

# CS512 COMPUTER VISION: PROJECT REPORT

## AUTOENCODERS FOR NOISE REDUCTION IN IMAGES

By

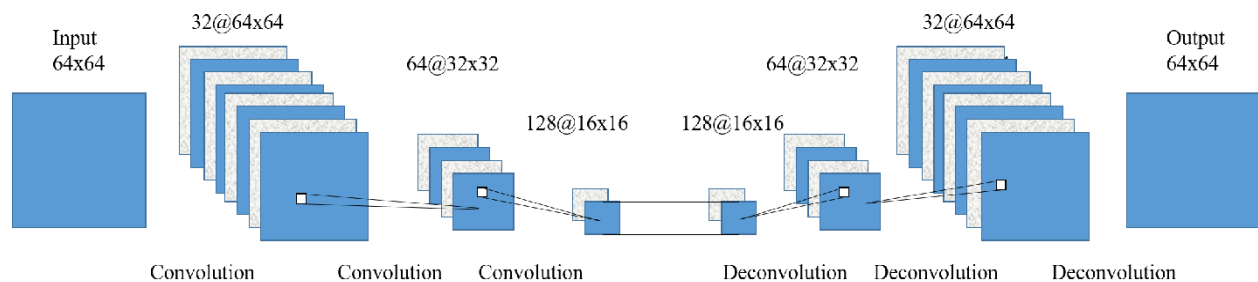
Russi Sinha and Vaibhav Sharma

### Abstract

The last decade has seen an astronomical shift from imaging with DSLR and point-and-shoot cameras to imaging with smartphone cameras. Due to the small aperture and sensor size, smartphone images have notably more noise than their DSLR counterparts. We follow the research on denoising of sonar images using Neural networks and try to implement the same denoising techniques on the mobile image dataset. We use a Smartphone Image Denoising Dataset(SIDD) – of ~2000 noisy images from 1 scene under different lighting using five representative smartphone cameras and generated their ground truth images. We try to show the performance of CNN based model using differing parameters but with the same Layer structure.

### 1. Introduction

Smartphone images now vastly outnumber images captured with DSLR and point-and-shoot cameras. But while the prevalence of smartphones makes them a convenient device for photography, their images are typically degraded by higher levels of noise due to the smaller sensors and lenses found in their cameras. This problem has heightened the need for progress in image denoising, particularly in the context of smartphone imagery. Many of the approaches used to produce noise-free ground truth images are not fully sufficient, especially for the case of smartphone cameras. For example, the common strategy of using low ISO and long exposure to acquire a “noise free” image is not applicable to smartphone cameras, as noise is still significant on such images even with the best camera settings. We implement a solution of using a denoising auto-encoder for image enhancement. The auto-encoder is a neural-network based algorithm that conducts supervised machine learning. It includes convolution networks and deconvolution networks with the same input and output matrix.



## 2. Background

### 2.1. Image Capture

The images are static indoor scenes to avoid misalignments caused by scene motion. In addition, a direct current (DC) light source is used to avoid the flickering effect of alternating current (AC) lights. There are adjustments of illumination brightness and color temperature (ranging from 3200K to 5500K). Five smartphone cameras (Apple iPhone 7, Google Pixel, Samsung Galaxy S6 Edge, Motorola Nexus 6, and LG G4) are used to capture these images.

The images are captured using the following protocol. Each scene is captured multiple times using different cameras, different settings, and/or different lighting conditions. Each combination of these is called a scene instance. For each scene instance, a sequence of successive images are captured, with a 1–2-second time interval between subsequent images. While capturing an image sequence, all camera settings (e.g., ISO, exposure, focus, white balance, exposure compensation) are fixed throughout the process.

Images are captured in 10 different scenes using five smartphone cameras under four different combinations (on average) of the following settings and conditions:

- 15 different ISO levels ranging from 50 up to 10,000 to obtain a variety of noise levels (the higher the ISO level, the higher the noise)
- Three illumination temperatures to simulate the effect of different light sources: 3200K for tungsten or halogen, 4400K for fluorescent lamps, and 5500K for daylight
- Three light brightness levels: low, normal, and high

For each scene instance, a sequence of 150 successive images are captured. Since noise is a random process, each image contains a random sample from the sensor's noise distribution. Therefore, the total number of images in our dataset – the Smartphone Image Denoising Dataset (SIDDD) – is ~3000 (5 cameras  $\times$  4 conditions  $\times$  150 images). For each image, corresponding ground truth image is generated and recording all settings with the raw data in DNG/Tiff files.

### 2.2. Machine Learning

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction. Autoencoders are Neural Networks which are commonly used for feature selection and extraction. The principle behind denoising autoencoders is to be able to reconstruct data from an input of corrupted data. After giving the autoencoder the corrupted data, we force the hidden layer to learn only the more robust features, rather than just the identity. The output will then be a more refined version of the input data. In general, the percentage of input nodes which are being set to zero is about 50%. Data denoising and dimensionality reduction for data visualization are considered as two main practical applications of autoencoders. With appropriate dimensionality and sparsity constraints, autoencoders can learn data projections that are more interesting than PCA or other basic techniques.

### 3. Methodology

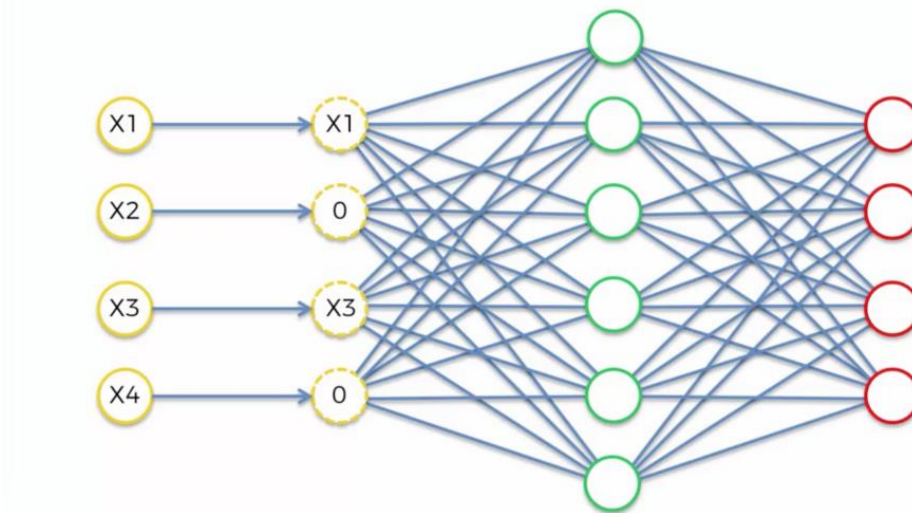
#### 3.1. Denoising Auto-encoder

Autoencoders are Neural Networks which are commonly used for feature selection and extraction. However, when there are more nodes in the hidden layer than there are inputs, the Network is risking to learn the so-called “Identity Function”, also called “Null Function”, meaning that the output equals the input, marking the Autoencoder useless.

Denoising Autoencoders solve this problem by randomly corrupting the data on purpose and inputting them into a neural network. One way to corrupt the data would be simply to randomly remove some parts of the data, so that the autoencoder is trying to predict the missing input.

The principle behind denoising autoencoders is to be able to reconstruct data from an input of corrupted data. After giving the autoencoder the corrupted data, we force the hidden layer to learn only the more robust features, rather than just the identity. The output will then be a more refined version of the input data.

In general, the percentage of input nodes which are being set to zero is about 50%. Other sources suggest a lower count, such as 30%. It depends on the amount of data and input nodes you have.



The auto-encoder has deconvolution layers to restore their input size. The artificial neural network has six layers in total and has three convolution processes and three deconvolution processes. The first layer has 32 depths, the second has 64 depths, and the third has 128 depths. However, since the size of the actual image is larger than 64×64, the structure should be changed slightly after learning. The actual learned values do not need to be re-learned because they are the weights and biases of each convolution and deconvolution. In addition, it is unsupervised machine learning because one image is used for both input and output layers. First, the images are cropped as [64, 64] size. Next, each cropped images artificially gets gaussian-noise and inserts to input layer. Finally, the output data is original cropped image and the model trains a great deal of images.

When calculating the Loss function, it is important to compare the output values with the original input, not with the corrupted input. That way, the risk of learning the identity function instead of extracting features is eliminated.

### 3.2. Image Resolution Restore Strategy

In an auto-encoder network, the resolution is halved by pooling when passing through the convolution layer. It is restored to the deconvolution layers, however the actual result is blurred by the resolution value that was decreased in the previous step. When passing through one layer, the width and height are reduced by half each, four times smaller, and one more layer is passed, so that it becomes 16 times smaller. In order to solve this problem, we artificially increased the image resolution by 16 times and put it into the network for the resolution recovery. Finally, the result is a 16× larger image, which is the last image reduction to make it the same size as the original image.

In contrast to convolutional layers, in which multiple input activations within a filter window are fused to output a single activation, deconvolutional layers associate a single input activation with multiple outputs. Deconvolution is usually used as learnable up-sampling layers. One can simply replace deconvolution with convolution, which results in an architecture that is very similar to recently proposed very deep fully convolutional neural networks.

Firstly, in the fully convolution case, the noise is eliminated step by step, i.e., the noise level is reduced after each layer. During this process, the details of the image content may be lost. Nevertheless, in the network with convolution and deconvolution, convolution preserves the primary image content. Then deconvolution is used to compensate the details. For fully convolutional networks, we use padding and up-sample the input to make the input and output the same size. For the combined network, the first 5 layers are convolutional, and the second 5 layers are deconvolutional. All the other parameters for training are the same.

On the other hand, to apply deep learning models on devices with limited computing power such as mobile phones, one has to speed-up the testing phase. In this situation, it is better to use downsampling in convolutional layers to reduce the size of the feature maps. In order to obtain an output of the same size as the input, deconvolution is used to up-sample the feature maps in the symmetric deconvolutional layers. We experimentally found that the testing efficiency can be well improved with almost negligible performance degradation.

## 4. Result

### 4.1. Training

#### 4.1.1. Data-set

Our data set consists of a scene captured multiple times using different phone cameras, different settings, and/or different lighting conditions, each of which is called a scene instance. The five smartphone cameras that were used to capture the scenes are Apple iPhone 7, Google Pixel, Samsung Galaxy S6 Edge, Motorola Nexus 6, and LG G4.

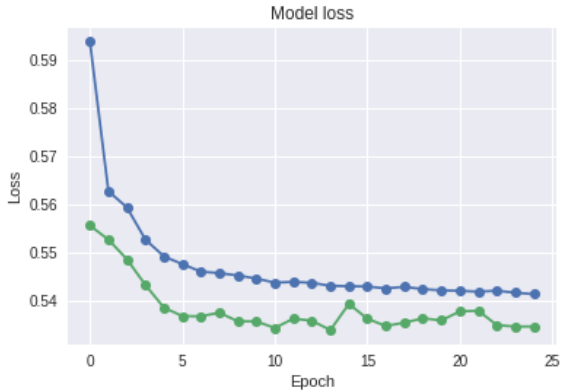
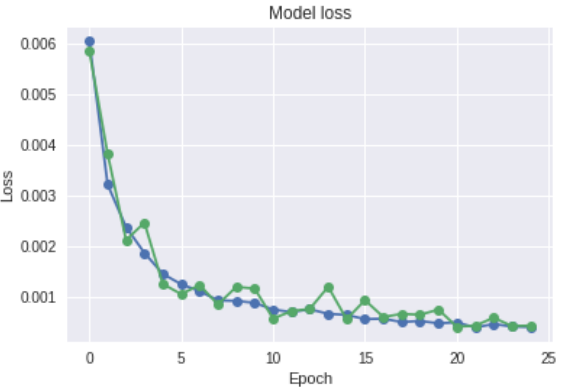
For each scene instance, a sequence of 150 successive images are captured, with a 1–2-second time interval between subsequent images. While capturing an image sequence, all camera settings (e.g., ISO, exposure, focus, white balance, exposure compensation) are fixed throughout the process.

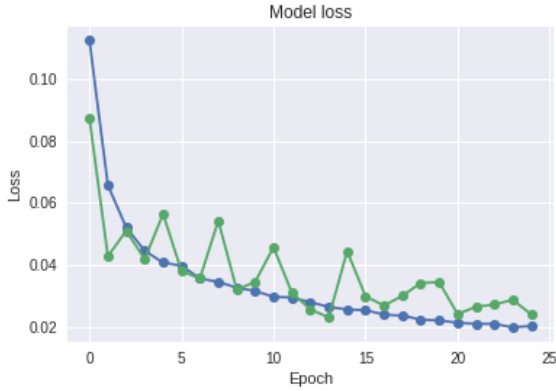
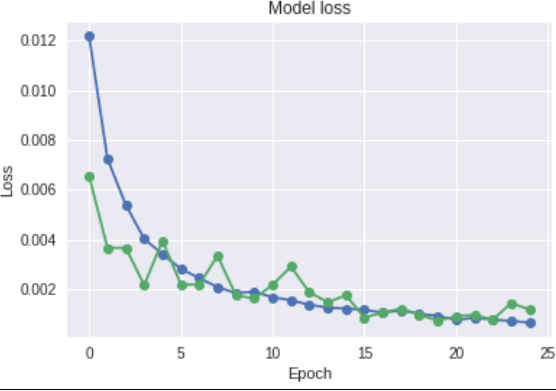
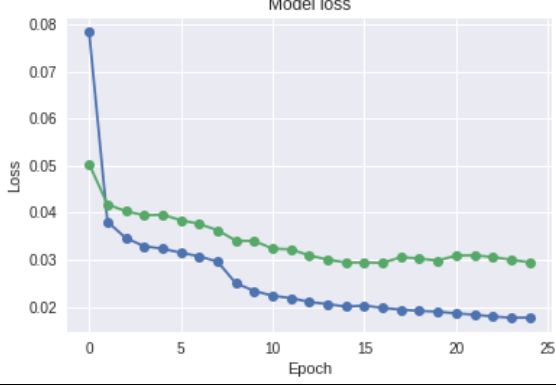
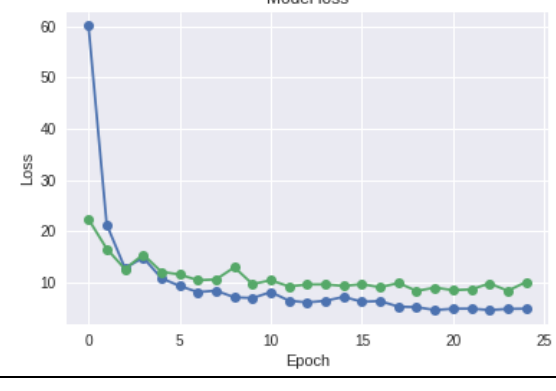
Each of these images are cropped to 12 smaller images of size 64x64 which gives us a total Of 9000 images ( $150 \times 12 \times 5 = 9000$ )

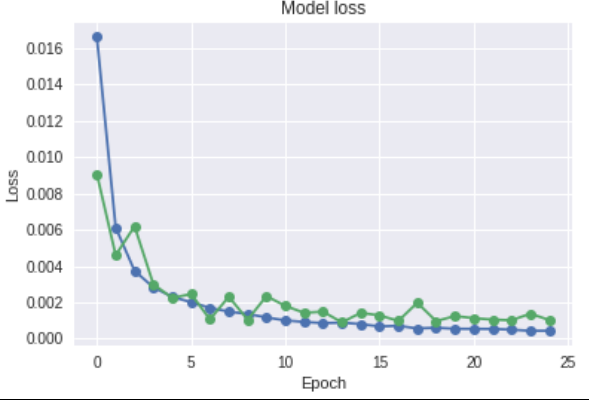
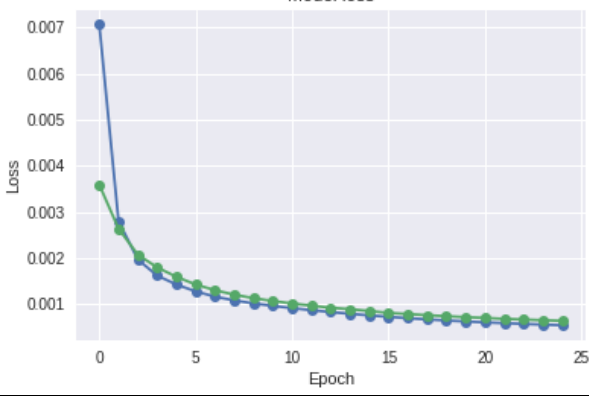
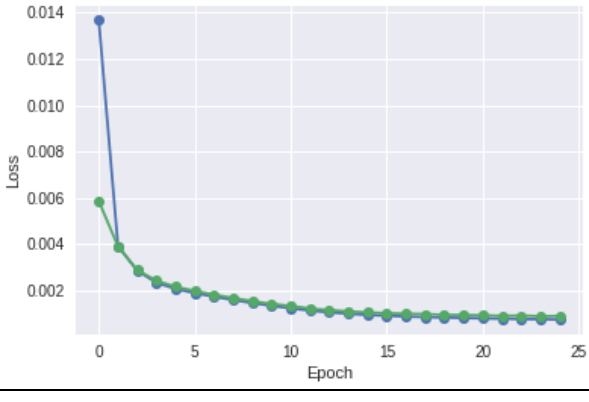
4.1.2. Loss Function

The model was trained using various optimizers and loss functions to test for which one gives the best results. The loss values of each of them were plotted in graphs. Few of them are mentioned below:



Optimizer	Loss function	Loss values
Adadelta	Binary crossentropy	
Adadelta	Logcosh	

Adadelta	Mean absolute error	 <p>Model loss</p> <table border="1"><thead><tr><th>Epoch</th><th>Training Loss (Blue)</th><th>Validation Loss (Green)</th></tr></thead><tbody><tr><td>0</td><td>0.11</td><td>0.09</td></tr><tr><td>1</td><td>0.065</td><td>0.045</td></tr><tr><td>2</td><td>0.05</td><td>0.05</td></tr><tr><td>3</td><td>0.045</td><td>0.04</td></tr><tr><td>4</td><td>0.04</td><td>0.055</td></tr><tr><td>5</td><td>0.038</td><td>0.04</td></tr><tr><td>6</td><td>0.035</td><td>0.035</td></tr><tr><td>7</td><td>0.035</td><td>0.055</td></tr><tr><td>8</td><td>0.032</td><td>0.03</td></tr><tr><td>9</td><td>0.03</td><td>0.035</td></tr><tr><td>10</td><td>0.028</td><td>0.045</td></tr><tr><td>11</td><td>0.028</td><td>0.03</td></tr><tr><td>12</td><td>0.025</td><td>0.025</td></tr><tr><td>13</td><td>0.025</td><td>0.02</td></tr><tr><td>14</td><td>0.025</td><td>0.045</td></tr><tr><td>15</td><td>0.025</td><td>0.03</td></tr><tr><td>16</td><td>0.025</td><td>0.025</td></tr><tr><td>17</td><td>0.025</td><td>0.03</td></tr><tr><td>18</td><td>0.025</td><td>0.035</td></tr><tr><td>19</td><td>0.022</td><td>0.035</td></tr><tr><td>20</td><td>0.022</td><td>0.025</td></tr><tr><td>21</td><td>0.022</td><td>0.025</td></tr><tr><td>22</td><td>0.022</td><td>0.028</td></tr><tr><td>23</td><td>0.022</td><td>0.028</td></tr><tr><td>24</td><td>0.02</td><td>0.025</td></tr><tr><td>25</td><td>0.02</td><td>0.025</td></tr></tbody></table>	Epoch	Training Loss (Blue)	Validation Loss (Green)	0	0.11	0.09	1	0.065	0.045	2	0.05	0.05	3	0.045	0.04	4	0.04	0.055	5	0.038	0.04	6	0.035	0.035	7	0.035	0.055	8	0.032	0.03	9	0.03	0.035	10	0.028	0.045	11	0.028	0.03	12	0.025	0.025	13	0.025	0.02	14	0.025	0.045	15	0.025	0.03	16	0.025	0.025	17	0.025	0.03	18	0.025	0.035	19	0.022	0.035	20	0.022	0.025	21	0.022	0.025	22	0.022	0.028	23	0.022	0.028	24	0.02	0.025	25	0.02	0.025
Epoch	Training Loss (Blue)	Validation Loss (Green)																																																																																	
0	0.11	0.09																																																																																	
1	0.065	0.045																																																																																	
2	0.05	0.05																																																																																	
3	0.045	0.04																																																																																	
4	0.04	0.055																																																																																	
5	0.038	0.04																																																																																	
6	0.035	0.035																																																																																	
7	0.035	0.055																																																																																	
8	0.032	0.03																																																																																	
9	0.03	0.035																																																																																	
10	0.028	0.045																																																																																	
11	0.028	0.03																																																																																	
12	0.025	0.025																																																																																	
13	0.025	0.02																																																																																	
14	0.025	0.045																																																																																	
15	0.025	0.03																																																																																	
16	0.025	0.025																																																																																	
17	0.025	0.03																																																																																	
18	0.025	0.035																																																																																	
19	0.022	0.035																																																																																	
20	0.022	0.025																																																																																	
21	0.022	0.025																																																																																	
22	0.022	0.028																																																																																	
23	0.022	0.028																																																																																	
24	0.02	0.025																																																																																	
25	0.02	0.025																																																																																	
Adadelta	Mean squared error	 <p>Model loss</p> <table border="1"><thead><tr><th>Epoch</th><th>Training Loss (Blue)</th><th>Validation Loss (Green)</th></tr></thead><tbody><tr><td>0</td><td>0.012</td><td>0.0065</td></tr><tr><td>1</td><td>0.007</td><td>0.0035</td></tr><tr><td>2</td><td>0.0055</td><td>0.0035</td></tr><tr><td>3</td><td>0.0045</td><td>0.002</td></tr><tr><td>4</td><td>0.004</td><td>0.004</td></tr><tr><td>5</td><td>0.003</td><td>0.002</td></tr><tr><td>6</td><td>0.0025</td><td>0.002</td></tr><tr><td>7</td><td>0.002</td><td>0.0035</td></tr><tr><td>8</td><td>0.0018</td><td>0.0018</td></tr><tr><td>9</td><td>0.0018</td><td>0.002</td></tr><tr><td>10</td><td>0.0018</td><td>0.002</td></tr><tr><td>11</td><td>0.0015</td><td>0.003</td></tr><tr><td>12</td><td>0.0015</td><td>0.002</td></tr><tr><td>13</td><td>0.0015</td><td>0.0015</td></tr><tr><td>14</td><td>0.0015</td><td>0.002</td></tr><tr><td>15</td><td>0.0015</td><td>0.0012</td></tr><tr><td>16</td><td>0.0015</td><td>0.0015</td></tr><tr><td>17</td><td>0.0015</td><td>0.0015</td></tr><tr><td>18</td><td>0.0012</td><td>0.0015</td></tr><tr><td>19</td><td>0.0012</td><td>0.001</td></tr><tr><td>20</td><td>0.0012</td><td>0.0012</td></tr><tr><td>21</td><td>0.0012</td><td>0.0012</td></tr><tr><td>22</td><td>0.0012</td><td>0.0012</td></tr><tr><td>23</td><td>0.0012</td><td>0.0015</td></tr><tr><td>24</td><td>0.0012</td><td>0.0012</td></tr><tr><td>25</td><td>0.0012</td><td>0.0012</td></tr></tbody></table>	Epoch	Training Loss (Blue)	Validation Loss (Green)	0	0.012	0.0065	1	0.007	0.0035	2	0.0055	0.0035	3	0.0045	0.002	4	0.004	0.004	5	0.003	0.002	6	0.0025	0.002	7	0.002	0.0035	8	0.0018	0.0018	9	0.0018	0.002	10	0.0018	0.002	11	0.0015	0.003	12	0.0015	0.002	13	0.0015	0.0015	14	0.0015	0.002	15	0.0015	0.0012	16	0.0015	0.0015	17	0.0015	0.0015	18	0.0012	0.0015	19	0.0012	0.001	20	0.0012	0.0012	21	0.0012	0.0012	22	0.0012	0.0012	23	0.0012	0.0015	24	0.0012	0.0012	25	0.0012	0.0012
Epoch	Training Loss (Blue)	Validation Loss (Green)																																																																																	
0	0.012	0.0065																																																																																	
1	0.007	0.0035																																																																																	
2	0.0055	0.0035																																																																																	
3	0.0045	0.002																																																																																	
4	0.004	0.004																																																																																	
5	0.003	0.002																																																																																	
6	0.0025	0.002																																																																																	
7	0.002	0.0035																																																																																	
8	0.0018	0.0018																																																																																	
9	0.0018	0.002																																																																																	
10	0.0018	0.002																																																																																	
11	0.0015	0.003																																																																																	
12	0.0015	0.002																																																																																	
13	0.0015	0.0015																																																																																	
14	0.0015	0.002																																																																																	
15	0.0015	0.0012																																																																																	
16	0.0015	0.0015																																																																																	
17	0.0015	0.0015																																																																																	
18	0.0012	0.0015																																																																																	
19	0.0012	0.001																																																																																	
20	0.0012	0.0012																																																																																	
21	0.0012	0.0012																																																																																	
22	0.0012	0.0012																																																																																	
23	0.0012	0.0015																																																																																	
24	0.0012	0.0012																																																																																	
25	0.0012	0.0012																																																																																	
Adam	Mean absolute error	 <p>Model loss</p> <table border="1"><thead><tr><th>Epoch</th><th>Training Loss (Blue)</th><th>Validation Loss (Green)</th></tr></thead><tbody><tr><td>0</td><td>0.078</td><td>0.05</td></tr><tr><td>1</td><td>0.038</td><td>0.042</td></tr><tr><td>2</td><td>0.035</td><td>0.04</td></tr><tr><td>3</td><td>0.032</td><td>0.04</td></tr><tr><td>4</td><td>0.032</td><td>0.04</td></tr><tr><td>5</td><td>0.03</td><td>0.038</td></tr><tr><td>6</td><td>0.03</td><td>0.038</td></tr><tr><td>7</td><td>0.03</td><td>0.035</td></tr><tr><td>8</td><td>0.025</td><td>0.035</td></tr><tr><td>9</td><td>0.022</td><td>0.032</td></tr><tr><td>10</td><td>0.022</td><td>0.032</td></tr><tr><td>11</td><td>0.022</td><td>0.03</td></tr><tr><td>12</td><td>0.02</td><td>0.03</td></tr><tr><td>13</td><td>0.02</td><td>0.028</td></tr><tr><td>14</td><td>0.02</td><td>0.028</td></tr><tr><td>15</td><td>0.02</td><td>0.028</td></tr><tr><td>16</td><td>0.02</td><td>0.03</td></tr><tr><td>17</td><td>0.02</td><td>0.03</td></tr><tr><td>18</td><td>0.02</td><td>0.03</td></tr><tr><td>19</td><td>0.018</td><td>0.03</td></tr><tr><td>20</td><td>0.018</td><td>0.03</td></tr><tr><td>21</td><td>0.018</td><td>0.03</td></tr><tr><td>22</td><td>0.018</td><td>0.03</td></tr><tr><td>23</td><td>0.018</td><td>0.03</td></tr><tr><td>24</td><td>0.018</td><td>0.03</td></tr><tr><td>25</td><td>0.018</td><td>0.03</td></tr></tbody></table>	Epoch	Training Loss (Blue)	Validation Loss (Green)	0	0.078	0.05	1	0.038	0.042	2	0.035	0.04	3	0.032	0.04	4	0.032	0.04	5	0.03	0.038	6	0.03	0.038	7	0.03	0.035	8	0.025	0.035	9	0.022	0.032	10	0.022	0.032	11	0.022	0.03	12	0.02	0.03	13	0.02	0.028	14	0.02	0.028	15	0.02	0.028	16	0.02	0.03	17	0.02	0.03	18	0.02	0.03	19	0.018	0.03	20	0.018	0.03	21	0.018	0.03	22	0.018	0.03	23	0.018	0.03	24	0.018	0.03	25	0.018	0.03
Epoch	Training Loss (Blue)	Validation Loss (Green)																																																																																	
0	0.078	0.05																																																																																	
1	0.038	0.042																																																																																	
2	0.035	0.04																																																																																	
3	0.032	0.04																																																																																	
4	0.032	0.04																																																																																	
5	0.03	0.038																																																																																	
6	0.03	0.038																																																																																	
7	0.03	0.035																																																																																	
8	0.025	0.035																																																																																	
9	0.022	0.032																																																																																	
10	0.022	0.032																																																																																	
11	0.022	0.03																																																																																	
12	0.02	0.03																																																																																	
13	0.02	0.028																																																																																	
14	0.02	0.028																																																																																	
15	0.02	0.028																																																																																	
16	0.02	0.03																																																																																	
17	0.02	0.03																																																																																	
18	0.02	0.03																																																																																	
19	0.018	0.03																																																																																	
20	0.018	0.03																																																																																	
21	0.018	0.03																																																																																	
22	0.018	0.03																																																																																	
23	0.018	0.03																																																																																	
24	0.018	0.03																																																																																	
25	0.018	0.03																																																																																	
Adamax	Mean absolute percentage error	 <p>Model loss</p> <table border="1"><thead><tr><th>Epoch</th><th>Training Loss (Blue)</th><th>Validation Loss (Green)</th></tr></thead><tbody><tr><td>0</td><td>60</td><td>22</td></tr><tr><td>1</td><td>20</td><td>15</td></tr><tr><td>2</td><td>12</td><td>12</td></tr><tr><td>3</td><td>10</td><td>15</td></tr><tr><td>4</td><td>10</td><td>12</td></tr><tr><td>5</td><td>8</td><td>12</td></tr><tr><td>6</td><td>8</td><td>12</td></tr><tr><td>7</td><td>7</td><td>10</td></tr><tr><td>8</td><td>7</td><td>13</td></tr><tr><td>9</td><td>7</td><td>10</td></tr><tr><td>10</td><td>7</td><td>10</td></tr><tr><td>11</td><td>7</td><td>9</td></tr><tr><td>12</td><td>6</td><td>9</td></tr><tr><td>13</td><td>6</td><td>9</td></tr><tr><td>14</td><td>6</td><td>9</td></tr><tr><td>15</td><td>6</td><td>9</td></tr><tr><td>16</td><td>6</td><td>9</td></tr><tr><td>17</td><td>6</td><td>9</td></tr><tr><td>18</td><td>6</td><td>8</td></tr><tr><td>19</td><td>6</td><td>8</td></tr><tr><td>20</td><td>6</td><td>8</td></tr><tr><td>21</td><td>6</td><td>8</td></tr><tr><td>22</td><td>6</td><td>9</td></tr><tr><td>23</td><td>6</td><td>8</td></tr><tr><td>24</td><td>6</td><td>9</td></tr><tr><td>25</td><td>6</td><td>10</td></tr></tbody></table>	Epoch	Training Loss (Blue)	Validation Loss (Green)	0	60	22	1	20	15	2	12	12	3	10	15	4	10	12	5	8	12	6	8	12	7	7	10	8	7	13	9	7	10	10	7	10	11	7	9	12	6	9	13	6	9	14	6	9	15	6	9	16	6	9	17	6	9	18	6	8	19	6	8	20	6	8	21	6	8	22	6	9	23	6	8	24	6	9	25	6	10
Epoch	Training Loss (Blue)	Validation Loss (Green)																																																																																	
0	60	22																																																																																	
1	20	15																																																																																	
2	12	12																																																																																	
3	10	15																																																																																	
4	10	12																																																																																	
5	8	12																																																																																	
6	8	12																																																																																	
7	7	10																																																																																	
8	7	13																																																																																	
9	7	10																																																																																	
10	7	10																																																																																	
11	7	9																																																																																	
12	6	9																																																																																	
13	6	9																																																																																	
14	6	9																																																																																	
15	6	9																																																																																	
16	6	9																																																																																	
17	6	9																																																																																	
18	6	8																																																																																	
19	6	8																																																																																	
20	6	8																																																																																	
21	6	8																																																																																	
22	6	9																																																																																	
23	6	8																																																																																	
24	6	9																																																																																	
25	6	10																																																																																	

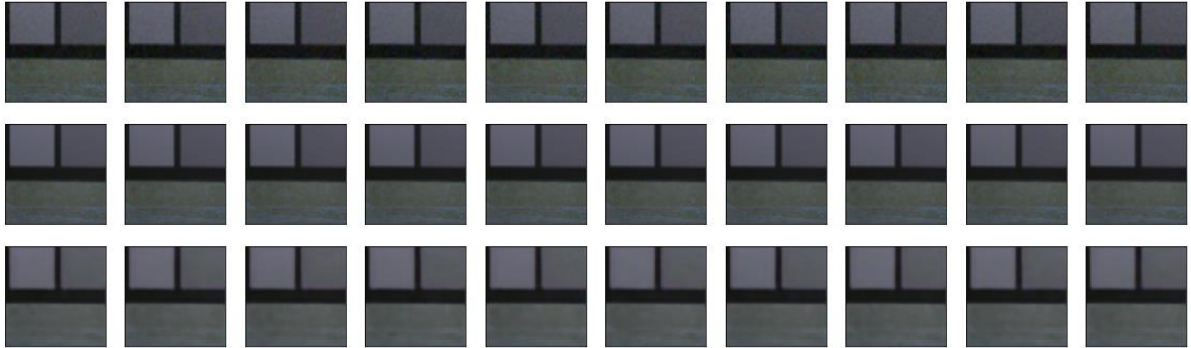
Rmsprop	Mean squared error	 <p>Model loss</p> <p>Loss</p> <p>Epoch</p>
Sgd	Logcosh	 <p>Model loss</p> <p>Loss</p> <p>Epoch</p>
Sgd	Mean squared error	 <p>Model loss</p> <p>Loss</p> <p>Epoch</p>

Out of all the results generated by the individual models, we have chosen the model with 'Adadelata' optimizer and 'Mean squared error' loss function since that gives us the best results.

#### 4.1.3. Training Process

The model was trained on 7200 samples and tested on 1800 samples. The training was done for 25 epochs for batch sizes of 20 with 'adadelta' optimizer and 'mean squared error' loss function. We chose this combination of optimizer and loss function since this gave better results as compared to other combinations of optimizers and loss functions.

The following graph was predicted by the model after the training process:



The first row consists of the noisy image, the second row consists of the ground truth image and the third row consists of the image predicted by the model.

#### 4.2. Whole Image Result

The autoencoder trained on cropped images has learned about the noise patterns encountered in the given scene. The training was done on cropped images and now it will be tested on a full-sized image. To ensure the image is read correctly by the model, we resize the image closer to 256x256, which is the input shape for the autoencoder, while maintaining the aspect ratio. The image is resized to 342x256 and then cropped 256x256 to ensure most of the image is considered for denoising while fitting the requirements of the autoencoder as well as maintaining aspect ratio. We see that image though blurry is smoothed all over compared to the grainy image which it was before input.





Here we see on the left is the grainy noisy image received as input for the autoencoder and on the right is the smoothed albeit blurry than what we expect output from the autoencoder.

## 5. Conclusion

Using the above methodology, we have compared all the possible variations to find a solution which performs best denoising of the image. We can further work on this model to ensure the image enhancement takes place as deterioration is caused by information loss while removal of noise. While machine learning algorithms are great tools for this task, there can be many more options which can be explored to make it better

## 6. References

<https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2>

<https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/denoising.html>

<http://papers.nips.cc/paper/6172-image-restoration-using-very-deep-convolutional-encoder-decoder-networks-with-symmetric-skip-connections.pdf>