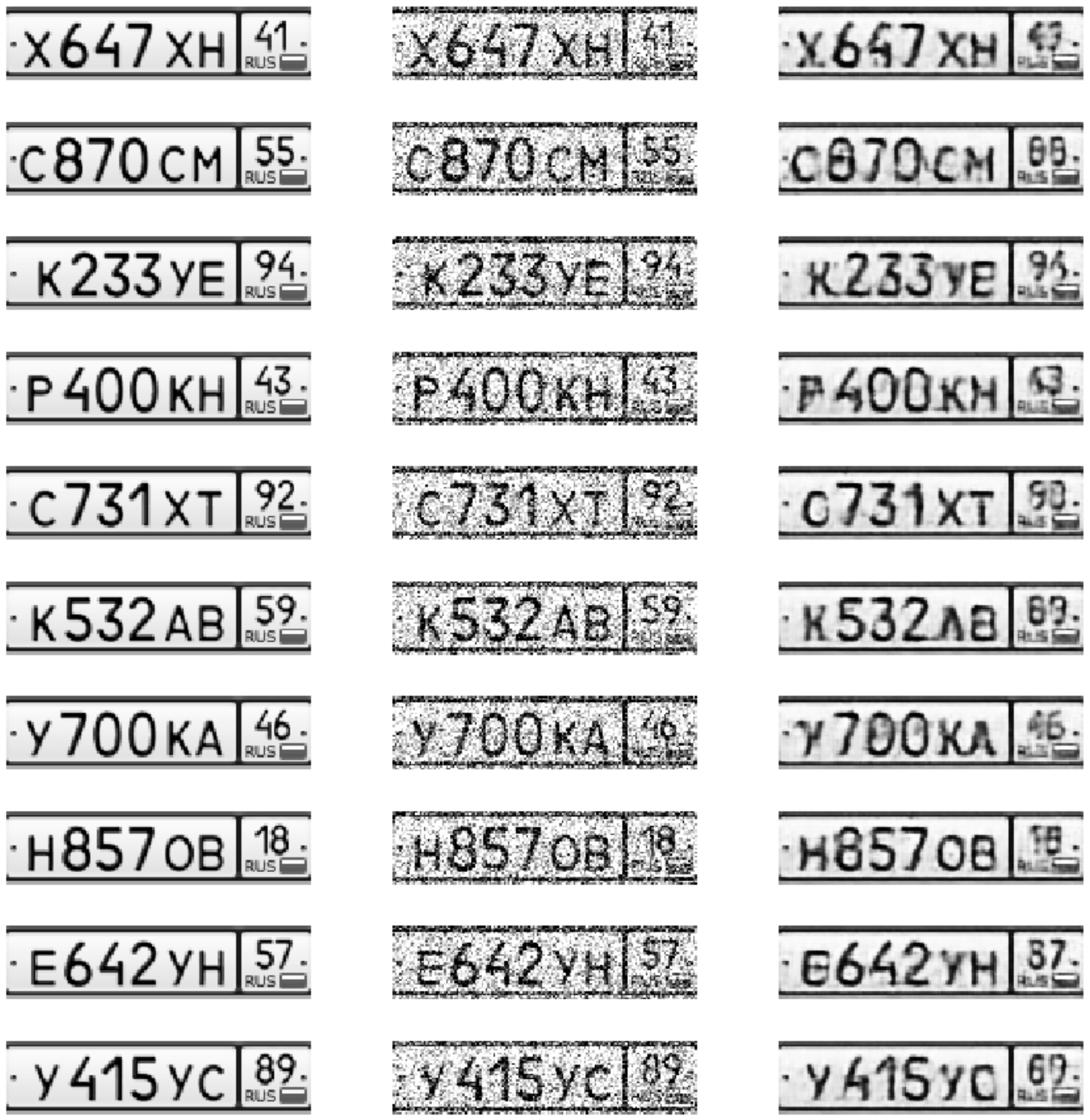


Fig. 21. Denoising output of Autoencoder study I (i) Ground Truth (GT)
(ii) noise added to GT (iii) Autoencoder result (proposed model) (by column)

Fig. 19 and 20 demonstrates some example results of our super-resolution auto encoder. Fig. 21, 22 and 23 shows results of our denoising autoencoder.

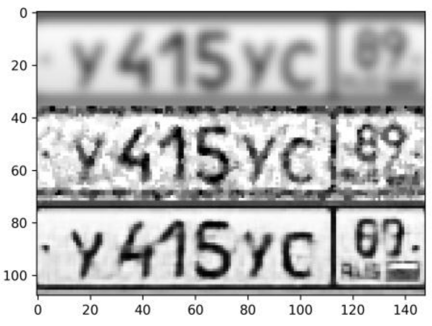


Fig. 22. Comparative study between Gaussian blur, Median blur and
Autoencoder Network (proposed model)

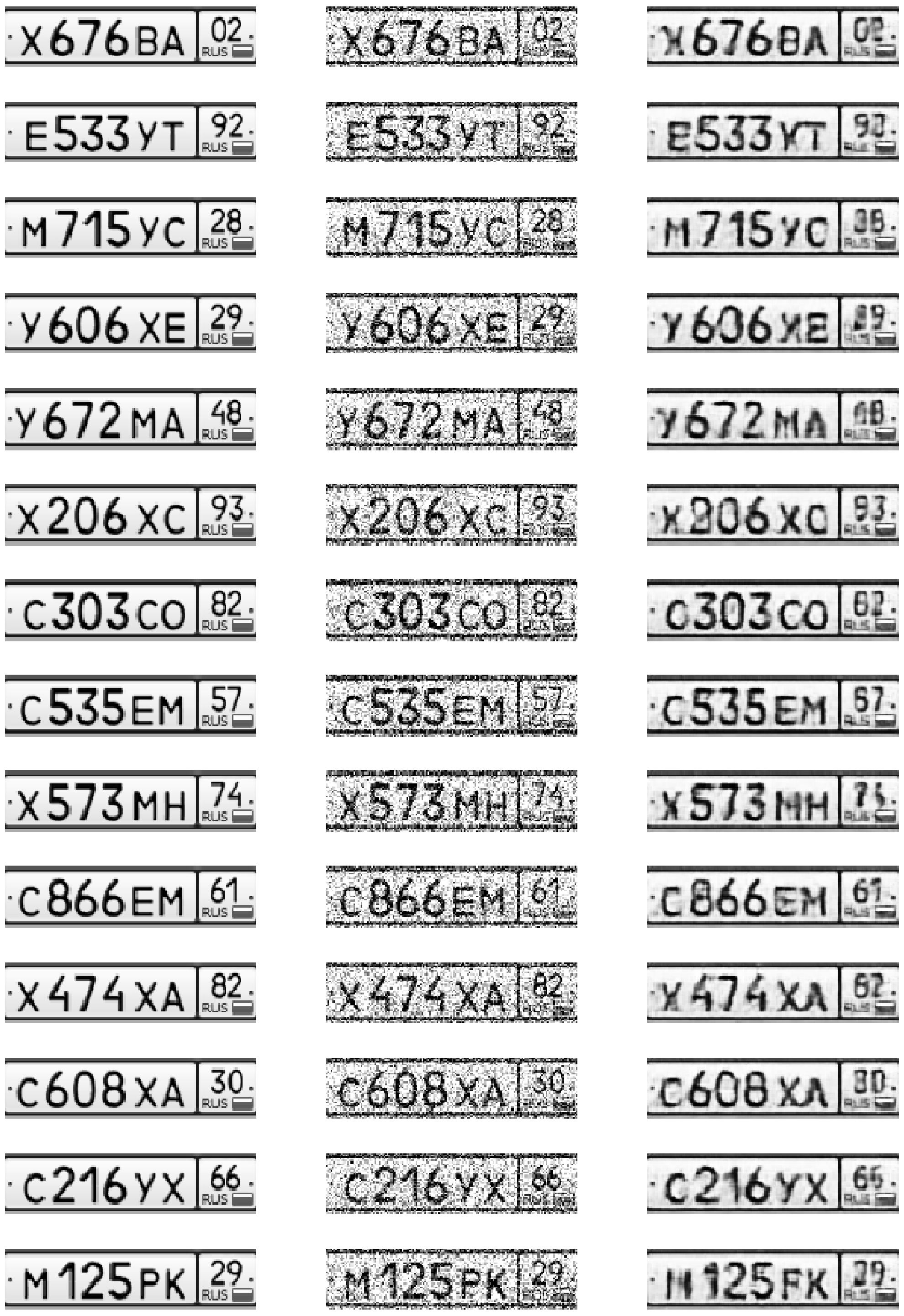


Fig. 23. Denoising output of Autoencoder study II (i) Ground Truth (GT) (ii) noise added to GT (iii) Autoencoder result (proposed model) (by column)

Fig. 13 shows comparative study of our model with commonly used filter of noise removal. From visual context we can clearly come to a conclusion that restoration of the license plate images from small dimensions and noisy images is done successfully for better understanding of the image content.

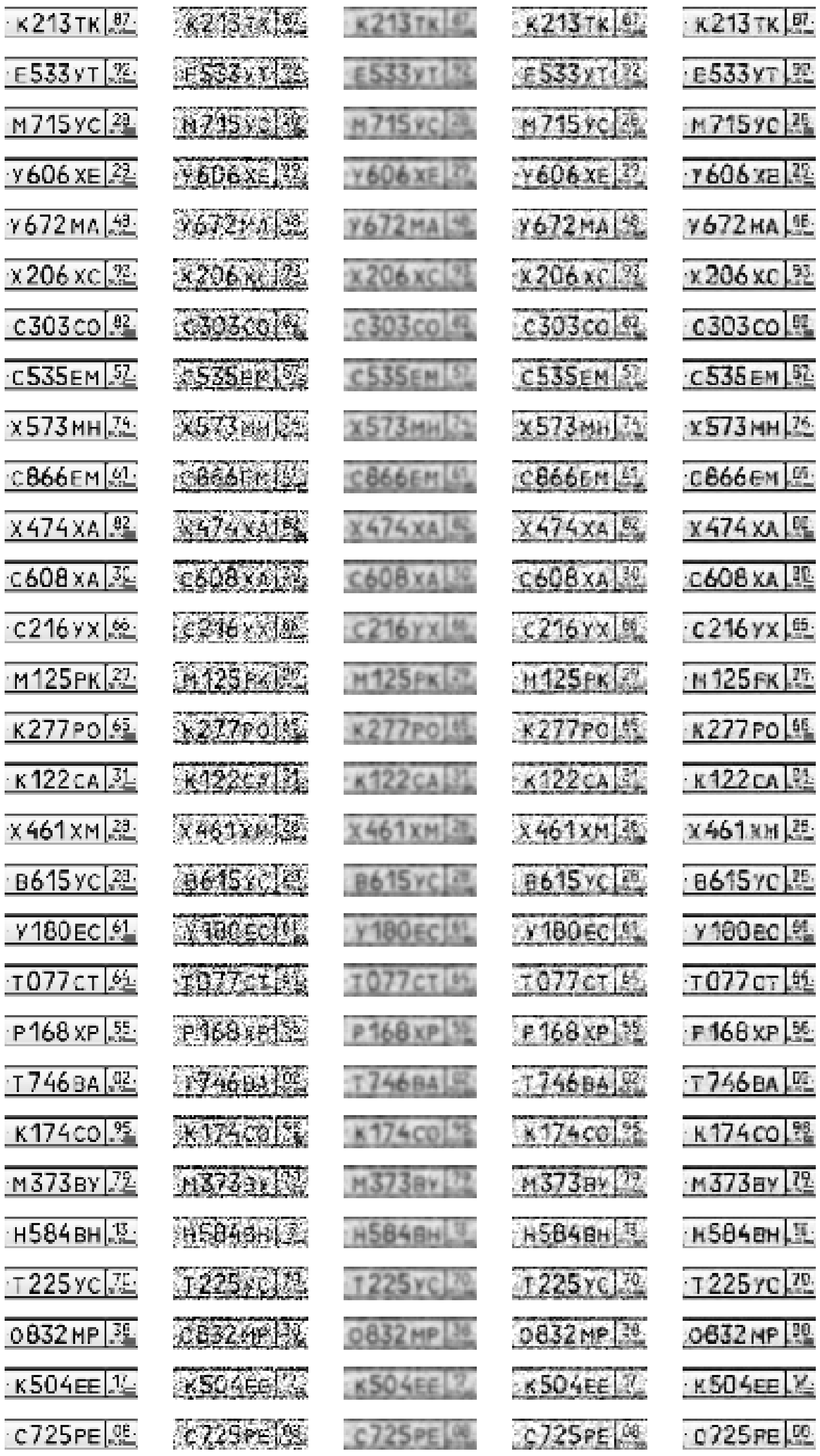


Fig. 24. Denoising Comparison I (i) original image (ii) noisy image (iii) Gaussian blur (iv) median blur (V) autoencoder network (proposed model)

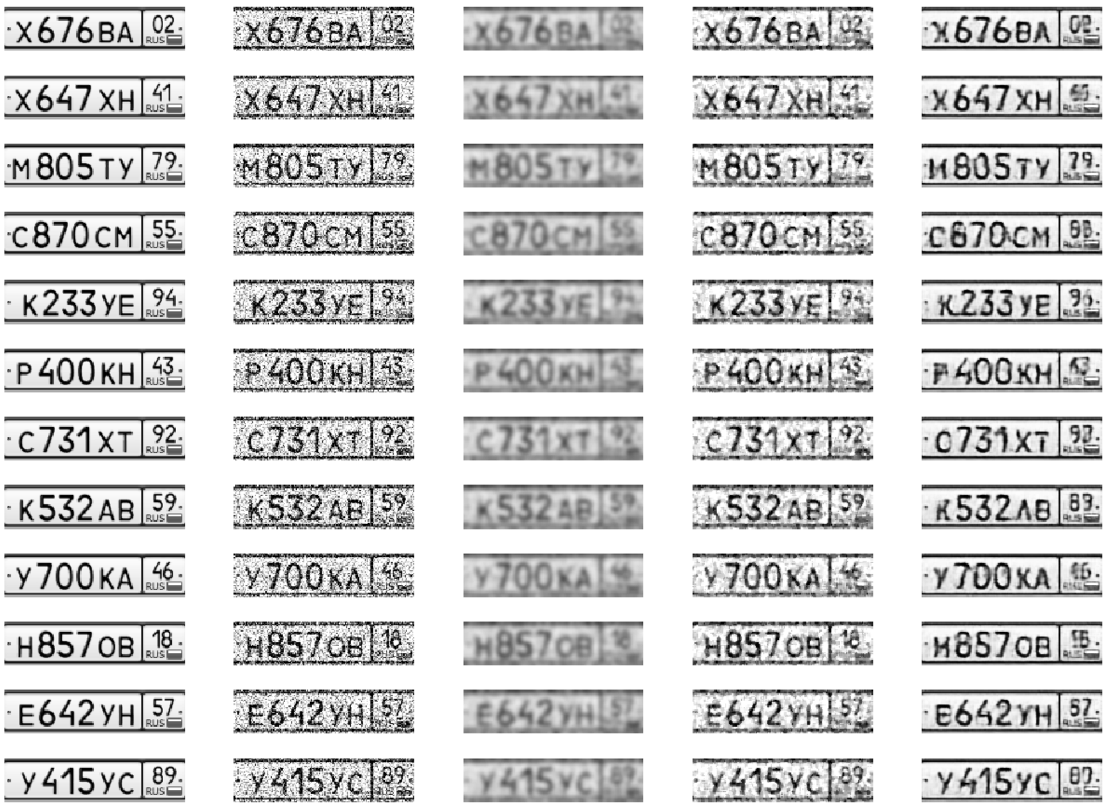


Fig. 25. Denoising Autoencoder comparison II (i) original image (ii) noisy image (iii) Gaussian blur (iv) median blur (V) autoencoder network (proposed model)

6. Conclusion

Quality enhancement of textual image is a very essential part for character recognition from natural environments. Lower dimension and noisy images are not easy to identify both from out naked eye perspective and OCR engines. A super-resolution denoising network is here by proposed that can increase the size of the region of interest by four times and clear the noise at the same time. The SSIM and PSNR results clearly suggest that the restoration of the images is done quite successfully. The structural similarity is maintained in both the networks. The model uses autoencoders which are very easy to train on any given dataset. Hence the training time for any particular dataset is also very small and hence can be transferred to different language and fronts very easily. The process is independent of any style front or language and hence it can be used over any texts. Generative adversarial training networks are not preferred due to their high training time. Hence this model proposes a language independent quality enhancement technique for textual images with a very low computational time.

Future scope of this work can be extended to motion blurs. Motion blur is very regular case surveillance footages; hence extension of this work towards motion blur would be of a great achievement. The textual image super-resolution can be also extended towards facial image super-resolution for person identification. As to conclude super-resolution would still be a challenging research domain in the upcoming years.

7. References

[1] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying Very Low-Resolution Recognition Using Deep Networks," IEEE Conference on Computer Vision and Pattern Recognition, pp. 4792-4800, 2016.

[2] Hou, H.S. and H.C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-26: pp. 508-517, 1978.

[3] W. Ruangsang, and Supavadee Aramvith, "Efficient Super-Resolution Algorithm using Overlapping Bicubic Interpolation," IEEE 6th Global Conference on Consumer Electronics (GCCE 2017), 2017.

[4] C. Zhou, Jinghong Zhou, "Single-Frame Remote Sensing Image Super-Resolution Reconstruction Algorithm Based on Two-Dimensional Wavelet", IEEE International Conference on Image, Vision and Computing, pp. 360 - 363, 2018.

[5] Xiaoning Wu, Yansheng Xing, Junping Yang and Cunlu Xu, "An Improved Algorithm with Importance Weight Value based on Super-Resolution Through Neighbor Embedding", 5th International Congress on Image and Signal Processing, pp. 294-297, 2012.

[6] Michael Elad, Arie Feuer, "Super-Resolution Reconstruction of Image Sequences", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21, NO. 9, pp. 817-834, 1999.

[7] C. E. Duchon., "Lanczos Filtering in One and Two Dimensions," Journal of Applied Meteorology, volume 18, pages 1016–1022, 1979.

[8] Christian Ledig, Lucas Theis, Ferenc Husz´ar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", IEEE Conference on Computer Vision and Pattern Recognition, pp. 105-114, 2017.

[9] Yang Yang , Ping Bi , Ying Liu, "License Plate Image Super-Resolution Based on Convolutional Neural Network," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), pp. 723-727, 2018.

[10] K. Ghosh K, S. Sarkar, K. Bhaumik, "Image Enhancement by Highorder Gaussian Derivative Filters Simulating Non-classical Receptive Fields in the Human Visual System," Proceedings Pattern Recognition and Machine Intelligence, vol. 3776, pp. 453-458, 2005.

[11] Ginu George, Rinoy Mathew Oommen, Shani Shelly, Stephie Sara Philipose, Ann Mary Varghese, "A Survey on Various Median Filtering Techniques For Removal of Impulse Noise From Digital Image", IEEE Conference on Emerging Devices and Smart Systems, pp. 235-238, 2018.

[12] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning Low-Level Vision," International Journal of Computer Vision, vol. 40, no.1, pp. 25-47, 2000.

[13] Zhenyu Wu ; Hong Hu, "Self-training-based no-reference SSIM estimation for single video frame," 2016 IEEE Region 10 Conference (TENCON), pp. 1415 - 1418, 2016.

Published Paper

IEEE Applied Signal Processing Conference 2018 (accepted)

Super-Resolution of Textual Images using Autoencoders for Text Identification

Subarno Pal
School of Education Technology
Jadavpur University, kolkata, India

Susovan Jana
School of Education Technology
Jadavpur University, Kolkata, India

Ranjan Parekh
School of Education Technology
Jadavpur University, Kolkata, India

***Abstract*— Textual image identification assumes a particular and sufficient resolution for the Region of Interest (ROI) on the image. The object or letter to be detected on the image must be of specific resolution otherwise most of the system fails to detect properly. Convolutional Neural Networks trained on normal textual images fail to detect characters, which is upscaled by interpolation methods. A deep neural network based model is proposed in this paper for upscaling and detection of textual images that show a significant increase in performance on the standard dataset. The proposed model consists of a customized autoencoder network for 4x upscaling of an image so that an automated system like the convolutional neural network can detect the upscaled images accurately. Results show a significant increase in the Structural Similarity Index (SSIM) and the Peak Signal to Noise Ratio (PSNR) measures for using the proposed model. Most importantly, the proposed approach achieves 96.44% identification accuracy for the Super-Resolution (SR) images.**

Keywords*— *Super-Resolution, Autoencoder, Digit Recognition

I. Introduction

Text recognition from images is a very popular field in computer vision. Very Low-Resolution Recognition (VLRR) [1] problem occurs when the ROI of the image is smaller than 16x16, and it becomes difficult to recognize even by human experts. Most of the efficient algorithms require minimum 16x16 resolution of ROI on the image. Images containing lower dimension of ROI gets very hard to be detected by any system trained on normal images. A very important example is House number/Licence plate number detection on video surveillance images on the zooming. Wide angled zoomed images lose resolution necessary for proper detection of texts. Handwritten/printed letters present in photographed textual images are often found to be falling under VLRR problem. Due to the lack of efficient systems during the capture, the ROI of few images remains very unclear.

Images with lower dimension have to be upscaled to a minimum higher dimension for proper detection or identification of the textual data. Various interpolations are very widely used and tested methods for image upscaling. Interpolation algorithms are independent of image content. The problem with interpolated images is that the resultant upscaled versions are hazy or unclear in places and the automated systems get confused on such cases. Interpolation methods compute the value of a pixel in an upscaled depending on the neighboring pixel of the low-resolution image. These processes are not context dependent, but the resultant images has lack of photorealistic effect.

The VLRR problem can only be solved if we can confidently upscale the image so that the automated system can be sure about the image. Image super-resolution [2, 3, 4, 5] is a very challenging task in deep learning and is mostly applied for upscaling photorealistic images.

This paper suggests a modified autoencoder network for upscaling of the textual images so that the resultant images can be easily identified by the automated identification systems. An autoencoder network usually consists of an encoder network and a decoder network coupled together. The encoder encodes an image in a different dimension and decoder network decodes it to the original structure. Autoencoder learns the distribution of the image in a particular space and is widely generative modeling of images. Generally, the input and output dimensions of the autoencoder remain the same but we modify the network in such a way that the network performs upscaling of the image. The experimentation was performed on down-scaled Modified National Institute of Standards and Technology (MNIST) handwritten digits dataset and SVHN (Street View House Numbers) dataset. The PSNR [6] and SSIM [7] comparisons proved the efficiency of the system over normal interpolation upscaling methods.

This paper mainly focuses on textual image super-resolution for proper identification of the same. Section II depicts some notable and related works. The proposed methodology is illustrated in section III. Experimental details and results obtained along with the comparisons are thoroughly discussed in section IV. Section V goes deeper into Analysis of the work and state why this model is reliable in this context. Finally, section VI draws the conclusion of this work with the future scope and followed by the references that are being used.

II. Related Works

Image Super-resolution is a highly studied research field in computer vision. This field has got some huge recent updates following the massive advancement in deep learning. Various generative networks are being proposed by many researchers having state of the art efficiency in different situations. Some of the very notable works discussed in this section.

Single Image Super Resolution (SISR) [8] has been dealt with prediction based methods for quite a long time. Filtering methods mainly Interpolations were widely used for many years. Bicubic interpolation [9, 10] was among one of the most popular methods in practice. It takes 16 pixels (for min. 4x4 images) for calculation of interpolation point on the image. It's smoother than Liner Interpolation (4 pixels consideration) or Nearest Neighbor Interpolation [11].

Lanczos Method [12] of interpolation was based upon 3-lobed Lanczos window for calculation of interpolation point. This algorithm used 36 neighboring points for calculation of interpolation value.

Methods that used deep learning [13] in this field brought an immense change in the generation of photorealistic images. Training is based on images of Low Resolutions (LR) for which Higher Resolution (HR) is known. Different generative methods are introduced such as variations of regressions [14], convolutional neural network [15], generative adversarial networks (GAN) [16], encoder-decoder networks etc. These methods are domain specific and context dependent. A landmark work in this domain by Twitter they proposed a perceptual loss function which was a combination of adversarial loss and content loss. They showed a remarkable gain in bringing back the texture from the heavily down-sampled image using their deep residual network and increase in Mean Opinion Score (MOS).

Methods involving deep learning based algorithms require huge amount data, training time costly GPUs, so they are very hard to adapt to a new dataset. Linear and Nearest Neighbor interpolations are very fast, but Bicubic and Lanczos are more effective and don't need any training. The proposed method is consisting of a modified variation of autoencoder which requires very less training time on particular datasets compared to GANs and performs better than interpolation methods in most cases.

III. Proposed method

This section describes the detailed view of the proposed method. The proposed method has two major part. The first part is Super-Resolution Autoencoder for upscaling the low-resolution images and the second part is a convolutional neural network for the identification of an object from an image. Fig. 1 shows the process flow of the proposed system.

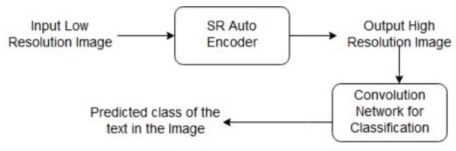

Input Low Resolution Image
SR Auto Encoder
Output High Resolution Image
Convolution Network for Classification
Predicted class of the text in the Image

Fig. 1. Flowchart for super-resolution and identification process.

A. Super-Resolution Autoencoder

A modified convolutional autoencoder model is proposed for image super-resolution favorable for text detection. The autoencoder network consists of mainly convolution blocks, which were fitted one after another. It takes input a low-resolution image and gives a higher resolution version of it as output. A low-resolution image is taken and converted into grayscale. This single-channel grayscale image is the input to the system.

Formally, given a set of $Y = \lfloor y_1, y_{2,}, ..., y_n \rfloor$, where $y_i \in R^d$, autoencoder is trained to minimize the reconstruction error to properly modeled data. The loss function 'l' is defined in Eqn. (1).

$$l = \sum_i \| \ y_i - \hat{y}_i \ \|^2 \tag{1}$$

y_i is the actual HR image and $\hat{y}_i$ is the reconstructed SR form of the LR image. The hidden layers of the network are mainly involved in encoding and decoding of the data.

$$h_i = f(Wy_i + b) \tag{2}$$

$$\hat{y}_i = f(W'h_i + b') \tag{3}$$

h_i denotes the encoded compact representation of data in the network. Refer to Eqn. (2). $\hat{y}_i$ is the reconstructed representation of the image. Refer to Eqn. (3). Activation functions are at the core of the network in every layer and it is denoted by *f(z)*.

Different activation functions are used at different stages of the network viz. 'sigmoid' and 'ReLU' mentioned in Eqn. (4) and Eqn. (5) respectively.

$$f(z) = \frac{1}{1+e^{-z}} \tag{4}$$

$$f(z) = \max(0, z) \tag{5}$$

Single layer encoder and decoder is not very efficient in learning complex varied datasets representations. To address this problem, a deep layered architecture is constructed for increasing the nonlinearity of the system.

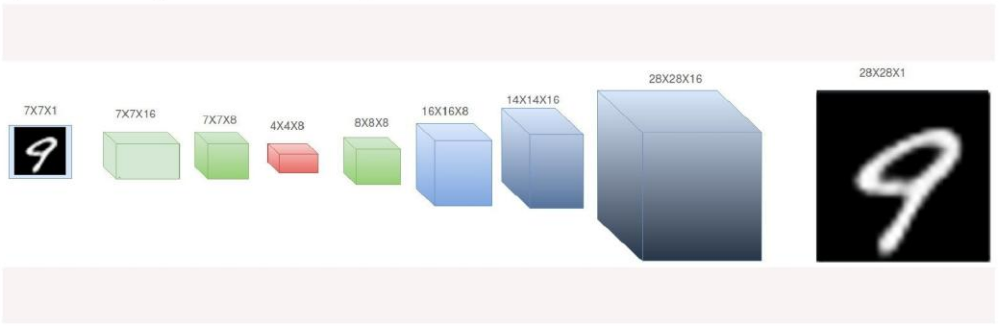


Fig. 2. The detailed architecture of the autoencoder network.

A greyscale version of the image is sent into the input layer of the network. The single channeled image forms the input to the network. Followed by the input layer the autoencoder consist of two more layers for encoding followed by four decoding layers. The encoder part of the network encodes the input to low dimensions. The encoder network consists of two convolution blocks 3x3 kernel size which results to two encoded layers. The convolution layers are followed by an upsampling layer which forms the compressed data or latent representation vector of the distribution. The encoded data is now decoded using a combination of 3x3 kernel size convolution layer followed by an upsampling layer with the 2x2 upsampling factor. This combination is used thrice in the network which is finally followed by a 3x3 convolution layer. The activation function of all the inner layers is 'ReLU', only the final layer has 'sigmoid'. Fig. 2 visualizes the detailed structure of the encoder network. The output of the network is a single channel higher resolution version of the input image which is used for text detection in the next subsection.

B. Convolution Neural Network

The text detection from the image is done using a simple convolutional neural network. The detection convolutional network consists of two convolution layers followed by a max-pooling layer. The max-pooling layer is connected to two successive dense layers for classification. 'ReLu' activation function described is used in the layers to add nonlinearity in the inner layers. Only the outer layer has a softmax function. Refer to Eqn. (6). Here, z is a vector of the inputs to the output layer (if you have 10 output units, then there are 10 elements in z) and again, j indexes the output units as j = 1, 2, ..., K.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{6}$$

Any type of convolution network can be used for identification purpose. Here, this configuration of the convolution neural network is used to validate the results of the autoencoder.

IV. Experimentation and Results

In this paper, the main focus is given on textual image super-resolution for proper text identification. The experimentation was done on two datasets, which were widely used for text recognition purpose (a) MNIST [17]-consisting of handwritten images and (b) SVHN [18] - street view house number dataset.

The SVHN [18] dataset contains RGB images of highly varied texture and color. So the dataset was preprocessed to bring uniformity in the distribution. Firstly, the RGB images were converted to grayscale images. Next, the dataset consisted of two main types of images, i.e. one with 'lighter background' and other with 'darker background'. Otsu's binarization was applied to the grayscale image. Left corner of the binary image was analyzed to find which of them has a lighter background.

The images with lighter backgrounds were inverted and further converted into images with 'darker background'.

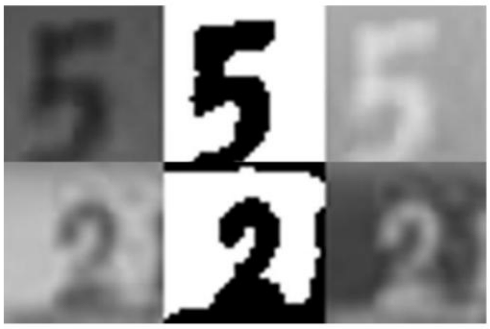


Fig. 3. Preprocessing of image data in SVHN dataset (Right-most column showing final output, i.e. all images with darker background).

Fig 3 shows how the preprocessing is done on the images. The preprocessing shows that the images with lighter backgrounds are identified and inverted. Inversion of those images transforms them into images with darker background as shown in the 3rd column.

A. Training the Model

At first, the images are converted into a low-resolution image by 4x by the pyramid-down algorithm. Then, the lower resolution image is reconstructed into 4x higher resolution by our model. The content loss (*l*) of the image is calculated by measuring the reconstruction activation loss. The network is trained with back-propagation throughout the system. The network is trained approximately 100 epochs on the training set to obtain the output.

B. Testing the Model

The test data was down-sampled by 4x using pyramid down algorithm and it was used as a test set for this proposed model. The SR image obtained from our network was compared using PSNR and SSIM values with the original High Resolution of the image.

The PSNR value computes the peak signal-to-noise ratio, in decibels, between two images, is used as a quality measurement between the original and an SR image. Higher PSNR denotes the better quality of the SR image. SSIM is the Structural Similarity Index Measure (SSIM), quantifies the quality degradation of the image in the reconstruction.

The accuracy of the system was evaluated using PSNR and SSIM methods. Finally, these images are used for identification and a high increment in the accuracy of the system was observed for our model.

TABLE I. Comparison of Various SR Methods on MNIST [17] Dataset

SR Method	PSNR	SSIM
Nearest Neighbor Interpolation	12.15	0.18
Lanczos Interpolation	12.42	0.21
Bicubic Interpolation	12.40	0.21
Proposed model (Autoencoder)	**19.79**	**0.85**

TABLE II. COMPARISON OF VARIOUS SR METHODS ON SVHM [18] DATASET

SR Method	PSNR	SSIM
Nearest Neighbor Interpolation	21.63	0.46
Lanczos Interpolation	22.08	0.52
Bicubic Interpolation	22.02	0.51
Proposed model (Autoencoder)	**25.85**	**0.73**

Table I. depicts a comparison of different SR methods using PSNR and SSIM on MNIST [17] dataset. Table II. presents a comparison of different SR methods using PSNR and SSIM on SVHM [18] dataset. Thus it can be said from table I and II that the PSNR and SSIM value remarkably increases when this Autoencoder network was used for both the datasets. The test results obtained are calculated after training the Autoencoder by 100 epochs. The increment in training time might increase the accuracy of the system even further.

Thus we can clearly see in Fig. 4. that the reconstructed images from SR Autoencoder are more prominent and distinct for MNIST dataset, hence it can be easily distinguished by automatic classification systems. Fig. 5 represents the same on SVHM dataset.

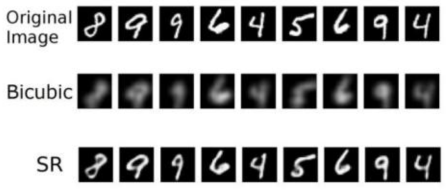


Fig. 4. Comparison of Bicubic (Middle) and Autoencoder Model (Bottom) with Original Image (Top) on MNIST [17] dataset.

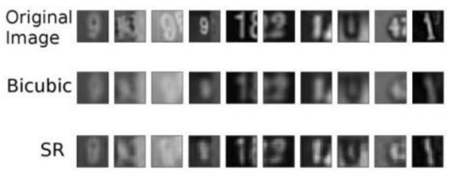


Fig. 5. Comparison of Bicubic (Middle) and Autoencoder Model (Bottom) with Original Image (Top) on SVHN [18] dataset.

To further verify the efficiency of the system we used the test SR image for identification purpose in a convolutional neural network. The network had **99%** test accuracy on MNIST dataset. The automated system on MNIST data showed a huge lack of performance for Interpolation processes, whereas the Autoencoder SR images showed decent performance. MNIST has 10 classes of handwritten digits 0-9, accuracy measured in terms of correctly identification of the digits by the automated system. Eqn. (7) depicts the calculation of identification accuracy of the system. Accuracy (A) percentage is calculated from $N_{Correct}$ = Total number of correctly predicted digit images, N_{Total} = Total number of test digit images.

$$A = \frac{N_{Correct}}{N_{Total}} \times 100\% \qquad (7)$$

Table III depicts that the autoencoder output is visually more attractive and prominent. So, the detection becomes more easy and accurate for the convolutional neural network or any other automatic systems.

TABLE III. IDENTIFICATION ACCURACY WITH CONVOLUTION NETWORK ON MNIST DATASET [17]

SR Method	Accuracy (%)
Nearest Neighbor Interpolation	52.67
Lanczos Interpolation	38.64
Bicubic Interpolation	36.30
Autoencoder	**96.44**

V. ANALYSIS

This section of the paper analyzes the output of this proposed system in details. The MNIST dataset consisted of 10 classes of digits from 0-9. For proper identification of these classes the SR data must be separable from one another, i.e. they form separable clusters in the data-space.

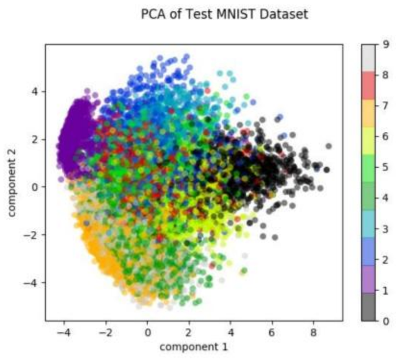


Fig. 6. PCA of SR images produced by Autoencoder on MNIST [17] Dataset.

To show our SR images are clustered into different data spaces we perform a Principal Component Analysis (PCA) over the 256 (28x28) pixel values of the image with two main components. Refer to Fig. 6. The SR images that proposed model has produced can be clearly clustered into different spaces by the PCA. Fig. 6 is showing that the images can be clearly identified by the automated systems. The reconstructed images are sharper and clearer for increasing the confidence of identification. Fig 7 and 8 compares the PSNR and the SSIM values obtained from the other interpolation methods and proposed autoencoder model for both the dataset. The graphical plot clearly shows improvement on image quality for using the deep neural network.

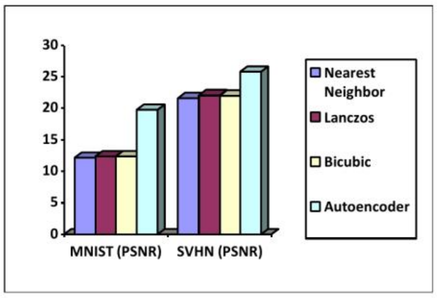


Fig. 7. PSNR comparison of different approaches in different datasets.

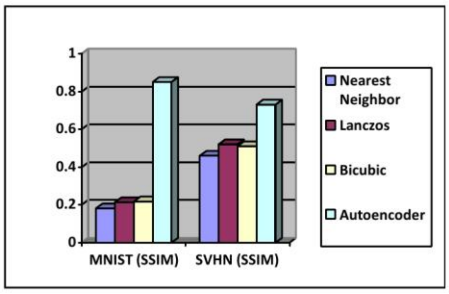


Fig. 8. SSIM comparison of different approaches in different datasets.

VI. Conclusion

This paper presents a novel approach for detection of low-resolution text by 4x super-resolution by using a modified autoencoder network. The model upscales a 7x7 image to 28x28 with the minimal runtime. The Autoencoder network proposed here, takes the very little amount of training time and resource for surpassing conventional interpolation methods, so it's preferred over GANs. This system is independent of any language, style, and font of the letters. It can also handle different lighting and shading conditions. The performance is better than the other interpolation methods. The experimentation was done on two mostly used popular datasets to validate the accuracy of the proposed method. This model can be easily transferred over different languages by training on respective datasets. Future work will include reliability of this network on increasing the number of languages and letters, hence increasing the number of predictive classes for the model. This model can be also adapted on basic autoencoder network tasks such as de-noising noisy low-resolution images and upscaling to higher resolution.

References

[1] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, "Studying Very Low-Resolution Recognition Using Deep Networks," IEEE Conference on Computer Vision and Pattern Recognition, pp. 4792-4800, 2016.

[2] M. O. Camponez, E. O. T. Salles, and M. Sarcinelli-Filho, "Superresolution image reconstruction using nonparametric Bayesian INLA approximation," IEEE Trans. Image Process., vol. 21, no. 8, pp. 3491–3501, Aug. 2012.

[3] Y. Tang and Y. Yuan, "Learning from errors in super-resolution," IEEE Trans. Cybern., vol. 44, no. 11, pp. 2143–2154, Nov. 2014.

[4] B. Langmann, W. Wehihs, O. Loffeld, and K. Hartmann, "Development and investigation of a long-range time-of-flight and color imaging system," IEEE Trans. Cybern., vol. 44, no. 8, pp. 1372–1382, Aug. 2014.

[5] X. Lu and X. Li, "Multiresolution imaging," IEEE Trans. Cybern., vol. 44, no. 1, pp. 149–160, Jan. 2014.

[6] C.-Y. Yang, C. Ma, and M.-H. Yang., "Single-image super-resolution: A benchmark," European Conference on Computer Vision (ECCV), pp. 372–386. Springer, 2014.

[7] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli., "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, 13(4): pp. 600–612, 2004.

[8] C. E. Duchon., "Lanczos Filtering in One and Two Dimensions," Journal of Applied Meteorology, volume 18, pages 1016–1022, 1979.

[9] W. Ruangsang, and Supavadee Aramvith, "Efficient Super-Resolution Algorithm using Overlapping Bicubic Interpolation," IEEE 6th Global Conference on Consumer Electronics (GCCE 2017), 2017.

[10] Hou, H.S. and H.C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-26: pp. 508-517. , 1978.

[11] H. Chang, D.Y Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition; Washington, pp. I-I, 2004.

[12] C. E. Duchon., "Lanczos Filtering in One and Two Dimensions," Journal of Applied Meteorology, volume 18, pages 1016–1022, 1979.

[13] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang., "Deep networks for image super-resolution with sparse prior," IEEE International Conference on Computer Vision (ICCV), pp. 370–378, 2015.

[14] H. Zhang, J. Yang, Y. Zhang, and T. S. Huang, "Image and video restorations via nonlocal kernel regression," IEEE Trans. Cybern., vol. 43, no. 3, pp. 1035–1046, Jun. 2013.

[15] C. Dong C, C.C. Loy, K. He, X. Tang, "Learning a deep convolutional network for image super-resolution," Proceedings of 13th European Conference on Computer Vision; Zurich, Switzerland, pp.184-199, 2014.

[16] C. Ledig, L.Theis, F. Husz´ar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," IEEE Conference on Computer Vision and Pattern Recognition, pp. 105-114, 2017.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner., "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[18] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

CODE

5/25/2019 super_resolution_auto_encoder.ipynb - Colaboratory

```
from google.colab import drive
drive.mount('/mnt/')
```

```
Drive already mounted at /mnt/; to attempt to forcibly remount, call drive.mount('
```

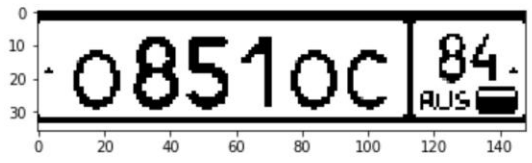
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
```

```
path = '/mnt/My Drive/Number_Plate/'
valid = np.load(path+'data/test.npy')
train_data = np.load(path+'data/train.npy')
```

```
train_data = 255-train_data
valid = 255- valid
train_data[train_data <127] = 0
train_data[train_data >127] = 255
valid[valid<127] = 0
valid[valid >127] = 255
x_train, x_test = np.split(train_data,[int(3*len(train_data)/4)])
```

```
plt.imshow(255-valid[90].reshape(36,148) , cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f2e6001f128>
```



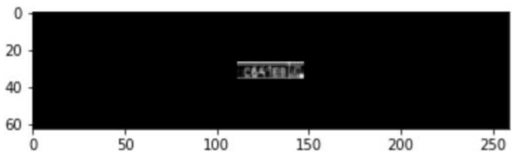
```
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
valid = valid.astype('float32') / 255.
```

Plotting the original image.

```
x_train_low = np.zeros((x_train.shape[0], x_train.shape[1]//4, x_train.shape[2]//4))

for i in range(x_train.shape[0]-1):
  x_train_low[i] = cv.pyrDown(cv.pyrDown(x_train[i]))

x_test_low = np.zeros((x_test.shape[0], x_test.shape[1]//4, x_test.shape[2]//4))
for i in range(x_test.shape[0]-1):
  x_test_low[i] = cv.pyrDown(cv.pyrDown(x_test[i]))

valid_low = np.zeros((valid.shape[0], valid.shape[1]//4, valid.shape[2]//4))
for i in range(valid.shape[0]-1):
  valid_low[i] = cv.pyrDown(cv.pyrDown(valid[i]))

x_train_low = np.reshape(x_train_low, (x_train_low.shape[0],x_train_low.shape[1], x_trai
print(x_train_low.shape)
x_test_low = np.reshape(x_test_low, (x_test_low.shape[0],x_test_low.shape[1], x_test_low
print(x_test_low.shape)
valid_low = np.reshape(valid_low, (valid_low.shape[0],valid_low.shape[1], valid_low.shap
print(valid_low.shape)
plt.imshow(cv.copyMakeBorder(valid_low[110].reshape(9,37),3*x_train.shape[1]//4,3*x_trai
#plt.imshow(x_train_low[10].reshape(36,148))
```

https://colab.research.google.com/drive/1sCRLhcawDyFNOjcAkQoYgDr4HvulUMM8#scrollTo=SJ3CDTWNUCTi&printMode=true 1/7

```
(8115, 9, 37, 1)
(2706, 9, 37, 1)
(561, 9, 37, 1)
<matplotlib.image.AxesImage at 0x7f2e4e7f69b0>
```



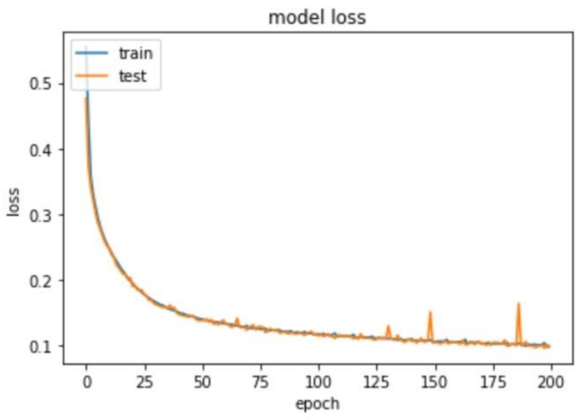
```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
from keras import backend as K

input_img = Input(shape=(9, 37, 1))  # adapt this if using `channels_first` image data f

x = Conv2D(16, (1, 1), activation='relu', padding='same')(input_img)
x = MaxPooling2D((1, 1), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(16, (3, 3), activation='relu')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')


print(autoencoder.summary())
```

```
    _________________________________________________________________
    Layer (type)                 Output Shape              Param #
    =================================================================
    input_2 (InputLayer)         (None, 9, 37, 1)          0
```

```
'''history = autoencoder.fit(x_train_low, x_train,
                epochs=200,
                batch_size=128,
                shuffle=True,
                validation_data=(x_test_low, x_test))
autoencoder.save_weights(path+'/weights/weightsfile_super_res.h5')'''
```

```
print(history.history.keys())
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig(path+'img/'+'super_res_training.png', dpi=300)
plt.show()
```

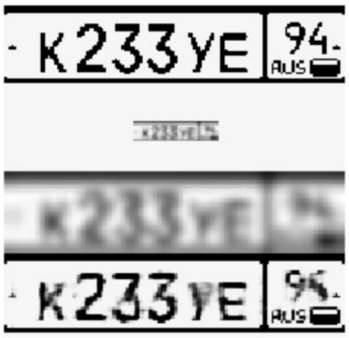
```
dict_keys(['val_loss', 'loss'])
```



Plotting the training loss.

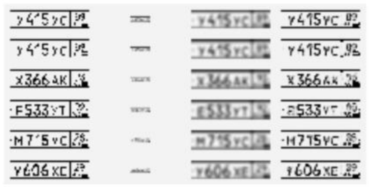
```
autoencoder.load_weights(path+'/weights/weightsfile_super_res.h5')
decoded_imgs = autoencoder.predict(valid_low)

m=3
a = valid[m].reshape(36,148)


h_pad = (3*x_train.shape[1]//8)
w_pad = (3*x_train.shape[2]//8)
b = cv.copyMakeBorder(valid_low[m].reshape(9,37),3*x_train.shape[1]//8+1,3*x_train.shape
print(b.shape)
c = cv.resize(valid_low[m],None,fx=4, fy=4, interpolation = cv.INTER_CUBIC).reshape(36,1

d = decoded_imgs[m].reshape(36, 148)

img1 = np.vstack((a,b))
img2 = np.vstack((c,d))
img = np.vstack((img1,img2))
plt.axis('off')
plt.imshow(255-img, cmap='gray')
plt.savefig(path+'img/'+'super_res_result.png', dpi=300)
```

```
(36, 148)
```



```
autoencoder.load_weights(path+'/weights/weightsfile_super_res.h5')
decoded_imgs = autoencoder.predict(valid_low)

m=10
a = valid[m].reshape(36,148)


h_pad = (3*x_train.shape[1]//8)
w_pad = (3*x_train.shape[2]//8)
b = cv.copyMakeBorder(valid_low[m].reshape(9,37),3*x_train.shape[1]//8+1,3*x_train.shape
print(b.shape)
c = cv.resize(valid_low[m],None,fx=4, fy=4, interpolation = cv.INTER_CUBIC).reshape(36,1

d = decoded_imgs[m].reshape(36, 148)

img1 = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.c
img2 = np.hstack((cv.copyMakeBorder(c,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.c
img = np.hstack((img1,img2))

for m in range(m,m+10):
  a = valid[m].reshape(36,148)


  h_pad = (3*x_train.shape[1]//8)
  w_pad = (3*x_train.shape[2]//8)
  b = cv.copyMakeBorder(valid_low[m].reshape(9,37),3*x_train.shape[1]//8+1,3*x_train.sha

  c = cv.resize(valid_low[m],None,fx=4, fy=4, interpolation = cv.INTER_CUBIC).reshape(36

  d = decoded_imgs[m].reshape(36, 148)

  img1 = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv
  img2 = np.hstack((cv.copyMakeBorder(c,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv
  img3 = np.hstack((img1,img2))
  img = np.vstack((img,img3))

plt.axis('off')
plt.imshow(255-img, cmap='gray')

plt.savefig(path+'img/'+'super_res_comb_result2.png', dpi=300)
```

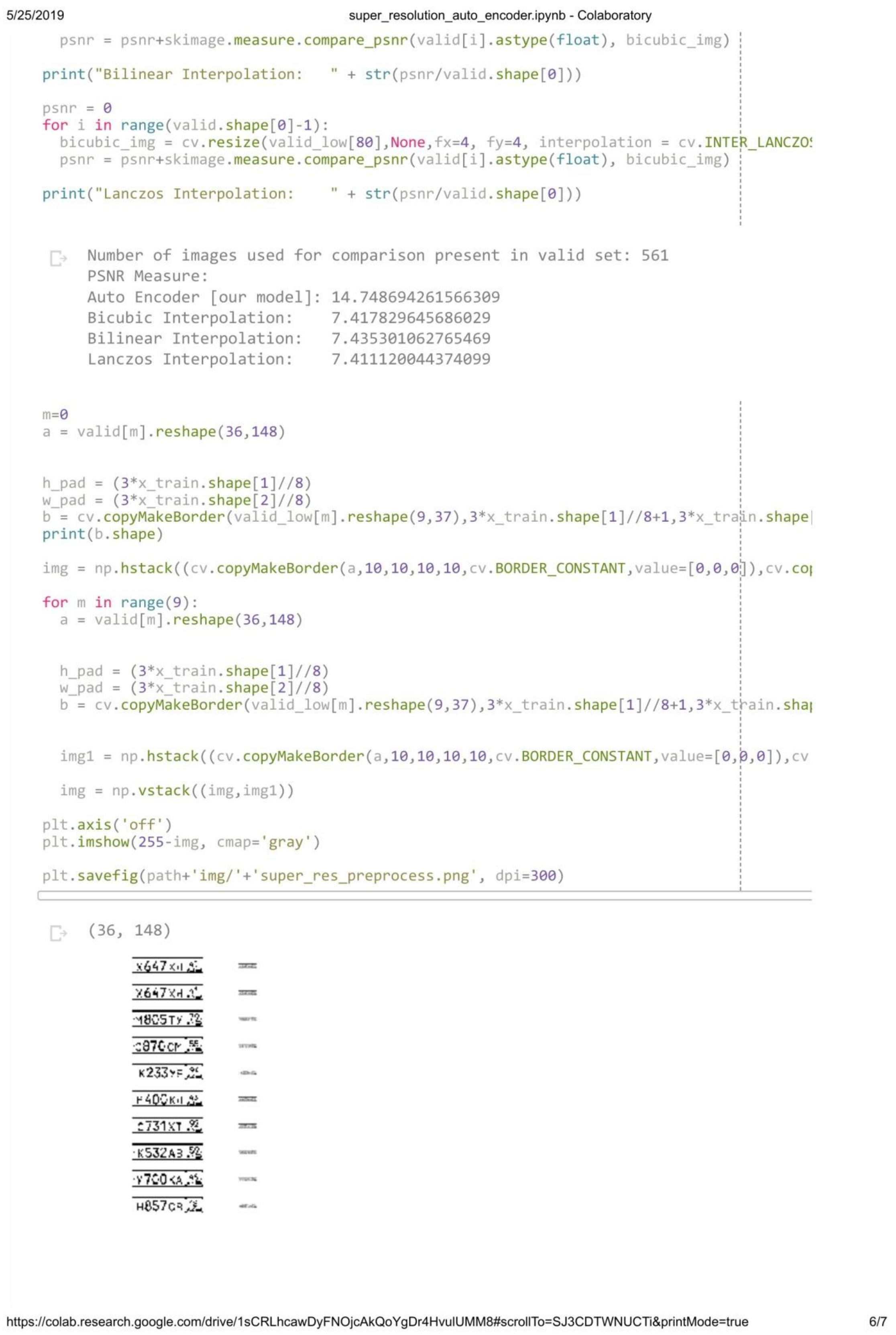
```
(36, 148)
```



```
import skimage.measure

print("Number of images used for comparison present in valid set: "+str(valid.shape[0]))

print("SSIM Measure:")
ssim = 0
for i in range(valid.shape[0]-1):
  ssim = ssim+skimage.measure.compare_ssim(valid[i], decoded_imgs[i], multichannel=True)

print("Auto Encoder [our model]: " + str(ssim/valid.shape[0]))

ssim = 0
for i in range(valid.shape[0]-1):
  bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_CUBIC)
  ssim = ssim+skimage.measure.compare_ssim(valid[i]2, bicubic_img, multichannel=True)

print("Bicubic Interpolation:    " + str(ssim/valid.shape[0]))

ssim = 0
for i in range(valid.shape[0]-1):
  bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_LINEAR
  ssim = ssim+skimage.measure.compare_ssim(valid[i].astype(float), bicubic_img, multicha

print("Bilinear Interpolation:   " + str(ssim/valid.shape[0]))

ssim = 0
for i in range(valid.shape[0]-1):
  bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_LANCZO
  ssim = ssim+skimage.measure.compare_ssim(valid[i].astype(float), bicubic_img, multicha

print("Lanczos Interpolation:    " + str(ssim/valid.shape[0]))
```

```
Number of images used for comparison present in valid set: 561
SSIM Measure:
Auto Encoder [our model]: 0.8139050785067062
Bicubic Interpolation:    0.10943069082587507
Bilinear Interpolation:   0.09642309083303438
Lanczos Interpolation:    0.10960605899220008
```

```
print("Number of images used for comparison present in valid set: "+str(valid.shape[0]))

print("PSNR Measure:")
psnr = 0
for i in range(valid.shape[0]-1):
  psnr = psnr+skimage.measure.compare_psnr(valid[i], decoded_imgs[i])

print("Auto Encoder [our model]: " + str(psnr/valid.shape[0]))

psnr = 0
for i in range(valid.shape[0]-1):
  bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_CUBIC)
  psnr = psnr+skimage.measure.compare_psnr(valid[i].astype(float), bicubic_img)

print("Bicubic Interpolation:    " + str(psnr/valid.shape[0]))

psnr = 0
for i in range(valid.shape[0]-1):
  bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_LINEAR
```



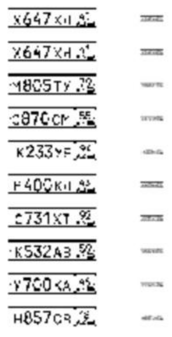
5/25/2019 super_resolution_auto_encoder.ipynb - Colaboratory

```
    psnr = psnr+skimage.measure.compare_psnr(valid[i].astype(float), bicubic_img)

  print("Bilinear Interpolation:   " + str(psnr/valid.shape[0]))

  psnr = 0
  for i in range(valid.shape[0]-1):
    bicubic_img = cv.resize(valid_low[80],None,fx=4, fy=4, interpolation = cv.INTER_LANCZOS
    psnr = psnr+skimage.measure.compare_psnr(valid[i].astype(float), bicubic_img)

  print("Lanczos Interpolation:    " + str(psnr/valid.shape[0]))
```

```
Number of images used for comparison present in valid set: 561
PSNR Measure:
Auto Encoder [our model]: 14.748694261566309
Bicubic Interpolation:    7.417829645686029
Bilinear Interpolation:   7.435301062765469
Lanczos Interpolation:    7.411120044374099
```

```
m=0
a = valid[m].reshape(36,148)


h_pad = (3*x_train.shape[1]//8)
w_pad = (3*x_train.shape[2]//8)
b = cv.copyMakeBorder(valid_low[m].reshape(9,37),3*x_train.shape[1]//8+1,3*x_train.shape
print(b.shape)

img = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.cop

for m in range(9):
  a = valid[m].reshape(36,148)


  h_pad = (3*x_train.shape[1]//8)
  w_pad = (3*x_train.shape[2]//8)
  b = cv.copyMakeBorder(valid_low[m].reshape(9,37),3*x_train.shape[1]//8+1,3*x_train.shap


  img1 = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv

  img = np.vstack((img,img1))

plt.axis('off')
plt.imshow(255-img, cmap='gray')

plt.savefig(path+'img/'+'super_res_preprocess.png', dpi=300)
```

```
(36, 148)
```



https://colab.research.google.com/drive/1sCRLhcawDyFNOjcAkQoYgDr4HvulUMM8#scrollTo=SJ3CDTWNUCTi&printMode=true 6/7

```
from google.colab import drive
drive.mount('/mnt/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
··········
Mounted at /mnt/
```

```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
```

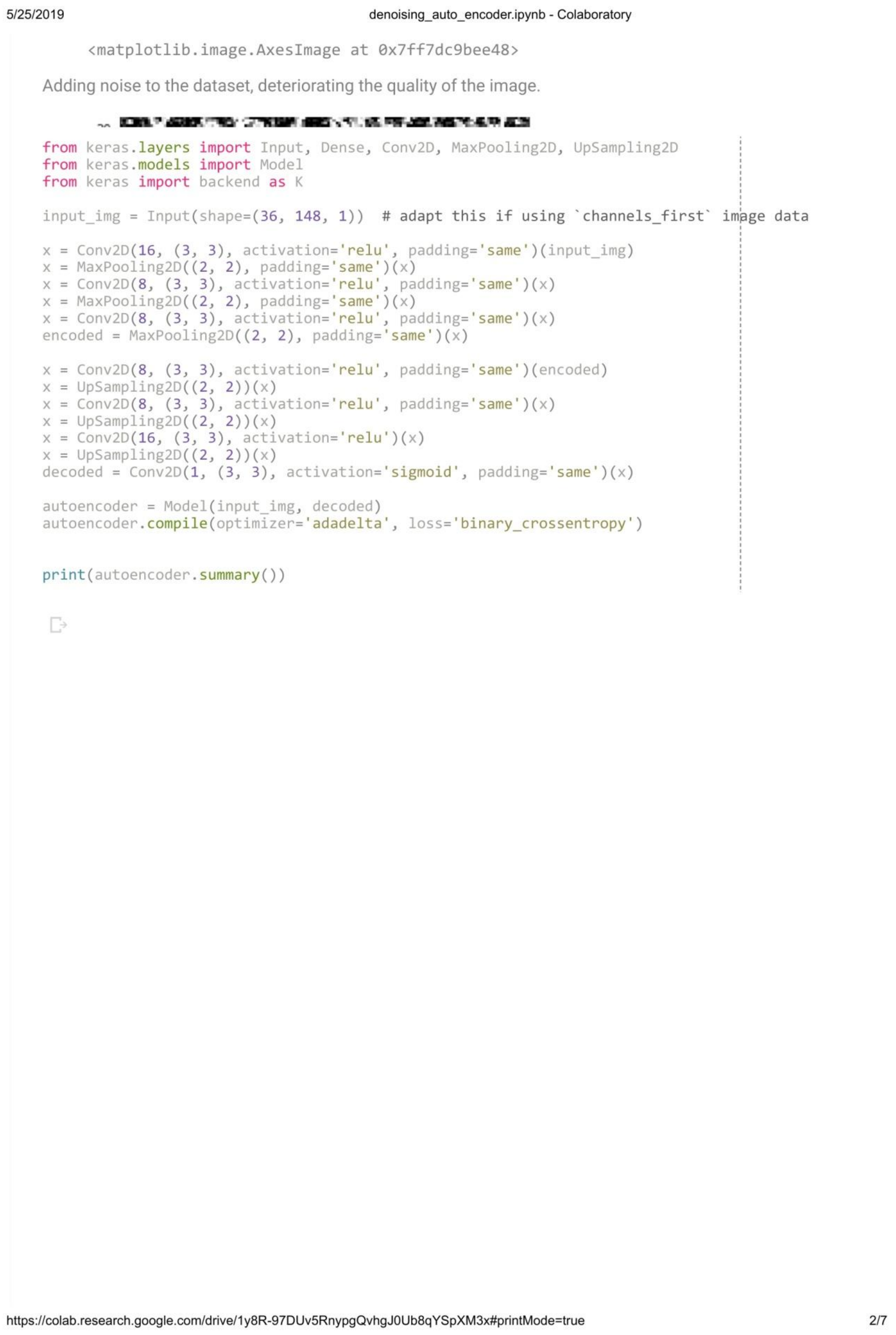
```
path = '/mnt/My Drive/Number_Plate/'
valid = np.load(path+'data/test.npy')
train_data = np.load(path+'data/train.npy')

train_data = 255-train_data
valid = 255-valid

x_train, x_test = np.split(train_data,[int(3*len(train_data)/4)])
```

```
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
valid = valid.astype('float32') / 255.

plt.imshow(x_train[0].reshape(36,148), cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7ff7df2b2048>
```

Plotting the original image.

```
noise_factor = 0.3
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_tra
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_test.
valid_noisy = valid + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=valid.sha

x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
valid_noisy = np.clip(valid_noisy, 0., 1.)

plt.imshow(x_train_noisy[0].reshape(36,148), cmap='gray')
```



5/25/2019 denoising_auto_encoder.ipynb - Colaboratory

```
<matplotlib.image.AxesImage at 0x7ff7dc9bee48>
```

Adding noise to the dataset, deteriorating the quality of the image.

```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
from keras import backend as K

input_img = Input(shape=(36, 148, 1))  # adapt this if using `channels_first` image data

x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(16, (3, 3), activation='relu')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')


print(autoencoder.summary())
```

https://colab.research.google.com/drive/1y8R-97DUv5RnypgQvhgJ0Ub8qYSpXM3x#printMode=true 2/7

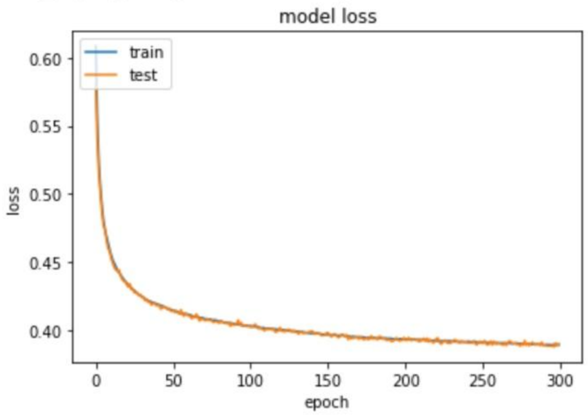
5/25/2019 denoising_auto_encoder.ipynb - Colaboratory

```
Using TensorFlow backend.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/f
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 36, 148, 1)        0
```

```
'''history = autoencoder.fit(x_train_noisy, x_train,
                epochs=300,
                batch_size=128,
                shuffle=True,
                validation_data=(x_test_noisy, x_test))'''
#autoencoder.save_weights(path+'weights/weightsfile_denoising.h5')
```

```
print(history.history.keys())
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.savefig(path+'img/'+'denoising_training.png', dpi=300)
plt.show()
```

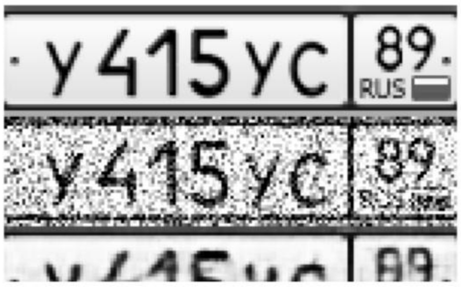
```
dict_keys(['val_loss', 'loss'])
```



Plotting the training loss.

```
autoencoder.load_weights(path+'weights/weightsfile_denoising.h5')
decoded_imgs = 255*autoencoder.predict(valid_noisy)
a = 255*valid[10].reshape(36,148) * 255
b = 255*valid_noisy[10].reshape(36, 148) * 255
c = decoded_imgs[10].reshape(36, 148) * 255
img1 = np.vstack((a,b))
img = np.vstack((img1,c))
plt.axis('off')
plt.imshow(255-img, cmap='gray')
plt.savefig(path+'img/'+'denoising_result.png', dpi=300)
```

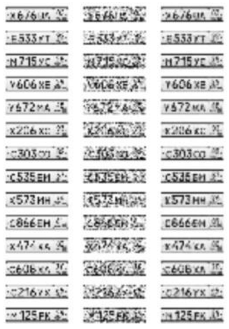
https://colab.research.google.com/drive/1y8R-97DUv5RnypgQvhgJ0Ub8qYSpXM3x#printMode=true 3/7



The above image shows result on validation set:

1. The original image
2. The Noisy image
3. Restored image with Autoencoder.

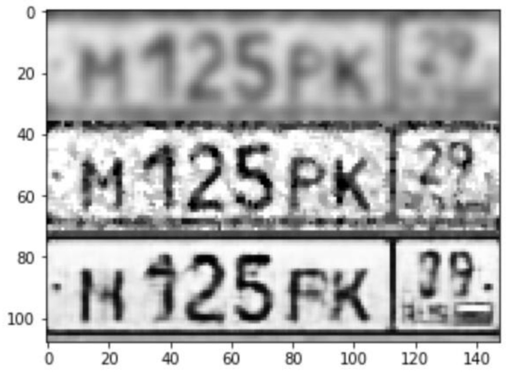
```
a = cv.copyMakeBorder(255*valid[50].reshape(36,148),10,10,10,10,cv.BORDER_CONSTANT,value=
b = cv.copyMakeBorder(255*valid_noisy[50].reshape(36, 148),10,10,10,10,cv.BORDER_CONSTAN
c = cv.copyMakeBorder(decoded_imgs[50].reshape(36, 148),10,10,10,10,cv.BORDER_CONSTANT,va
for i in range(12,25):
    img =255* valid[i].reshape(36,148)
    img_n = 255*valid_noisy[i].reshape(36,148)
    res = decoded_imgs[i].reshape(36, 148)
    a = np.vstack((a, cv.copyMakeBorder(img,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]
    b = np.vstack((b, cv.copyMakeBorder(img_n,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,
    c = np.vstack((c, cv.copyMakeBorder(res,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]

img1 = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.co
img = np.hstack((img1,cv.copyMakeBorder(c,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]))
plt.axis('off')
plt.imshow(255-img, cmap='gray')
plt.savefig(path+'img/'+'denoising_output2.png', dpi=300)
```



```
a = cv.copyMakeBorder(255*valid[50].reshape(36,148),10,10,10,10,cv.BORDER_CONSTANT,value=
b = cv.copyMakeBorder(255*valid_noisy[50].reshape(36, 148),10,10,10,10,cv.BORDER_CONSTAN
for i in range(12,25):
    img =255* valid[i].reshape(36,148)
    img_n = 255*valid_noisy[i].reshape(36,148)
    res = decoded_imgs[i].reshape(36, 148)
    a = np.vstack((a, cv.copyMakeBorder(img,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]
    b = np.vstack((b, cv.copyMakeBorder(img_n,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,

img = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.co
plt.axis('off')
plt.imshow(255-img, cmap='gray')
plt.savefig(path+'img/'+'denoising_preprocess.png', dpi=300)
```

Viewing multiple results from the validation dataset.

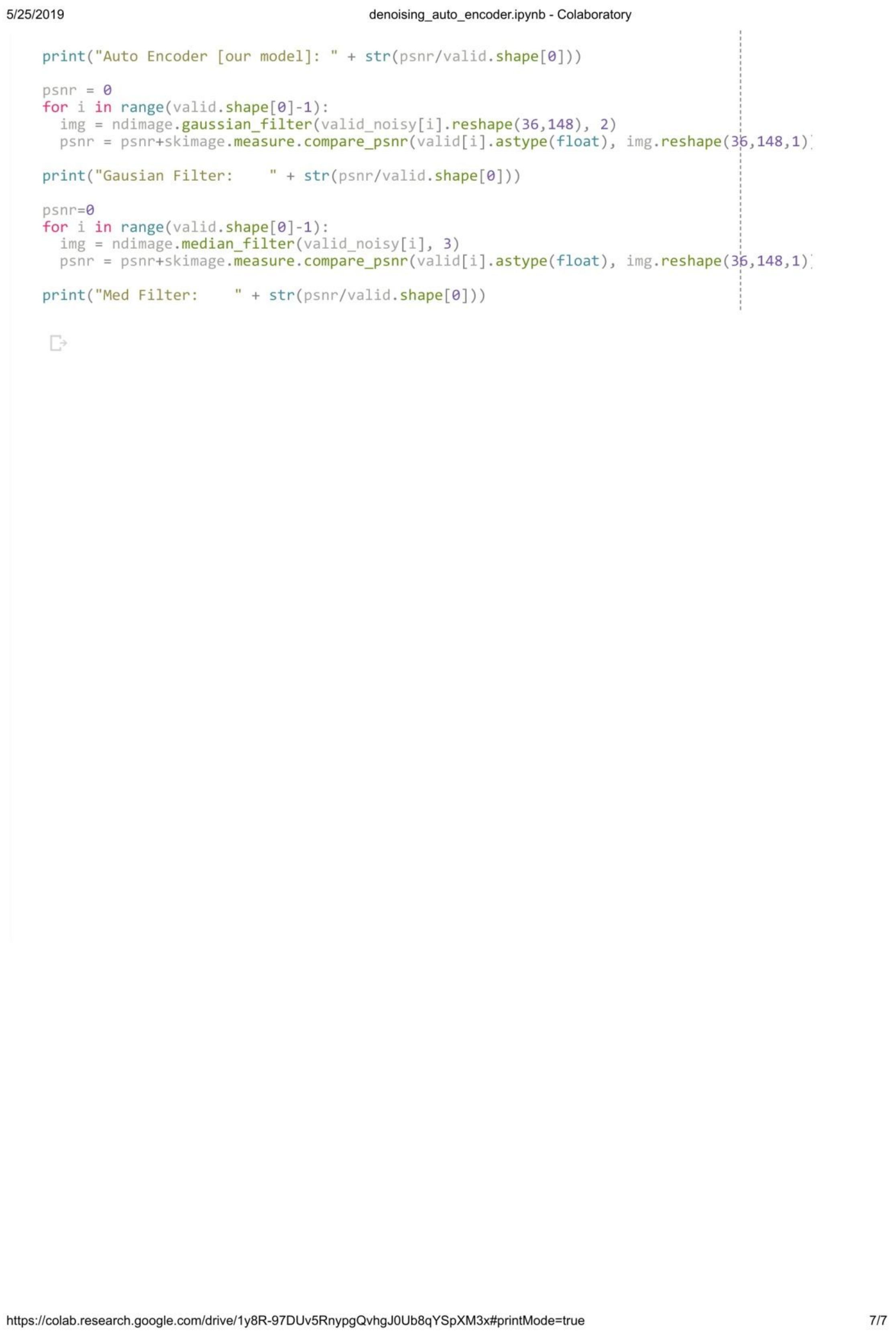
```
from scipy import ndimage
noisy = 255* valid_noisy[i].reshape(36,148)
gauss_denoised = ndimage.gaussian_filter(255* valid_noisy[i].reshape(36,148), 2)
med_denoised = ndimage.median_filter(noisy, 3)
img = np.vstack((gauss_denoised,med_denoised))
img = np.vstack((img,decoded_imgs[i].reshape(36, 148)))
plt.imshow(255-img,cmap='gray')
plt.savefig(path+'img/'+'denoising_output_comp.png', dpi=300)
```



```
a = cv.copyMakeBorder(255*valid[78].reshape(36,148),10,10,10,10,cv.BORDER_CONSTANT,value
b = cv.copyMakeBorder(255*valid_noisy[78].reshape(36, 148),10,10,10,10,cv.BORDER_CONSTAN
c = cv.copyMakeBorder(decoded_imgs[78].reshape(36, 148),10,10,10,10,cv.BORDER_CONSTANT,v
d = cv.copyMakeBorder(ndimage.gaussian_filter(255* valid_noisy[78].reshape(36,148), 2),1
e = cv.copyMakeBorder(ndimage.median_filter(255* valid_noisy[78].reshape(36,148), 3),10,
for i in range(12,48):
    img =255* valid[i].reshape(36,148)
    img_n = 255*valid_noisy[i].reshape(36,148)
    res = decoded_imgs[i].reshape(36, 148)
    gaus = ndimage.gaussian_filter(255* valid_noisy[i].reshape(36,148), 2)
    medf = ndimage.median_filter(255* valid_noisy[i].reshape(36,148), 3)
    a = np.vstack((a, cv.copyMakeBorder(img,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]
    b = np.vstack((b, cv.copyMakeBorder(img_n,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,
      c = np.vstack((c, cv.copyMakeBorder(res,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]
      d = np.vstack((d, cv.copyMakeBorder(gaus,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0
      e = np.vstack((e, cv.copyMakeBorder(medf,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0

img1 = np.hstack((cv.copyMakeBorder(a,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.c
img2 = np.hstack((cv.copyMakeBorder(d,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]),cv.c
img3 = np.hstack((img1,img2))
img = np.hstack((img3,cv.copyMakeBorder(c,10,10,10,10,cv.BORDER_CONSTANT,value=[0,0,0]))
plt.axis('off')
plt.imshow(255-img, cmap='gray')
plt.savefig(path+'img/'+'denoising_comb_output2.png', dpi=300)
```

```
import skimage.measure

print("Number of images used for comparison present in valid set: "+str(valid.shape[0]))

print("SSIM Measure:")
ssim = 0
for i in range(valid.shape[0]-1):
  ssim = ssim+skimage.measure.compare_ssim(valid[i], decoded_imgs[i]/255, multichannel=T

print("Auto Encoder [our model]: " + str(ssim/valid.shape[0]))

ssim = 0
for i in range(valid.shape[0]-1):
  img = ndimage.gaussian_filter(valid_noisy[i].reshape(36,148), 2)
  ssim = ssim+skimage.measure.compare_ssim(valid[i].astype(float), img.reshape(36,148,1)

print("Gausian Filter:    " + str(ssim/valid.shape[0]))

ssim=0
for i in range(valid.shape[0]-1):
  img = ndimage.median_filter(valid_noisy[i], 3)
  ssim = ssim+skimage.measure.compare_ssim(valid[i].astype(float), img.reshape(36,148,1)

print("Med Filter:    " + str(ssim/valid.shape[0]))
```

```
import skimage.measure

print("Number of images used for comparison present in valid set: "+str(valid.shape[0]))

print("PSNR Measure:")
psnr = 0
for i in range(valid.shape[0]-1):
  psnr = psnr+skimage.measure.compare_psnr(valid[i], decoded_imgs[i]/255)
```



5/25/2019 denoising_auto_encoder.ipynb - Colaboratory

```
print("Auto Encoder [our model]: " + str(psnr/valid.shape[0]))

psnr = 0
for i in range(valid.shape[0]-1):
  img = ndimage.gaussian_filter(valid_noisy[i].reshape(36,148), 2)
  psnr = psnr+skimage.measure.compare_psnr(valid[i].astype(float), img.reshape(36,148,1)

print("Gausian Filter:    " + str(psnr/valid.shape[0]))

psnr=0
for i in range(valid.shape[0]-1):
  img = ndimage.median_filter(valid_noisy[i], 3)
  psnr = psnr+skimage.measure.compare_psnr(valid[i].astype(float), img.reshape(36,148,1)

print("Med Filter:    " + str(psnr/valid.shape[0]))
```

https://colab.research.google.com/drive/1y8R-97DUv5RnypgQvhgJ0Ub8qYSpXM3x#printMode=true 7/7