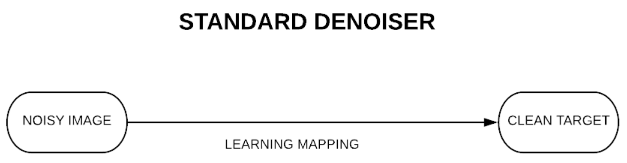
NOISE2NOISE: LEARNING IMAGE RESTORATION WITHOUT CLEAN DATA

Final Report: Abhinav Singh

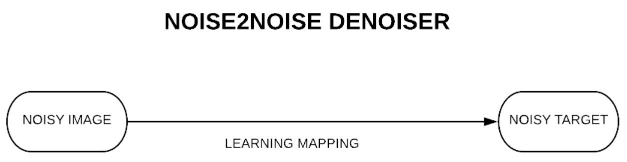
I. INTRODUCTION

We attempt to reconstruct a clean image from noisy images using machine learning techniques. It uses Convolutional Neural Networks in particular to achieve this and describes various approaches attempted by the authors in order to do this. It focuses on training a Denoiser with pairs of noisy images to predict an image, which is as good as a clean image. The current problem is that the clean targets which are necessary for training a denoiser, are difficult to obtain. For example, images taken from the Hubble telescope are too costly, as they require shutter to be opened for prolonged periods of time to get clean target images, but long exposures are not cost efficient. So, this paper focuses on corrupt or noisy images as well as noisy targets, and exploits how it can be prolific and cost efficient, with or without clean images, using algorithms like U-Net with skip connections.

In Convolutional Neural Networks, the success rate is more for simpler images, but it doesn’t give good results for complex images like Astrophotography, which is abundant with Noisy H-alpha spectrum. In CNNs the images is converted into vectors and mostly used for classification problems. On the other hand, if a model has a requirement for individual pixel classification of the image, it can be solved with U-Net. It converts image into vector and then convert it back into an image, which preserves the real structure of the image. For example, corrupt Image with randomly positioned and oriented strings just pasted on top of it, is fixed by using U-Net Structure, which learns in a supervised fashion to detect where the strings are and then removes them.



In the original implementation, Standard Denoiser is not being used, but instead of taking clean targets, Noisy targets are taken into account, using U-Net with skip connections on the corrupt targets.



II. OUR APPROACH TO THE PROBLEM

Original Implementation is on Tensorflow, with access of extremely powerful GPU to train high resolution **KODAK** image set, but we are using Pytorch framework with Pillow imaging library. As we didn't have a cuda for parallel processing, The project is implemented in **jupyter notebook** on **Google collab** to have a proper GPU access.

In implementation, we have used both **LeakyReLU** as well as **ELU** activation function separately. The original implementation used only LeakyReLU.

III. Data

Original dataset used in this research is the KODAK dataset available at http://r0k.us/graphics/kodak/ for unrestricted usage in research, but any dataset would work as image is getting trained by its noisy image. So we have used COCO 2017 dataset which is available at http://images.cocodataset.org/zips/val2017.zip

IV. Implementation

Load and train and test the Image datasets: using Torchvision:

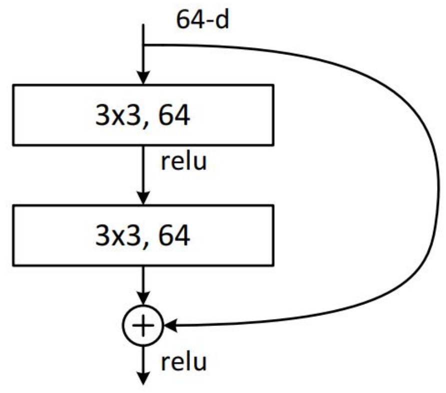
We are loading the Coco2017 dataset which are clean images, then cropping the images randomly for specific dimension. As we want noisy images for training, we apply following noises on the dataset:

a. Gaussian noise.
b. Shot noise or Poisson noise.
c. Multiplicative bernoulli noise.

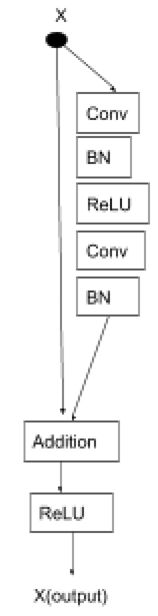
The training should be noisy dataset to noisy dataset. Therefore, Gaussian noise is added defaultly on each image.

Defining a Convolutional neural network model:

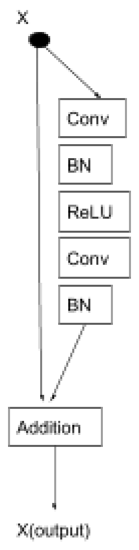
We are using CNN, and implemented model with Residual block to prevent the problem of vanishing gradient. ResNet with Skip connections has been used to have a proper gradient.



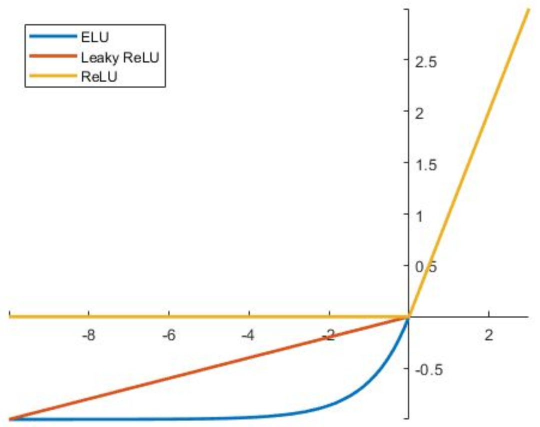
In modelling CNN, Originally RED30 was used, but we are using **SRResnet** network which is derived from **Super Resolution GAN** i.e. **SRGAN** is used to produce high resolution images and SResnet instead of using ReLU, it uses Parametric ReLU. The paper (https://arxiv.org/abs/1609.04802) extends SRResnet as part of SRGAN architecture and results that are obtained from it, have high frequency features like fur of animals, but it was observed that PSNR data was not adequate. Further BN(Batch Norm) that is used for removing overfitting problems, but slows down the training speed, There is a **modified SRResnet** implementation which doesn't use batch norm, and showed better results regarding SRResnet, but we are using SRResnet in the experimentation with kernel size=3.



(a)ResNet



(b)SRResNet



Training Details:

For activation of each neuron, the original implementation used **LeakyReLU** activation function. But we tried both LeakyReLU as well as another variant of ReLU activation function i.e. **ELU**, as dying ReLU problem doesn't occur in **ELU**.

In training procedure, we have used **ADAM**, an adaptive learning rate optimization algorithm, which increases the learning rate in parameters we have set its learning rate to 0.001.

Defining a loss function:

In our model, L2 loss function is used i.e. Mean Squared Error function. But option to use L1 loss function is given.

$$L2LossFunction = \sum_{i=1}^{n}(y_{true} - y_{predicted})^2$$

Result:

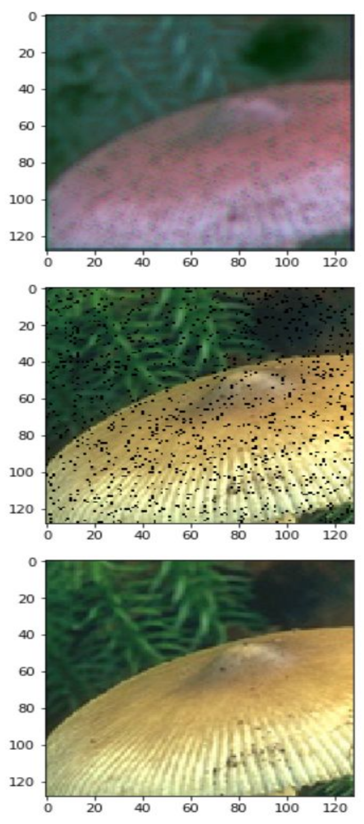
Last Report Results:

In each case where we applied **ELU** activation function, the training loss and validation loss had less average loss values compared to when we applied **LeakyReLU** on the model.

Result Samples:

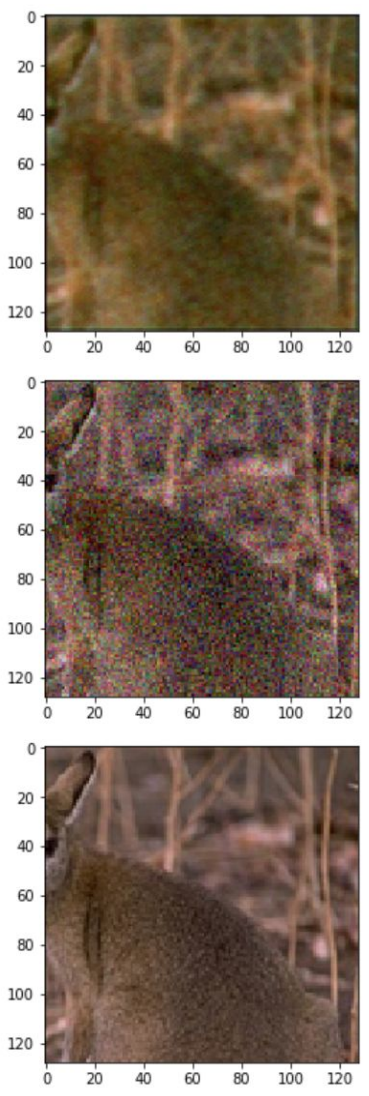
1. **When multiplicative bernoulli noise was applied on the model:**

```
Training loss = 0.7625266313552856,
Validation loss = 0.22174744307994843
```



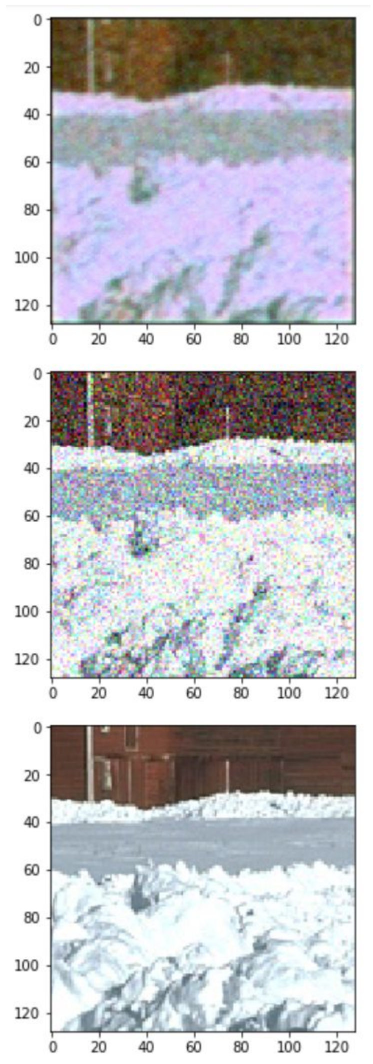
2. **When Gaussian noise was applied:**

```
Training loss = 0.2199152261018753,
Validation loss = 0.03344931825995445
```



3. **When Poisson noise was applied:**

```
Training loss = 0.21640892326831818,
Validation loss = 0.09912391006946564
```

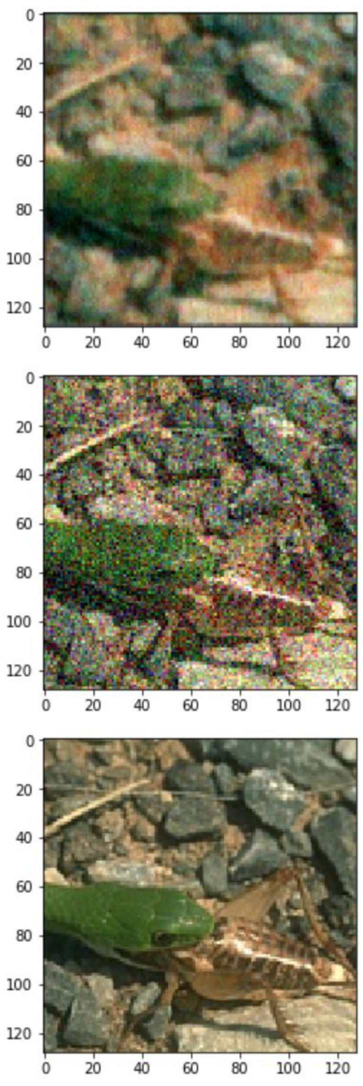


,

Results when ELU activation function was applied:

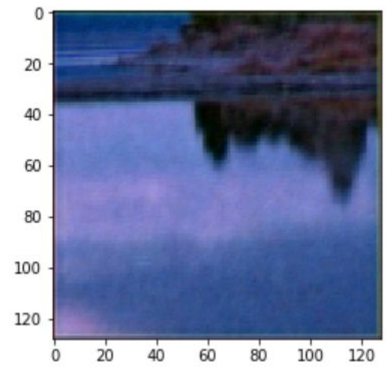
1. **When multiplicative bernoulli noise was applied on the model:**

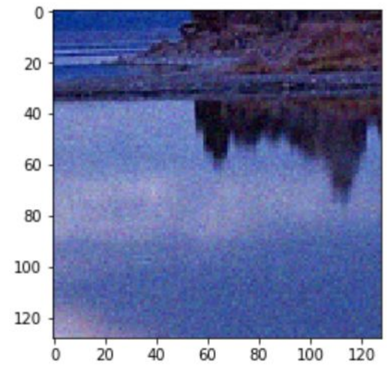
```
Training loss = 0.37333303689956665,
Validation loss = 0.04607954993844032
```

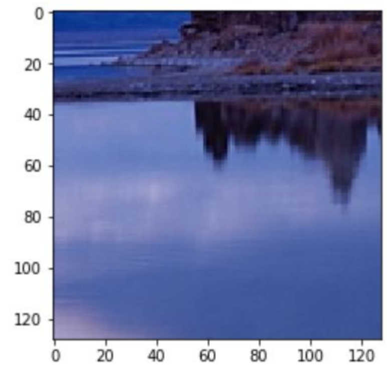


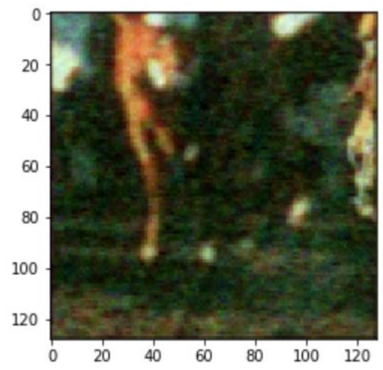
2. **When Gaussian noise was applied:**

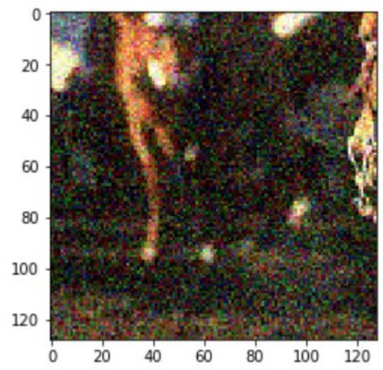
```
Training   loss   =   0.19688072800636292,
Validation loss = 0.02959379367530346
```

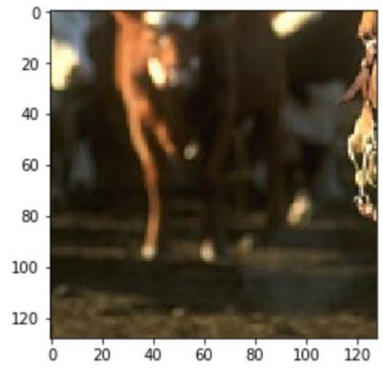












3. **When Poisson noise was applied:**

```
Training   loss   =   0.1934657245874405,
Validation loss = 0.05446384474635124
```

New Results:

Test for Gaussian and PSNR:

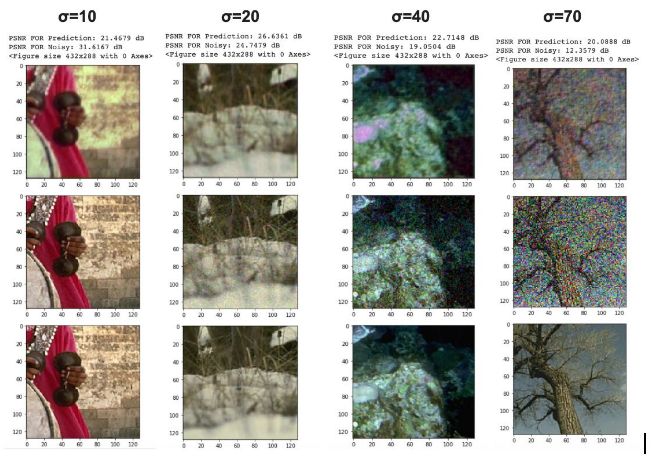
Test encompasses checking of PSNR values for both **Noisy image** and **Predicted image** with respect to Gaussian noise, here **σ** represents noise standard deviation and **ELU** activation function is used in all layers.

PSNR is **Peak signal to noise ratio**, which represents a measure of peak error.

$$PSNR = 10log_{10}(\frac{R^2}{MSE})$$

R: represents maximum pixel value

MSE: represents Mean Squared Error



```
PSNR FOR Prediction: 21.4679 dB
PSNR FOR Noisy: 31.6167 dB
<Figure size 432x288 with 0 Axes>
```

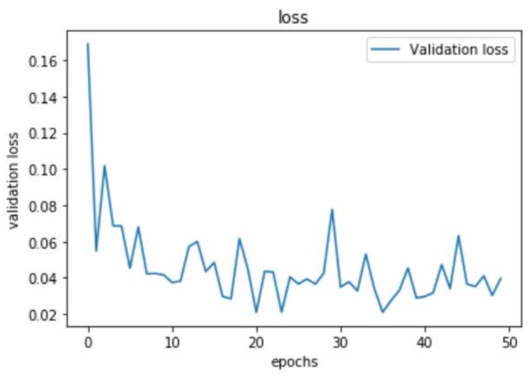
```
PSNR FOR Prediction: 26.6361 dB
PSNR FOR Noisy: 24.7479 dB
<Figure size 432x288 with 0 Axes>
```

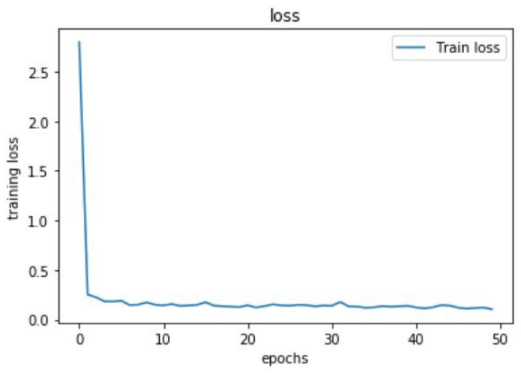
```
PSNR FOR Prediction: 22.7148 dB
PSNR FOR Noisy: 19.0504 dB
<Figure size 432x288 with 0 Axes>
```

```
PSNR FOR Prediction: 20.0888 dB
PSNR FOR Noisy: 12.3579 dB
<Figure size 432x288 with 0 Axes>
```

The network outputs decent results,considering training is done by using noisy image only and here top image being the prediction, mid one is noisy one and bottom one is the ground truth.

Both validation loss and training loss are done on for 50 epochs.





IV.LIMITATIONS AND IMPROVEMENTS

1. The data is comparatively smaller with only **300 training** and **100 validation** image dataset, to come up with proper and concrete results.
2. Modified SRResnet can be used, without using Back Norm, which is proven in paper **Enhanced Deep Residual Networks for SingleImage-Super-Resolution**(https://arxiv.org/abs/1707.02921)
3. Single noisy image can be used to train the network, according to the paper **Noise2Void**(https://arxiv.org/pdf/1803.04189).
4. The properly additing of Poisson noise is really is arduous task, as Poisson noise depends on the data, basically it is non-additive.

V:REFERENCES

- [1] Lehtinen, Jaakko, et al. Noise2Noise: Learning Image Restoration without CleanData.(https://arxiv.org/abs/1803.04189)
- [2]Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network(https://arxiv.org/abs/1609.04802)
- [3]Enhanced Deep Residual Networks for SingleImage-Super-Resolution(https://arxiv.org/abs/1707.02921)
-