SMAI PROJECT

1. BANALA VISHWA TEJA REDDY-2020102058.
2. NANDITHA MERUGU-2020102061
3. NITHIN KONDURU-2020101104.
4. HARSHA VARDHAN THATIPAMULA-2020101106.

TOPIC : Neural Nearest Neighbors Networks

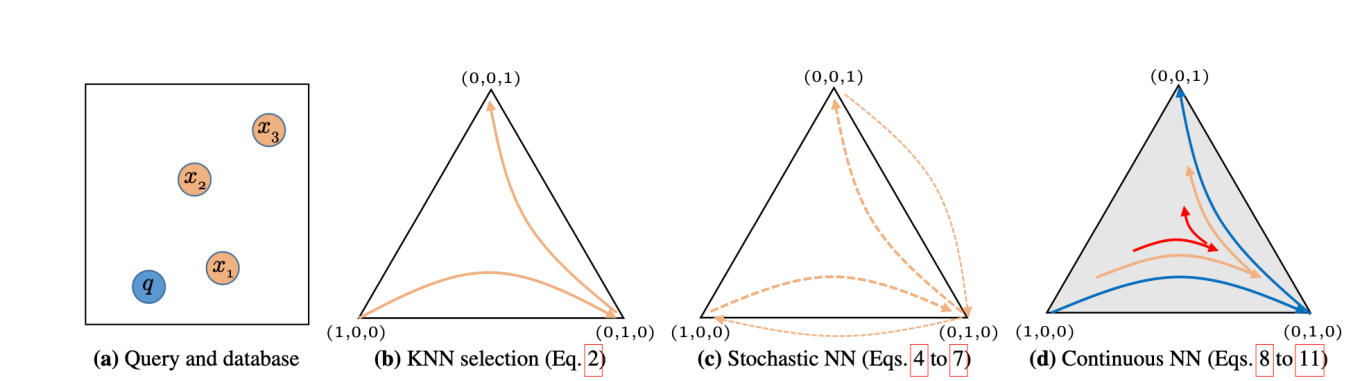
INTRODUCTION

By providing previously unheard-of predictive accuracy, convolutional neural networks (CNNs) have transformed numerous aspects of machine learning and its applications. The majority of network topologies combine convolutional layers and element-wise operations to focus on local processing. Numerous techniques, such as dilated convolutions or hourglass-shaped architectures , have been investigated to expand the receptive field size in order to draw information from a sufficiently wide context. However, they compromise localization accuracy for context size. In order to acquire larger receptive fields, stacking even more convolutional blocks has thus remained the preferred method for many dense prediction tasks, such as image analysis and restoration.

The self-similarity of natural signals is used by conventional picture restoration methods to enhance the receptive field size through non-local processing. They take advantage of the fact that image structures frequently appear in the same image, creating a strong prior for image restoration. As a result, techniques like non-local means or BM3D gather data from the entire image to restore a local patch. In this case, matching patches are typically chosen based on some manually created idea of similarity, such as the Euclidean distance between input intensity patches. It has only recently been thought to incorporate this kind of non-local processing into neural network architectures for image restoration.

In KNN selection Rule, we tend to find the label for a query based on the maximum number of the points that belongs to some class but if the dataset is of maximum size then the finding the best label for the query is not optimal. In general, the metrics we consider for KNN are distance metrics, weighted etc. Those methods are proven low optimized. Thus we need to optimize the feature space with respect to the application performance. This paper proposed a continuous deterministic relaxation of KNN selection
that maintains differentiability with respect to the pairwise distances.The parameter we consider here is the 't' the temperature parameter for differentiability.

BRIEF INTRODUCTION TO THE ALGORITHM IN THE PAPER



(a) Query and database **(b)** KNN selection (Eq. 2) **(c)** Stochastic NN (Eqs. 4 to 7) **(d)** Continuous NN (Eqs. 8 to 11)

KNN Selection algorithm, here we need to find the label for the query by considering the k nearest datapoints from the dataset.Consider the given example: Hereqindicatedthequeryandwhereasx1,X2andx3arethe datapoints. In KNN, we only consider the locality as we only focus on the distance from the Query to the certain datapoint xi. Stochastic Neighbour Selection, here we tend to find the probability of xi being the Neighbour of q with respect to t where ' t' is the temperature parameter. Continuous Neighbour Selection, here we tend to find the probability of xi being the

Neighbour of xj with respect to t where ' t' is the temperature parameter. Depending on the parameter 't', this path can interpolate between a more uniform weighting and the original KNN selection.

1-NN STOCHASTIC NEAREST NEIGHBOURS

We continue by interpreting the KNN selection criterion as the limit distribution of the following k categorical distributions. Let w (index vector) be the nearest Neighbour of the 'q'. Say if w1 = i , that means i-th entry set to one while the others are zeros. The expressions and distrubitions will be looking like below

$$\mathbb{P}\left[w^1 = i \mid \alpha^1, t\right] \equiv \mathrm{Cat}(\alpha^1, t) = \frac{\exp\left(\alpha_i^1/t\right)}{\sum_{i' \in I} \exp\left(\alpha_{i'}^1/t\right)} \tag{3}$$

$$\text{where} \quad \alpha_i^1 \equiv -d(q, x_i). \tag{4}$$

With w1 = I we indicate that w1 is being used as a one-hot coded vector in this case, with the i-th item being set to one and the others being zero. Cat(w1 | α1,t) will converge to a deterministic ("Dirac delta") distribution with a centre at the index of the database item that is closest to q in the case when t 0. So, sampling from Cat(w1 | α1,t) can be viewed as a stochastic relaxation of 1-NN .

K-NN STOCHASTIC NEAREST NEIGHBOURS(EXTENSION OF 1-NN)

The above is similar to 1-NN. So, we generalize this to k points and do the above process iteratively. I.e,By putting forth an iterative method to create more conditional distributions Cat(wj+1 | αj+1, t), we now generalise this to arbitrary k. To make sure that this index cannot be sampled again, we compute αj+1 by changing the wj -th entry of j to negative infinity:

$$\alpha_i^{j+1} \equiv \alpha_i^j + \log(1 - w_i^j) = \begin{cases} \alpha_i^j, & \text{if } w^j \neq i \\ -\infty, & \text{if } w^j = i. \end{cases}$$

The updated logits are used to define a new categorical distribution for the next index to be sampled:

$$\mathbb{P}\left[w^{j+1} = i \mid \alpha^{j+1}, t\right] \equiv \mathrm{Cat}(\alpha^{j+1}, t) = \frac{\exp\left(\alpha_i^{j+1}/t\right)}{\sum_{i' \in I} \exp\left(\alpha_{i'}^{j+1}/t\right)}.$$

From the index vectors wj, we can define the *stochastic nearest neighbors* {X1,...,Xk} of q using

$$X^j \equiv \sum_{i \epsilon I} w_i^j . x_i$$

When the temperature parameter t approaches zero, the distribution over the {X1,...,Xk} Will be a deterministic distribution centered on the k nearest Neighbours of q. In Continuous deterministic relaxation, the basic idea is to replace the one-hot coded weight vectors with their continuous expectations.

CONTINUOUS DETERMINISTIC RELAXATION

Our basic idea is to replace the one-hot coded weight vectors with their continuous expectations. This will yield a deterministic and continuous relaxation of the stochastic nearest neighbors that still converges to the hard KNN selection rule in the limit case of t → 0. Concretely, the expectation w¯1 of the first index vector w1 is given by

$$\bar{w}_i^1 \equiv \mathbb{E}\left[w_i^1 \mid \alpha^1, t\right] = \mathbb{P}\left[w^1 = i \mid \alpha^1, t\right].$$

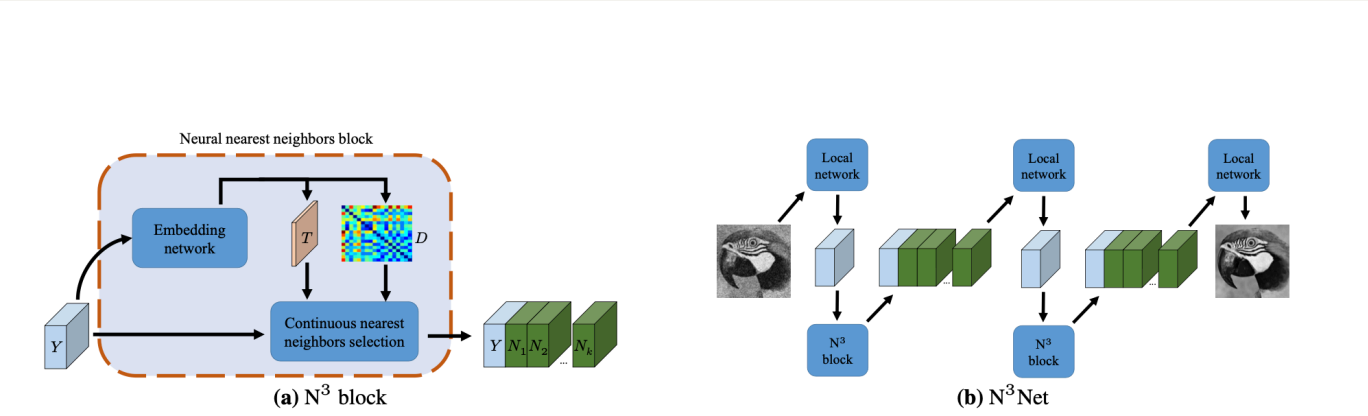
$$\bar{\alpha}_i^{j+1} \equiv \bar{\alpha}_i^j + \log(1 - \bar{w}_i^j) \quad \text{with} \quad \bar{\alpha}_i^1 \equiv \alpha_i^1.$$

$$\bar{w}_i^{j+1} \equiv \mathbb{E}\left[w_i^{j+1} \mid \bar{\alpha}^{j+1}, t\right] = \mathbb{P}\left[w^{j+1} = i \mid \bar{\alpha}^{j+1}, t\right].$$

we define *continuous nearest neighbors* {X¯1,...,X¯k} of q using the w¯j as

$$\vec{X}^j \equiv \sum_{i \epsilon I} \vec{w_i^j}. x_i$$

NNN-Nearest Neural Neighbour Block



First, an embedding network takes the input and produces a feature embedding as well as temperature parameters. These are used in a second step to compute continuous nearest neighbors feature volumes that are aggregated With the input. Y is the image data. We are passing Y into Embedding Network. In Embedding Network, we need to compress the image data into the embeddings based on features on the image data. This is done using CNN (Convolution Neural Network). So, $E = f_E(Y)$ Then we need to compute Tensor $T = f_T(Y)$ And also by using the pairwise distance matrix D and the above calculated T in the mathematical equations that we came across in the previous slide for k nearest Neighbours. The temperature parameter t is the corresponding center pixel in T. We do the previous step under Continuous Nearest Neighbours selection. From this step we get k continuous nearest Neighbours volumes N1, N2, ..., Nk. Finally we concatenate these to the input Y for better pixel. We take the Average of those into the New Y and keep on passing will sharpen the figure taken if any noise is included the noise will be removed.

EXPERIMENTS

We now analyze the properties of our novel N3Net and show its benefits over state-of-the-art baselines. We use image denoising as our main test bed as non-local methods have been well studied there. Moreover, we evaluate on single image super-resolution and correspondence classification.

GAUSIAN DENOISING

We consider the task of denoising a noisy image D, which arises by corrupting a clean image C with additive white Gaussian noise of standard deviation σ: D=C+N with N~N(0,σ2).Our baseline architecture is the DnCNN model of Zhang *et al.* consisting of 16 blocks, each with a sequence of a 3 × 3 convolutional layer with 64 feature maps, batch normalization , and a ReLU activation function. In the end, a final 3 × 3 convolution is applied, the output of which is added back to the input through a global skip connection.

We use the DnCNN architecture to create our N3Net for image denoising. Specifically, we use three DnCNNs with six blocks each, Fig above at N3 net .. The first two blocks output 8 feature maps, which are fed into a subsequent N3 block that computes 7 neighbor volumes. The concatenated output again has a depth of 64 feature channels, matching the depth of the other intermediate blocks. The N3 blocks extract 10 × 10 patches with a stride of 5. Patches are matched to other patches in a 80 × 80 region, yielding a total of 224 candidate patches for matching each query patch. More details on the architecture can be found in the supplemental material.

	Model	Matching on	PSNR [dB]	SSIM
(i)	1 × DnCNN (d=17)	–	29.97	0.879
(ii)	1 × DnCNN (d=18)	–	29.92	0.885
(iii)	3 × DnCNN (d=6), KNN block (k=7)	noisy input	30.07	0.891
(iv)	3 × DnCNN (d=6), KNN block (k=7)	DnCNN output (d=17)	30.08	0.890
(v)	3 × DnCNN (d=6), Concrete block (k=7)	learned embedding	29.97	0.889
(ours light)	2 × DnCNN (d=6), N^3 block (k=7)	learned embedding	29.99	0.888
(ours full)	3 × DnCNN (d=6), N^3 block (k=7)	learned embedding	**30.19**	**0.892**

The above one is taken from the paper for variance (sigma=25) and we should learn and train embedings and get the PSNR in dB and SSIM for the above results of the parrot image we got in the datasets.

CONCLUSIONS

Non-local approaches have received extensive research, for instance in image restoration. However, current methods use KNN selection on a manually defined feature space, which might not be the best strategy for the problem at hand. We developed the first continuous relaxation of the KNN selection rule that preserves differentiability with respect to the pairwise distances utilised for neighbour selection in order to get around this restriction. In order to create a unique network block known as the N3 block, which serves as a general building block for neural networks, we integrated continuous closest neighbours selection. In the context of image denoising, SISR, and correspondence classification, where we outperform cutting-edge CNN-based techniques and non-local approaches, we provided examples of its benefit.We anticipate that end-to-end trainable designs for other input domains, such as text or other sequence-valued data, will also gain from the N3 block.

CODES AND FUNCTIONS USED IN CODE

---

def cnn() - This is the cnn processing required in the embedding network. Variable ksize shows the size of the kernel. The padding determines the variables excluded from the edges of the image cnn_bn is the batch normalisation flag. This emirates for the range of (cnn_depth -1) to extend cnn_layers from nn.Conv2d If the depth of CNN is greater than 0 our function will append nn.conv2d to it, Lastly, it will return the variable net which is just the CNN layers in sequential order.

Class NeuralNearestNieghbourBlock() - This is the implementation of the NeuralNearestNieghbourBlockIt defines two things: Embedding network and Continuous KNNIt requires nn.Module as input.First it defines the embedcnn function using cnn(), then finds the T tensor and D matrix. Secondly, it uses KNN (from non_local.py) to get the final output of NeuralNearestNieghbourBlock.The structure as given in image above is defined in forward() function.

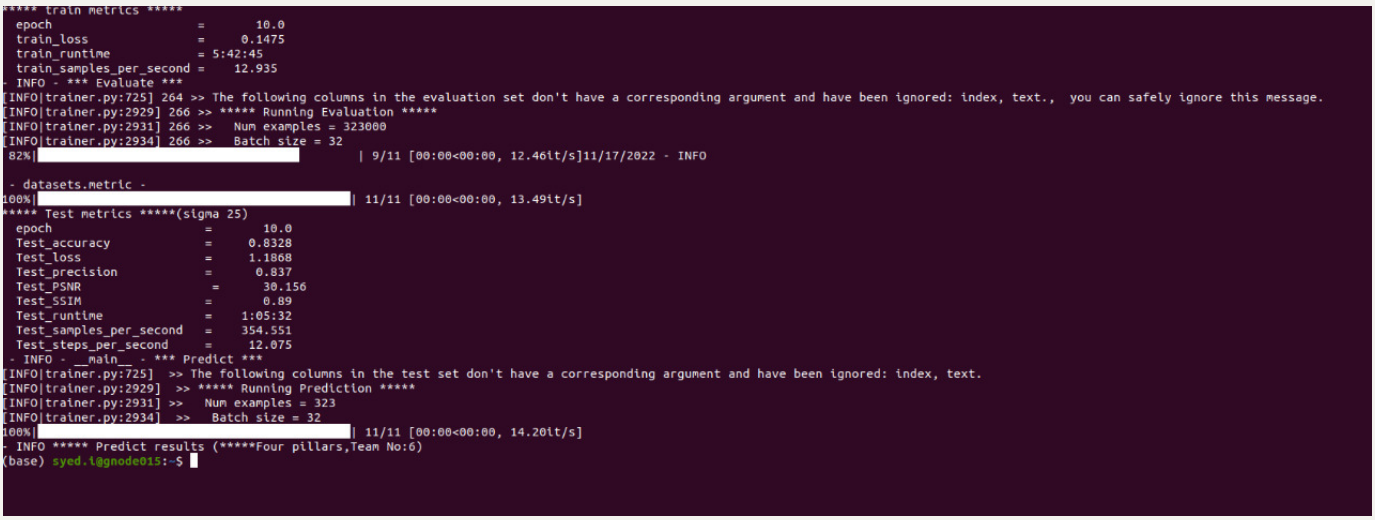
Class DnCNN() - This is the implementation of Local network using DnCNN architecture.It is a standard and effective architectureIt mainly consists of the hourglass-like structure in the order from left to right as : *conv - relu - layers - conv*. This is defined in forward().

Class NeuralNearest Nieghbour Net() - This is the final class that binds all the substructures and blocks together to finally get the main net.Takes *nblocks* as input, and defines *nblocks* numbers of DnCNN and NeuralNearestNieghbourBlock in sequence as per the diagram given above.

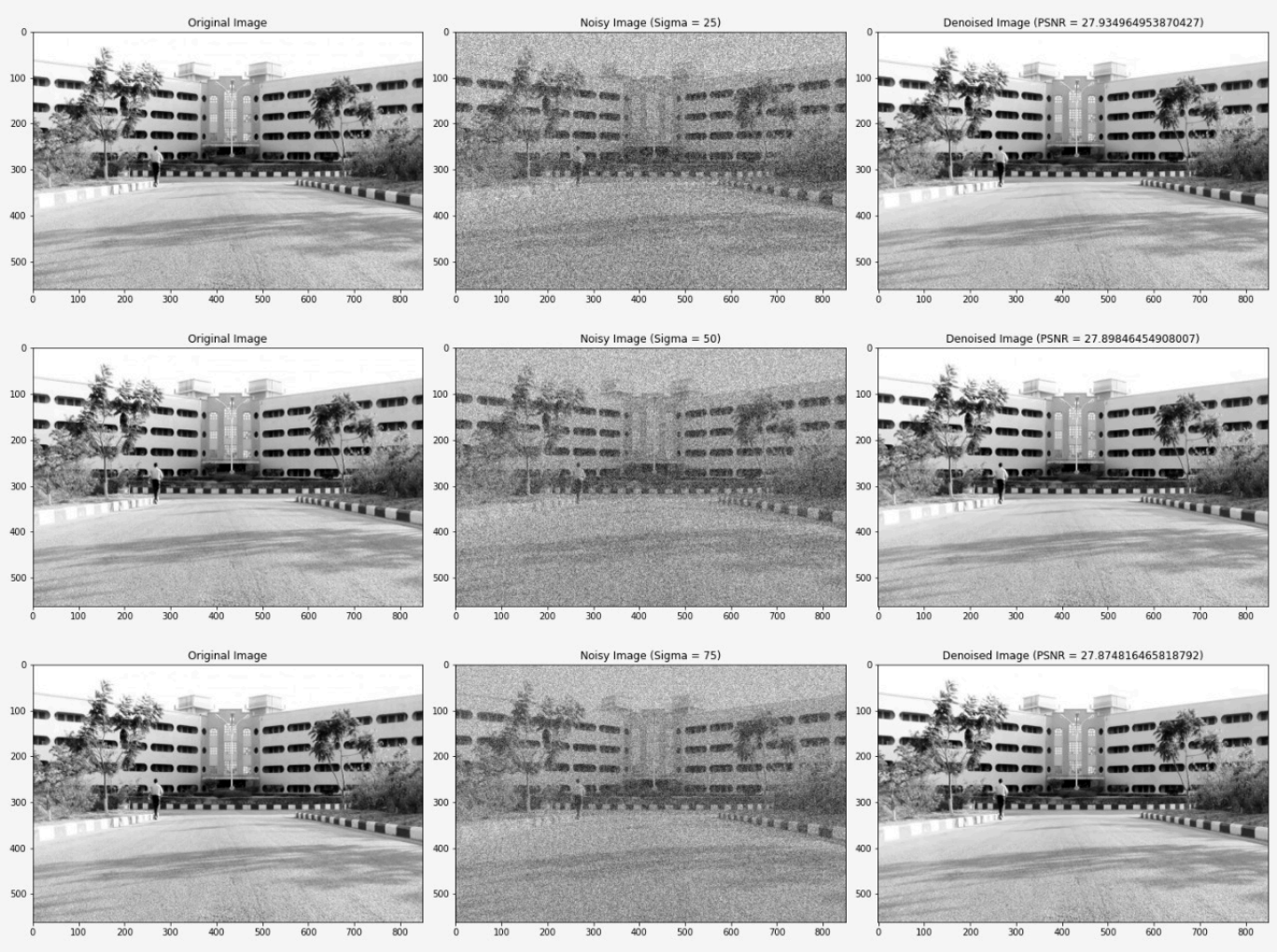
RESULTS IN DENOISING THE IMAGES

DENOISING IMAGES ON ADDING GAUSIAN NOISE TO THE PICTURES

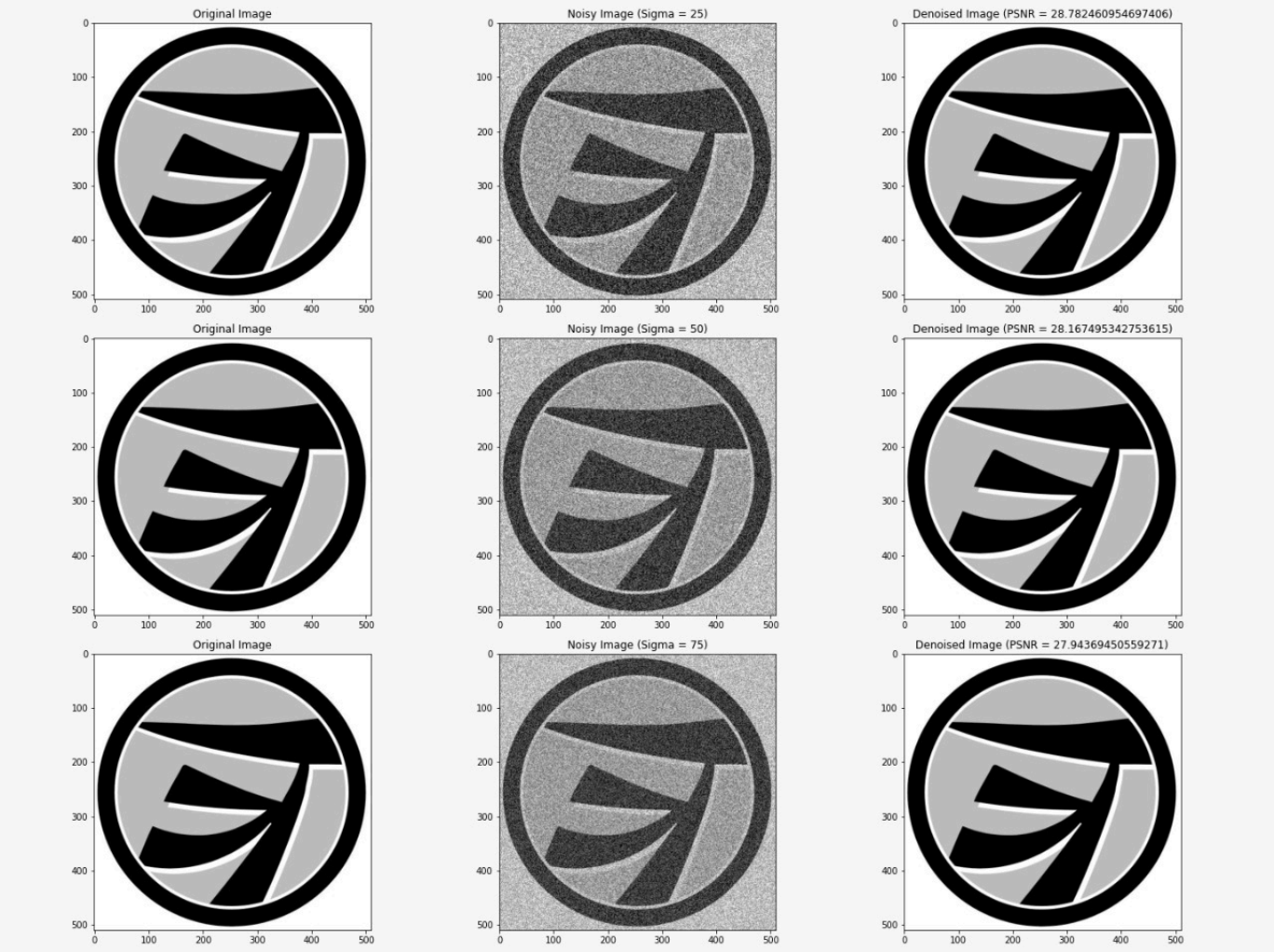
TERMINAL SNIPPET



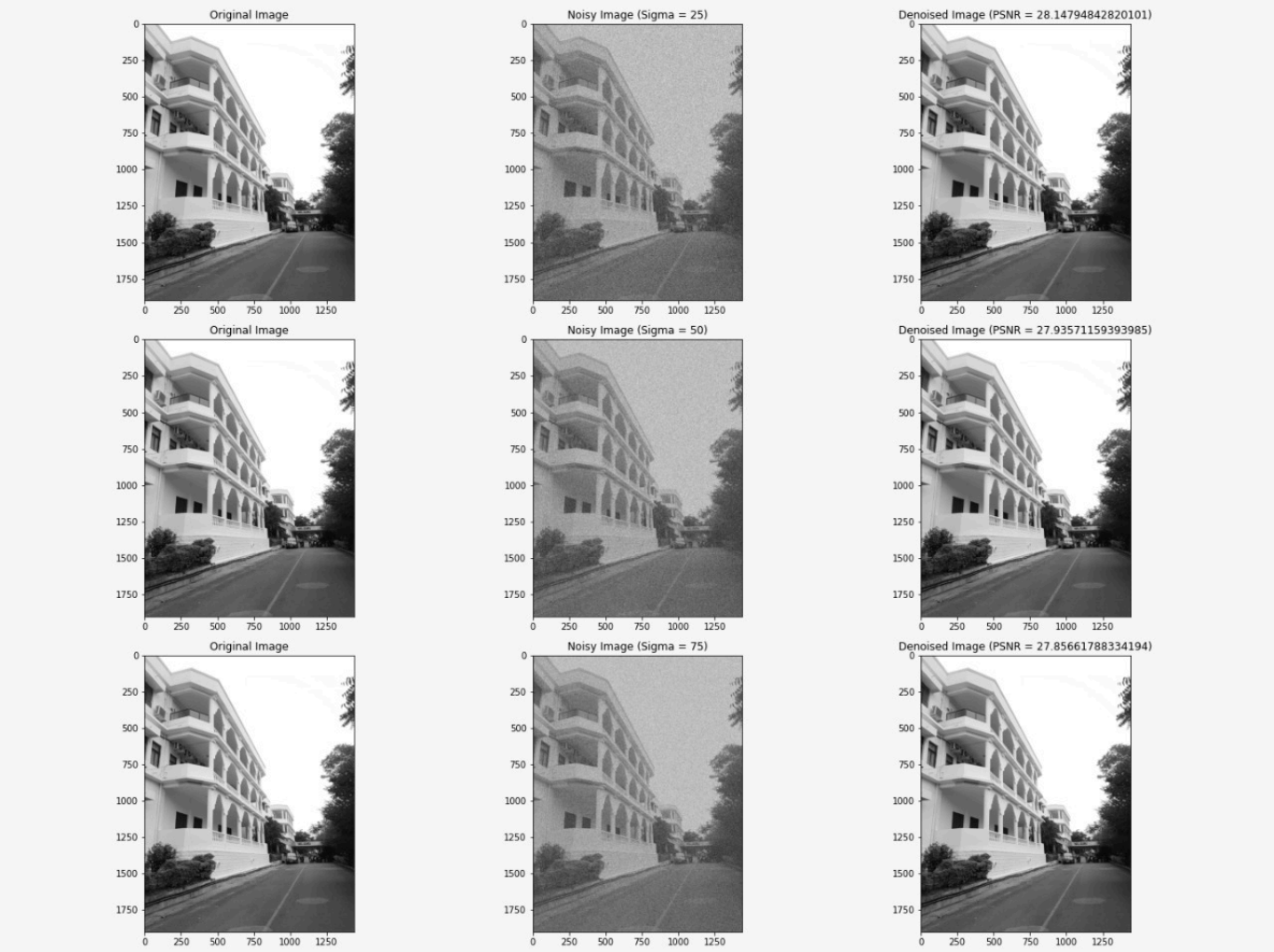
EXP1


Original Image
Noisy Image (Sigma = 25)
Denoised Image (PSNR = 27.934964953870427)
Original Image
Noisy Image (Sigma = 50)
Denoised Image (PSNR = 27.89846454908007)
Original Image
Noisy Image (Sigma = 75)
Denoised Image (PSNR = 27.874816465818792)

EXP2


Original Image
Noisy Image (Sigma = 25)
Denoised Image (PSNR = 28.782460954697406)
Original Image
Noisy Image (Sigma = 50)
Denoised Image (PSNR = 28.167495342753615)
Original Image
Noisy Image (Sigma = 75)
Denoised Image (PSNR = 27.9436945055927 1)

EXP3


Original Image
Noisy Image (Sigma = 25)
Denoised Image (PSNR = 28.14794842820101)
Original Image
Noisy Image (Sigma = 50)
Denoised Image (PSNR = 27.93571159393985)
Original Image
Noisy Image (Sigma = 75)
Denoised Image (PSNR = 27.85661788334194)