Improving Neural Net performance by preprocessing input using J-invariant functions

***Abstract*—Neural networks(NNs) are one of the most important and useful tools in the current era of science. They have proven to achieve outstanding performance in various applications like pattern recognition and classification, forecasting, fraud detection, etc. NNs have a lot of advantages and disadvantages. One specific disadvantage in the case of a classification problem is that the classifier's performance deteriorates largely with image quality ie. if we provide an NN with noisy images, it fails to classify them correctly. In this research work, we try to overcome this shortcoming by preprocessing the low-quality images before classification. We denoise the images using a special kind of denoiser called the j-invariant denoiser which is followed by a NN based classifier. The proposed method finds its usefulness in enhancing the quality of images and in turn improves the classification accuracy.**

Keywords - Image denoising;j-invariant denoisers;neural networks.

I. Introduction

Neural network architectures are inspired from the strange arrangement and interconnections of the neurons in a human brain. The neurons in a brain aggregate non-linearly the sensory inputs and trigger an output if it exceeds a particular threshold. An artificial neural network adopts this complicated biological process in a very simplified manner to create an artificial neural network model. NNs solve problems by learning from the examples that are used to train them, just like a human brain learns, unlike the usual method of programming with task-specific codes. Artificial Neural Networks(ANNs) are sets of connected neurons organized in the form of layers. The connections are called edges. The neurons in a brain are interconnected via synapses which help transmit information between them. In ANNs, this role is played by the edges. When an artificial neuron receives a signal, a real number, it processes the input and sends out an output, calculated by some non-linear function of the sum of its inputs, to other neurons connected to it. All edges typically have weights that are modified as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Multi-layered neural networks that are comprised of more than three hidden layers are generally referred to as deep neural networks. Over the years NNs have been used extensively in a variety of fields. Today, NNs are used for solving many business problems such as sales forecasting, customer research, data validation, and risk management and other applications like self-driving cars, stock market prediction, etc.

Although Neural networks are found to be efficient for tasks like image classification due to their layer-wise design to emulate latent features from data, their performance degrades considerably in the presence of noise, making them less suitable for real-life scenarios. Noise in image is a common phenomenon in many real-life scenarios for eg. taking videos at ultra-fast rates at low illumination, probing individual molecules with an electron microscope, etc. and several studies have been conducted in the previous couple of decades with the intention to overcome the effect of noise in the image data.

One of the methods is training NNs with noise injected training images. [4] discusses a detailed analysis of training the same model using images containing gaussian and salt and pepper noise. It also analyses the performance of the NN when trained with different noise levels for each kind of noise. The results obtained signify that training neural nets using data corrupted by some kinds of noise could be favourable for applications that face images of varying quality. It also improves the pliancy of the neural net to other types of noise for various noise levels.

A second method involves denoising the input images and then classifying them using the NN. In [6], the denoising step is carried out by a variety of auto-encoders namely denoising autoencoder (DAE), convolutional denoising autoencoder (CDAE) and denoising variational autoencoder (DVAE) as well as two hybrid AEs (DAE-CDAE and DVAE-CDAE). The proposed methods were able to obtain satisfactory accuracy with 50% noise when the model was trained with images containing only 20% noise. Similarly in [7], DAEs were used in combination with CNNs for noisy image classification. An array of cascaded DAEs was used for denoising and then classified using CNN to obtain considerably good accuracy in case of massive noisy data.

A supervised learning algorithm learns from labeled training data and helps us predict outcomes for unforeseen data using previous experience. Here, objects which are labeled into classes are used to train the model and new images are tested on this trained model. In unsupervised learning, the task of machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unsupervised learning has a lot of advantages over supervised learning. It takes place in real-time and does not require high computation time, unlike supervised learning. The manual labour of classifying the data into various classes is eliminated in unsupervised learning.

In this work, we use j-invariant functions(explained in section II) combined with optimisation using self-supervision [1] to denoise the images. Self-supervision uses the data itself

as the label. It has all the advantages of unsupervised learning and hence it does not require the ground truth to denoise the image. The denoised images are then classified using a convolutional neural network(CNN), since it is computationally more effective than NNs. We have observed an improvement in accuracy when the images were denoised using j-invariant denoisers than when denoised using the original denoisers.

The main contributions of the paper have been listed below:

1) We reduce the effect of noise present in data by denoising them using j-invariant functions and then using the improved images for classification tasks. The j-invariant denoiser can be applied when noise in the data is independent between sets of dimensions, conditioned on the ground truth.
2) We test the proposal for various kinds of noise like gaussian and salt and pepper noise and on benchmark datasets like MNIST etc. Traditionally used denoisers such as Nl-means, median and wavelet filters were used for the denoising purpose.
3) We compare the accuracy obtained from classifying noisy images, images denoised using traditional denoisers and images denoised using their j-invariant counterparts and arrive at various conclusions.

II. Related Work

The forthcoming subsections discuss various concepts described in [1] that we use in this paper.

A. *J-invariant function*

Definition:
If J is a partition of the dimensions of a data, then a function f(x), x$\in$ J, is called J-invariant if the value of f(x) does not depend on the value of x itself [1].

Every traditional denoising function can be adapted into a J-invariant function. [1] gives an example of a median denoising filter being converted into a J-invariant filter. A median filter denoises the image by replacing every pixel by the median of the pixels from the surrounding of radius 'r' whereas a J-invariant median filter excludes the center pixel while taking the median of the neighbouring pixels.

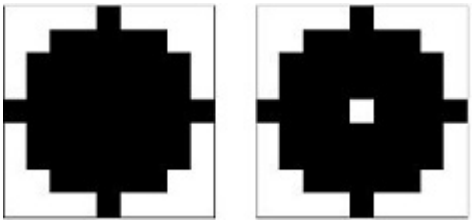

Fig. 1. **Left**: A normal median filter that replaces the centre pixel with the median of the pixels surrounding it, shown by the black pixels. **Right**: A j-invariant median filter that replaces the pixel by the median of all pixels surrounding it except the pixel itself, thus making the median independent of the pixel (Figure taken from noise2self/github).

The median filter has just one hyper-parameter, the radius of the filter. The same is the case for denoisers like wavelet filter, NL-means filter, etc. which we have used in this paper. NNs like autoencoders, variational autoencoders, etc. can also be used as denoisers. These have a large number of hyper-parameters and thus by optimising the hyper-parameters, their J-invariant versions will be able to adapt better to the image.

B. *Conversion of any denoiser into a J-invariant function*

The following method from [1] is used to convert any traditional denoising function into a j-invariant function.

Let f_θ be a classical denoising function, and consider some partition J of the pixels. Let s(x) be the average of all pixels in the neighbourhood of x. Then the function g_θ defined by

$$g_\theta(x)_J = f_\theta(\mathbf{1}_J.s(x) + \mathbf{1}_{J^c}.x)_J \tag{1}$$

where $\mathbf{1}_J$ is the indicator function of set *J*,
for each J $\in$ *J*, is *J-invariant*.

C. *Self-supervised loss*

The loss function used in [1] is the self-supervised loss.

$$L(x) = E||f(x) - x||^2 \tag{2}$$

Denoisers are functions of hyper-parameters, which can be varied. The above mentioned self-supervised loss function is minimised for different denoisers varying in their hyper-parameters to obtain the optimal denoiser ie. if we have a set of j-invariant denoisers, the optimal denoiser is the one with the hyper-parameter value that gives the least self-supervised loss. [1] gives the example of varying the radius in case of a median filter and finding the optimal filter radius that gives the minimum self-supervised loss.

$$\begin{aligned}\text{Optimal hyper-parameter}(\theta) &= argmin_\theta L(x,\theta) \\ &= argmin_\theta E||f(x,\theta) - x||^2\end{aligned} \tag{3}$$

We will consider an example of converting a denoising NN into a j-invariant denoiser. Usually, denoising NNs are trained with noisy images and their ground truth versions are considered as labels. To obtain a j-invariant denoiser, instead of this, we do the following steps for training the NN:

- We compute the outputs of the model on images created by replacing a grid of pixels with the average of their neighbors.
- We evaluate the loss by applying the loss function on the manipulated pixels of the output and the noisy image.
- This loss is then what is minimised during training.

So basically, if x is the manipulated pixels of the noisy image and f(x) is the corresponding pixels of the model's output of the noisy image, where f is the denoising NN, the loss which is minimised during training is $(f(x)-x)^2$. This is essentially the self-supervision loss. Thus by training using the above method we obtain the optimal j-invariant version of the denoising NN.

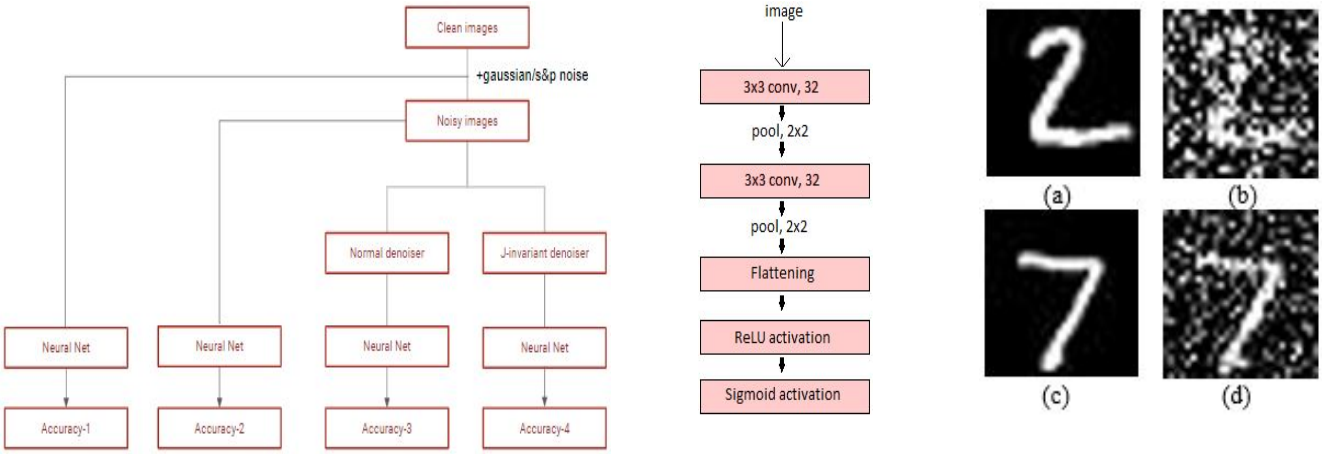

Fig. 2. **Left**:Shows the the flowchart of the process followed in the testing of the proposed idea. We obtain 4 different datasets which are then classified using the classifier trained on clean images. We would ideally obtain the following relation from the experiment, Accuracy-1>Accuracy-4>Accuracy-3>Accuracy-2. **Middle**: The figure shows the 5-layer CNN used in experiment 1. **Right**: Examples from MNIST dataset, fig. (b) is fig. (a) subjected to a 0.6 variance gaussian noise and (d) is fig. (c) subjected to a 0.4 variance gaussian noise.

III. Proposed method for denoising and classification

Our method of improving the performance of the classifier in the presence of noisy data is to initially denoise the corrupted images using j-invariant denoisers and then classify them using the classifier. We try to prove that classifiers give better accuracy when denoised with j-invariant denoisers rather than when denoised with their original counterparts. Fig. 2 shows us the flowchart of the process followed during testing of the proposed idea. The idea was tested using a variety of traditionally used denoisers like median, wavelet and nl-means filters.

The following is the algorithm followed during the experimentation of the proposed method. The algorithm calculates the accuracy of classification of the NN from the classes assigned by it to each denoised image.

Algorithm Method followed in experiment

Input: Noisy test dataset($x_1, x_2, ..., x_n$)
Output: Accuracy1, Accuracy2
Initialize i $\leftarrow$ 1
while i<=n **do**
    normal_denoised_dataset = append(f(x_i))
    j-invariant_denoised_dataset = append(g(x_i))
end while
Accuracy1 = $evaluate_n$(img_normal)
Accuracy2 = $evaluate_n$(img_j-invariant)
return Accuracy1, Accuracy2

,where f:any denoiser, g:j-invariant denoiser corresponding to f, normal_denoised_dataset:contains all images of noisy test dataset denoised using f, j-invariant_denoised_dataset:contains all images of noisy test dataset denoised using g. We will observe Accuracy2 $>$ Accuracy1.

IV. Implementation

Two separate sets of experiments were conducted using two different datasets, denoisers and classifiers. We describe the process followed below :

1) **Dog-cat dataset + Wavelet, nl-means denoisers[1] + 5-layer custom CNN**
   A binary dataset containing two classes of dog and cat images was chosen and the images were subjected to gaussian and salt and pepper noise to obtain the noisy dataset. This was then denoised, first using wavelet and nl-means filters and then using the j-invariant versions of the same denoisers. These were then classified using a 5-layer custom CNN. The accuracies obtained from normal denoised and j-invariant denoised images were compared to arrive at our conclusions.
2) **Rotated MNIST dataset + Wavelet, nl-means denoisers + RotEqNet[2]**
   The rotated MNIST dataset was corrupted with gaussian and salt and pepper noise to obtain the noisy dataset and was consecutively denoised using normal and j-invariant versions of wavelet and nl-means filters leaving us with 4 different datasets. All 4 datasets were then classified using the RotEqNet(Rotation Equivariant Vector Field Network) and obtained accuracies were compared and analysed to arrive at the conclusions.

We conducted two different experiments as briefed in section III. In both of them, we train the classifiers with clean images from the training dataset, which is then used to classify various denoised images. We used a learning rate of 0.001 and a weight decay of 0.01 for training in both experiments. The Adam optimiser was chosen to train the weights.

V. Experiments and Discussion

In our experiments, we use j-invariant functions to denoise our input images and then classify them using classifiers as described in Section III. For this purpose, we use two different datasets.

A. *Datasets*

- **Rotated MNIST Dataset**
  The MNIST dataset consists of 60,000 training images and 10,000 testing images of dimensions 28x28, which was downloaded from http://yann.lecun.com/exdb/mnist/. Rotated versions of the images comprising the Rotated MNIST dataset is obtained by random rotation of the images as mentioned in footnote 4.
- **Dog-cat binary Dataset**
  A binary dataset containing two classes namely dog and cat, with 12,500 training images and 6,250 testing images in each class was chosen for experimentation. This dataset with images of dimensions 224x224 was downloaded from https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification.

B. *Experimental results*

The results obtained from the 2 experiments described in Section III are summarised below:

1) Experiment-1: Our first experiment was denoising the noisy dog-cat dataset using normal and j-invariant denoisers corresponding to wavelet and nl-means filters, then classifying it using the 5-layer CNN shown in section IV.
The 5-layer CNN model was trained using the clean(no noise added) images of the dog-cat dataset.
Training accuracy = 67.76%
Validation accuracy = 69.92%
Testing accuracy(of clean images) = 60.75%

Gaussian noise					
		Wavelet filter		Nl-means filter	
Variance	Noisy	Normal	J-Invariant	Normal	J-Invariant
0.01	59	57.8	49.2	54	51
0.1	53.2	52	50.9	**56**	**58.5**
0.5	49.7	**49.85**	**51**	50.5	49
0.65	49.8	**50**	**50.1**	**47.5**	**48**
0.8	49.7	**50.1**	**51.17**	**47.5**	**49**
Salt and Pepper noise					
		Wavelet filter		Nl-means filter	
Amount	Noisy	Normal	J-Invariant	Normal	J-Invariant
0.01	56.54	55.3	49.5	53.5	52.5
0.1	55	**50.05**	**52.5**	**51.5**	**52.5**
0.5	48.9	50	50	48	47.5
0.65	49.48	**50**	**50.1**	**46**	**46.5**
0.8	50	**50.2**	**50.31**	**50**	**50.5**

TABLE I
Top half: Gives the accuracies of classification of various datasets obtained by the addition of Gaussian noise, then denoising using original and j-invariant denoisers. **Bottom half:** Same as top half except that s&p noise was added to obtain the noisy dataset.

2) Experiment-2: Our second experiment was denoising the rotated MNIST dataset using normal and j-invariant denoisers corresponding to wavelet and nl-means filters, then classifying it using the RotEqNet [2].
The RotEqNet was trained using the clean(no noise added) images of the rotated MNIST dataset.
No. of epochs = 10
Training accuracy = 83.58%
Validation accuracy = 88.22%
Testing accuracy(of clean images) = 91.21%

Gaussian noise					
		Wavelet filter		Nl-means filter	
Variance	Noisy	Normal	J-Invariant	Normal	J-Invariant
0.01	69.3	79.44	76.76	**69.91**	**73.09**
0.05	24.01	**30.42**	**36.88**	**23.88**	**31.99**
0.1	14.11	**15.46**	**23.19**	**13.89**	**18.91**
0.5	9.907	**10.15**	**11.88**	**10.11**	**11.02**
0.8	9.83	**10.07**	**10.85**	**10.07**	**10.44**
Salt and Pepper noise					
		Wavelet filter		Nl-means filter	
Amount	Noisy	Normal	J-Invariant	Normal	J-Invariant
0.01	76.84	86.71	85.23	76.6	72.57
0.05	58.4	**72.9**	**76.07**	**59**	**59.33**
0.1	40.72	**53.12**	**54.44**	**38.07**	**40.95**
0.3	10.45	**10.97**	**12.13**	**10.56**	**11.3**
0.6	9.823	**10.07**	**10.1**	**10.07**	**10.08**

TABLE II
Gives accuracies of classification when various datasets obtained by the addition of Gaussian and s&p noise, then denoising using original and j-invariant denoisers.

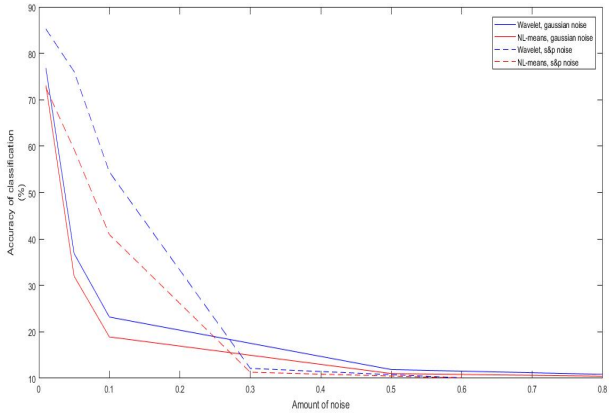

Fig. 3. Gives us the comparison of accuracies obtained by classification of noisy images denoised by 2 different(nl-means,wavelet) j-invariant denoisers for gaussian and s&p noise.

VI. Results and analysis

A. *Variation of accuracy with type of denoier used*

For experiment-1, table I shows that for both gaussian and salt and pepper noise, the accuracy obtained from denoising the noisy dataset using j-invariant denoiser is higher than that obtained from corresponding normal denoiser for noise levels above 40%. We also observe the poor performance of denoisers when the noise levels are low. This is because of the fact that all denoisers have a smoothening effect when they denoise the images. This sometimes deteriorates the information that NNs use for classification. ∴ While low noise doesn't change the value of pixels greatly, denoising worsens the pixels by changing them too much. Hence the NN's performance is better when denoised using j-invariant denoiser than normal denoiser for noise levels above 40%.

For experiment-2, we observe from results in table II that, for most cases, the classifier classifies the images denoised by j-invariant denoisers with higher accuracy than those denoised by the original denoising filters. This is true for most of the noise range (above 1%). An increase in accuracy of 1-7% was observed in case of both wavelet and nl-means denoisers. This also indicates the considerable improvement in the NN's performance in presence of j-invariant denoisers.

Figure 3, obtained from experiment-2, shows us that, in the presence of gaussian as well as salt and pepper noise, the j-invariant wavelet denoiser performs better than j-invariant nl-means denoiser.

B. *Variation of accuracy with hyper-parameter of the denoiser*

Every denoiser has variable hyper-parameters which can be varied to obtain the optimal denoiser as discussed in section II.

Bilateral filter		Nl-means filter		Median filter	
Hyper-parameter	Accuracy	Hyper-parameter	Accuracy	Hyper-parameter	Accuracy
0.01	23.81	1	49.62	1	60.69
0.03	23.81	2	49.621	2	61.05
0.05	23.8102	3	49.621	3	51.6
0.1	23.8102	4	49.62	4	35.16
0.5	23.81	5	49.63	5	14.09

TABLE III
Gives the variation of accuracy of classification with variation in hyper-parameters of 3 different denoisers. This table is the observation for noisy images containing gaussian noise of variance 0.05.

From table III we observe that for bilateral and nl-means filter, the accuracy obtained after denoising does not vary with variation in hyper-parameters. We have observed similar behavior in wavelet denoisers also. But for the median filter, we can observe a significant variation. Therefore an optimal median denoiser will have to be chosen by minimising the self-supervision loss, as described in section II. When we use NNs as denoisers we can include the self-supervision loss into the loss function which is minimised in every epoch as discussed in section II-C.

VII. Conclusion

Neural networks(NNs) are one of the most important and useful tools in the current era of science. They have proven to achieve outstanding performance in various contemporary applications. Albeit their efficiency in tasks like image classification, their performance has been observed to degrade in the presence of noise. To lessen the effect of noise on the classifiers, we first denoise the images using j-invariant functions and then use the denoised images for classification. From the experiments done to confirm our hypothesis, we infer that the classifier gives better accuracy when denoised using the j-invariant version of a denoiser for higher noise levels. In some cases, we obtain this result over the entire noise range. We also observed how different filters vary with hyper-parameters. For all the denoisers we have used in the experiments, the accuracy value remains independent of the hyper-parameter and hence the optimisation part was not required. Hence any traditionally used denoiser can be converted into a j-invariant denoiser and can be used to preprocess noisy input images for obtaining better classification results.

References

[1] Joshua Batson and Loıc Royer. "Noise2Self: Blind Denoising by Self-Supervision". In: *CoRR* abs/1901.11365 (2019).

[2] Diego Marcos et al. "Rotation Equivariant Vector Field Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017).

[3] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[4] Tiago Nazaré et al. "Deep Convolutional Neural Networks and Noisy Images". In: Jan. 2018, pp. 416–424. ISBN: 978-3-319-75192-4.

[5] Sudipta Singha Roy, Mahtab Ahmed, and MAH Akhand. "Classification of massive noisy image using auto-encoders and convolutional neural network". In: *2017 8th International Conference on Information Technology (ICIT)*. IEEE. 2017, pp. 971–979.

[6] Sudipta Singha Roy, Mahtab Ahmed, and Muhammad Aminul Haque Akhand. "Noisy image classification using hybrid deep learning methods". In: *Journal of ICT* 17.2 (2018), pp. 233–269.

[7] Sudipta Singha Roy et al. "A robust system for noisy image classification combining denoising autoencoder and convolutional neural network". In: *International Journal of Advanced Computer Science and Applications* 9.1 (2018), pp. 224–235.