Fusing Convolutional Neural Networks with a Restoration Network for Increasing Accuracy and Stability

Hamed H. Aghdam(✉), Elnaz J. Heravi, and Domenec Puig

Department of Computer Engineering and Mathematics,
University Rovira i Virgili, Tarragona, Spain
{hamed.habibi,elnaz.jahani,domenec.puig}@urv.cat

Abstract. In this paper, we propose a ConvNet for restoring images. Our ConvNet is different from state-of-art denoising networks in the sense that it is deeper and instead of restoring the image directly, it generates a pattern which is added with the noisy image for restoring the clean image. Our experiments shows that the Lipschitz constant of the proposed network is less than 1 and it is able to remove very strong as well as very slight noises. This ability is mainly because of the shortcut connection in our network. We compare the proposed network with another denoisnig ConvNet and illustrate that the network without a shortcut connection acts poorly on low magnitude noises. Moreover, we show that attaching the restoration ConvNet to a classification network increases the classification accuracy. Finally, our empirical analysis reveals that attaching a classification ConvNet with a restoration network can significantly increase its stability against noise.

Keywords: Image restoration · Image denoising · Traffic sign classification · Deep learning

1 Introduction

Convolutional Neural Networks (ConvNets) have considerably advanced compared with AlextNet [1] which won the ImageNet competition in 2012. In particular, depth of ConvNets have greatly increased last years. Szegedy *et al.* [2] created a network consisting of multiple Inception modules. Besides, Simonyan and Zisserman [3] proposed a network with 19 layers. The idea behind this ConvNet is to increase the depth rather than its width. Srivastava *et al.* [4] showed how to train very deep networks by directly flowing information from previous layers to next layers through a gate function. Recently, He *et al.* [5] trained a 152-layer ConvNet and won the ImageNet competition. Their ConvNet is similar to [4] in the sense that information flows directly to the following layers. However, the gate function in [4] has been replaced with an identity mapping function.

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part I, LNCS 9913, pp. 178–191, 2016.
DOI: 10.1007/978-3-319-46604-0_13

Despite the impressive results obtained by ConvNets, Szegedy *et al.* [6] showed that small perturbation of input images can alter their classification result. The perturbed samples are called *adversarial examples*. The difference between the original image and its adversarial sample is not sometimes even recognizable to human eye. They study the reason by computing the *upper bound* of the Lipschitz constant for each layer. The results suggest that instability of ConvNets might be due to the fact that they are highly non-linear functions. As the result, a small change in the input may considerably change the output. We also carried out an empirical study on various ConvNet architectures trained on different datasets [7]. In this work, we generated 1200 noisy images for each sample in the test sets using a Gaussian noise with $\sigma \in [1 \dots 40]$. The results showed that all the ConvNets in our experiments were unstable to image degradation even when the samples were degraded using the Gaussian noise with $\sigma = 1$. Moreover, the ConvNets were unstable regardless of the class of the object.

Contribution: In this paper, we deal with adversarial examples from another perspective. Denoting the *softmax* layer of a ConvNet (*i.e.* the last layer in a classification ConvNet) by $\mathcal{L}_\theta(x)$, the general idea is to find a parameter vector θ such that:

$$\forall_{\|\nu\| \leq \epsilon} \; \mathcal{L}_\theta(x + \nu) = \mathcal{L}_\theta(x) \tag{1}$$

where ν is an additive noise vector whose magnitude is less than ϵ. As we mentioned earlier, solving instability problem using the above formulation may require to add new terms to the loss function or devise new sets of layers to our ConvNet. Instead, we utilize currently available layers, optimization technique and loss function to increase the stability of the ConvNets. Mathematically, we are looking for two sets of parameters θ_1 and θ_2 such that:

$$\forall_{\|\nu\| \leq \epsilon} \; \mathcal{L}_{\theta_1}(\mathcal{F}_{\theta_2}(x + \nu)) = \mathcal{L}_{\theta_1}(\mathcal{F}_{\theta_2}(x)). \tag{2}$$

In other words, our aim is to find a function $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ that is able to map all points around $x \in \mathbb{R}^n$ to the same point. If we can find such a function, the sample x and all its adversarial examples will be mapped to the same point. Then, the classification ConvNet will be able to produce the same output for all adversarial examples. In Sect. 3, we explain how to model the function $\mathcal{F}$ using a ConvNet and train a classification ConvNet using $\mathcal{F}$. Our experiments in Sect. 4 show that the proposed formulation is able to increase the stability of ConvNets against additive noise. Also, it increases the classification accuracy. Last but not the least, we show that our formulation is able to deal with high magnitude as well as low magnitude noises.

2 Related Work

Szegedy *et al.* [6] discovered that ConvNets are sensitive to small variations of the input. They found the additive noise ν which was able to reduce the score of the true class close to zero. They also studied the non-linearity of ConvNets using the Lipschitz theorem. Similarly, Papernot *et al.* [8] produced adversarial

samples which were incorrectly classified by the ConvNet. They produced these samples by modifying 4.02 % of the input features. We also proposed an objective function to find the minimal additive noise ν in the closest distance to the decision boundary in which $x+\nu$ falls into the wrong class [7]. Goodfellow *et al.* [9] argued that the instability of ConvNet to adversarial examples is due to linear nature of ConvNets. Based on this idea, they proposed a method for quickly generating adversarial examples. They used these examples to reduce the test error.

Gu and Rigazio [10] stacked a denoising autoencoder (DAE) to their ConvNet and preprocessed the adversarial examples using the DAE before feeding them to the ConvNet. They mentioned that the resulting network can be still attacked by new adversarial examples. Inspired by contractive autoencoders, they added a smoothness penalty to the objective function and trained a more stable network.

Instead of trying to minimize the classification score, Sabour *et al.* [11] tried to find a degraded image closest to the original image that its representation mimics those produced by natural images. Fawzi *et al.* [12] provided a theoretical framework for explaining the adversarial examples. Their framework suggests that the instability to noise is due to low flexibility of classifiers.

3 Proposed Method

As we mentioned in (2), our aim is to learn the function $\mathcal{F}(.)$ that is able to map the clean sample x and adversarial sample $x+\nu$ to vector y. In this paper, we set $y = x$ for simplicity. In other words, $\mathcal{F}(x+\nu) = \mathcal{F}(x) = x$. With this formulation, $\mathcal{F}$ can be seen as a function for denoising the additive noise from x. In contrast to [13], we do not restrict $\mathcal{F}$ to Gaussian noise. Furthermore, contrary to [14] that models $\mathcal{F}$ using 16 M parameters, we are interested in models with much fewer parameters and higher capacity.

Our model, $\mathcal{F}$, must be able to model wide range of random noises. From one perspective, $\mathcal{F}$ can be seen as an associative memory that is able to memorize all patterns $X = \{x_1 \dots x_i \dots_M\}, x_i \in \mathbb{R}^N$ in our dataset and map every sample $\{x_i + \nu | \|\nu\| \leq \epsilon\}$ to x_i. Here, x_i is an image patch and X is the set of all possible image patches collected from all classes of objects in our dataset.

The required memorizing capacity of $\mathcal{F}$ directly depends on the diversity and complexity of the patterns in X. Neural networks are non-linear functions with high capacity to memorize complex patterns. On the one hand, conventional neural networks (consisting only of fully connected layers) cannot be deep if we want to increase the width of the network since it can dramatically increase the number of the parameters. Another disadvantage of fully connected networks for modeling $\mathcal{F}$ is their computational complexity. On the other hand, complex patterns can be modeled and memorized better using deep models. Convolutional neural networks address both these problems using the weight sharing strategy. This makes it possible to increase both width and depth of the network. For this reason, we model $\mathcal{F}$ using a deep ConvNet. Compared with [14] and [15], our ConvNet is deeper and has more representation power. Also, in contrast to [17]

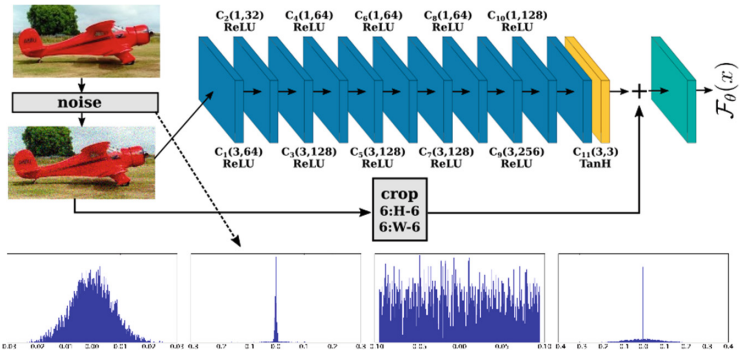

Fig. 1. The proposed ConvNet for modeling $\mathcal{F}$ in (2). Different noises including the Gaussian noise are generated as ν in (2). The histograms indicate a fews examples of probability density functions utilized for generating noise.

that uses a network with 4.5 M parameters, our ConvNet requires only 0.4 M (400 K) parameters. Finally, our network is different from all these networks in the sense that it facilitates learning $\mathcal{F}(x) = x$ using a shortcut connection. Figure 1 illustrates the architecture of our ConvNet.

The network consists of 11 convolution layers in which it only utilizes 3×3 or 1×1 kernels. Except the last convolution layer, there is a 1×1 convolution layer after every 3×3 convolution layer. The 1×1 kernels have been mainly used for two purposes. First, they apply a non-linear transformation on the input similar to the idea in [16]. Second, they reduce the number of the parameters in the next layer by decreasing the number of the feature maps. In fact, the number of the feature maps in each 1×1 layer is half of the number of the feature maps in the previous 3×3 layer.

Since the depth of the network is high, it is not practical (*e.g.* vanishing gradient problem) to use saturating non-linearities such as the hyperbolic or sigmoid functions. For this reason, we use *leaky rectified liner units* [17] with $\alpha = 0.01$ as the activation function in our network. However, the activation function of the last convolution layer is the *hyperbolic* function. Using this activation function, the last convolution layer is able to generate negative values which is crucial to remove noise from the image. In addition, the noise level must be in $[-1, 1]$.

In contrast to the previous methods that we mentioned in Sect. 2, the last convolution layer does not produce the restored image. Instead, it predicts a pattern that in which the clean image is restored by adding this pattern with the degraded image. In the case of additive noise, the output of the last convolution layer is $-\nu$ (see (2)). Adding $-\nu$ with the noisy input $x + \nu$ produces the clean input x. It should be noted that our network is able to denoise any kind of noise and it is not restricted to additive noise. In the case of non-additive noise, the last convolution layer generates a pattern that restores the clean image when it is added to the degraded image. The shortcut connection in our network plays

a similar role as in [4,5] and it facilitates the training of the deep network. However, its major advantage is that it makes it easier for the network to learn the identity mapping function (*e.g.* when the image is not degraded).

It should be noted that the noise generation module in Fig. 1 is only used during the training phase. In the test phase, the noise generation module is omitted. In this paper, we have only concentrated on additive noise. The noise generation module creates noisy patterns with various probability density functions. Five examples of the probability density functions have been shown in the figure. Besides the Gaussian and uniform distributions, there are also other density functions that generate sparse noise patterns.

The proposed network accepts inputs bigger than 12×12 pixels. Given a $H \times W$ image, the output of the last convolution layer is a $H - 12 \times W - 12$ image. Hence, the input image cannot be directly added with the output of the last convolution layer. To this end, the image is cropped (central cropping) in the shortcut connection before adding with the last convolution layer. Although the need for cropping could be solved by padding the input image in the first convolution layer, notwithstanding, our experiments showed that 0-padding the image in the first layer decreases the training performance due to the border effect. However, after the network has been trained, we zero-pad the image in the first convolution layer and omit the cropping module in the test phase.

Finally, although the receptive field of each convolution kernel is small, the whole network works with 13×13 receptive fields. In other words, any element in the last convolution layer is computed using a 13×13 patch in the input image. In terms of associative memory network, our ConvNet tries to associate 13×13 image patches to a single number. It is possible to increase the receptive field by adding more 3×3 layers to the network and keep the number of the parameters low by adding 1×1 layers. However, we did not find a significant improvement by adding more layers to the network and the performances were comparable to the 11-layer network when they are trained on the datasets that we have used in this paper.

As we show in the next section, the trained network can be stacked to any classification network to restore the input images. The classification network (all layers) must be fine-tuned after freezing the layers of the restoration network.

4 Experiments

The restoration network is trained by minimizing the L_2 regularized *mean square error*:

$$E = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \|x_i - \mathcal{F}_\theta(x_i + \nu_j)\|^2 + \lambda \|\theta\|^2 \tag{3}$$

where N is the total number of the images and M is the number of times that a noise pattern is generated for each sample. Also, θ is the set of network weights and biases and λ is the regularization coefficient. It is worth mentioning that the noisy patterns are generated on the fly. That said, we have implemented

a degradation module which accepts a mini-batch of clean images and outputs their degraded version. Each sample in the mini-batch is degraded M times.

4.1 Implementation Details

We implemented the proposed network using Caffe framework [18]. However, the original Caffe framework is only able to generate Gaussian and uniform noises with known standard deviation and ranges. For this reason, we modified the noise generation class in the library[1] in order to generate Gaussian noises with random standard deviations and uniform noises with random ranges. Also, the generated noises are probabilistically processed by sparsifying the results or manipulating noise intensities heuristically.

All the weights in the network are initialized using the method in [19]. Although the input of the network could be images bigger than 12×12 pixels, we use 48×48 image patches as the input of the network. It is also possible to use bigger image patches, but, it will reduce the mini-batch size because of limitations of the memory on the GPU device. The network is trained using the momentum stochastic gradient descend, mini-batch size 50 and exponential learning rate annealing. We set momentum to 0.9, learning rate to 0.001, learning rate annealing to 0.99996, λ to $1e-6$ and maximum number of iterations to 10^5. Also, following the idea in [15] we set the learning rate of the last convolution layer to 0.0001. Note that we *do not* apply zero-padding in the training phase. Besides, the network input is normalized to $[0, 1]$. This is important since if we do not normalize the input of the network to this range, the output of the last convolution layer must be scaled accordingly to be in the same range as the input image.

After training the network, zero-padding is applied only in the first convolution layer (the size of padding is 6 for each side). Also, it is possible to feed images with arbitrary size to the network. We only need to always crop the network input 6 pixels from each side before adding with the output of the last convolution layer. Last but not the least, it is possible to use other color spaces instead of RGB space. The only modification that the network needs is to set the number of the feature map in the last convolution layer equal to the number of the channels in the input layer.

4.2 Dataset

In this section, we first assess the restoration capability of our ConvNet. Then, we evaluate how preprocessing of the inputs of a classification network using our ConvNet can affect the accuracy. To this end, we carry out our experiments on the German Traffic Sign Recognition Benchmark (GTSRB) dataset [20] which contains 43 classes of vertical traffic signs. Each color image in this dataset contains one traffic sign and they vary from 15×15 to 250×250 pixels. The training set consists of $39,209$ images and the test set contains $12,630$ images.

[1] We added a new function to filler.hpp file.

We use 10 % of the training set for validation. We first crop the image within the annotated bounding box and resize all the images to 48×48 pixels.

The GTSRB dataset has some important characteristics. First, it has been collected considering real scenarios and it contains many degraded images. Second, the imaging device is noisy and it has degraded edges of the objects in some cases. Third, the resolution of images are low. Therefore, a slight change in the image may affect the classification score.

Qualitative analysis: We randomly degraded some of the samples in the test set and restored them using our ConvNet. Figure 2 shows the results. Four different images have been shown for each sample starting from the noisy image in the left. Next, the output of the last convolution layer (*i.e.* ν) and the restored image are illustrated in the second column and third columns. Finally, the fourth column shows ν after normalizing the values between $[0, 1]$.

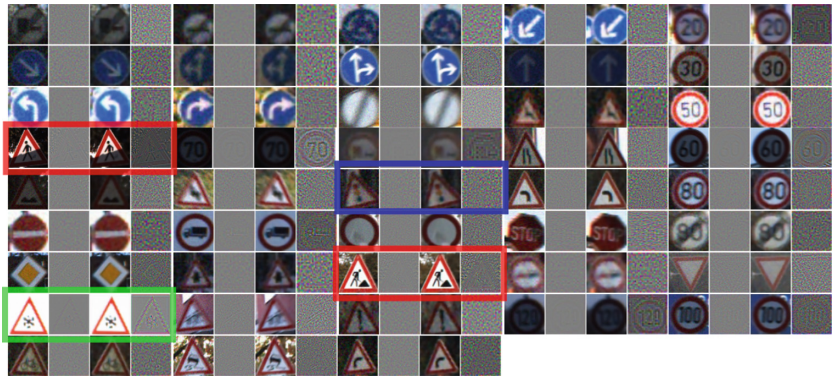

Fig. 2. Samples of restored images for each classes of traffic signs (best viewed in color and electronically). (Color figure online)

First we observe that samples indicated by the red rectangles are not strongly degraded. In other words, the restored image must be very close to the degraded image. We see that the ConvNet is able to accurately restore the image without manipulating the edges of the traffic sign. However, it slightly changes the background of the object. In addition, the background of the sample inside the green rectangle is completely bright. Also, the white color inside the traffic sign is not noisy. However, due to excessive amount of illumination, the edges of the traffic sign are slightly degraded. As it is clear from the normalized ν inside this rectangle, the ConvNet only manipulates the edges and ignores the clean pixels of the image. Finally, the strongly degraded image in the blue rectangle has been properly recovered using our ConvNet. In particular, the edges of the traffic sign are more clean and its background it more smoother. As we will discuss shortly, these three properties improves the classification accuracy of the classification ConvNet trained on this dataset.

Exploratory analysis: In order to analyze the performance of the proposed ConvNet, we created several versions of our ConvNets with various depth. In addition, we also implemented the ConvNet in [15] for the task of restoration. The architecture of the ConvNets are illustrated in Table 1. In the case of our ConvNets, the activation function of the last layer is the hyperbolic function. Also, the architecture and training procedure of last row in the table is exactly the same as in [15] except it is adapted on RGB images. It is worth mentioning that there is no shortcut connection in [15].

Table 1. Different ConvNet architectures trained for image restoration. The activation function of the last layer in all the ConvNets is hyperbolic function. Refer to the text for explanation.

ConvNet	#3x3	#1x1	#3x3	#1x1	#3x3	#1x1	#3x3	#1x1	#3x3	#1x1	#3x3
L11 (our)	64	32	128	64	128	64	128	64	256	128	3
L9	64	32	128	64	128	64	128	64	3	-	-
L7	64	32	128	64	128	64	3	-	-	-	-
L5	64	32	128	64	3	-	-	-	-	-	-
L3	64	32	3	-	-	-	-	-	-	-	-
	#9x9	#1x1	#5x5								
Dong [15]	64	32	3								

We analyzed the restoration capability of these ConvNets as follows: We picked 1263 samples from the test set and for each sample, 150 noisy samples were generated using the same generation module as in Fig. 1. Then, the noisy samples were fed to every ConvNet and the *peak signal to noise ratio* (PSNR) of the restored images were computed using the following equation:

$$psnr = 10 \log_{10} \left(\frac{255^2}{\frac{1}{HW} \sum_{m=i}^{H} \sum_{n=j}^{W} (x - x')^2} \right). \tag{4}$$

In the above equation, x is the clean image and x' is the noisy or restored image. Figure 3 shows different properties of the above ConvNets. The top plots show the PSNR of the noisy images versus the PSNR of the restored images along with the best polynomial line fitted on the result of each ConvNet, separately. We have fitted a line on the top-left plot and a second order polynomial on the top-right plot. In terms of restoration, the model in [15] has the lowest PSNR if the PSNR of the noisy image is more than 20 dB. In addition, L7, L9 and L11 ConvNets show similar performances. However, L5 ConvNet starts to act poorly compared with these three network when the PSNR of the input is more than 37 dB. Finally, L3 shows a more accurate behavior in higher PSNRs but its performance decreases quickly when the PSNR of the input is less than 35 dB.

Note that we are not only looking for a network that is able to accurately restore images. In (2) we mentioned that our aim is to increase the performance

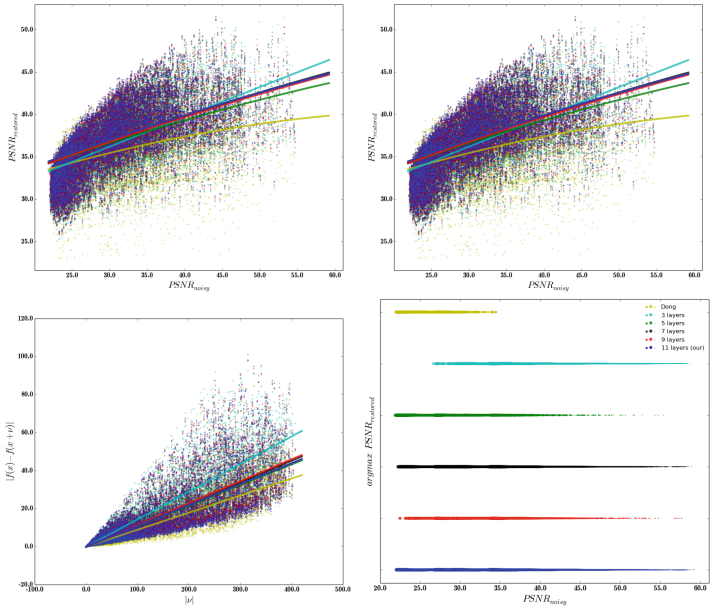

Fig. 3. PSNR of the ConvNet in Table 1 computed on the test samples. Refer to text for explanation (best viewed in color and electronically). (Color figure online)

and the stability a ConvNet by mapping all nearby samples around x to the same point. In other words, the output of the ConvNet must change slightly when the input changes dramatically. We have investigated this in the bottom-left plot in Fig. 3. The horizontal axis shows the magnitude of noise and the vertical axis shows the difference between the output of the networks supplied with the clean and noisy images. First, we observe that all the network are contraction. This means that the Lipschitz constant is in range $[0, 1)$ which is a desirable property for our task. Second, the network in [15] has the smallest value of K since for any magnitude of noise, the difference between the output of the clean image and the output of the noisy image is smallest compared with the other networks. Third, L5, L7, L9 and L11 act similarly but the level of contraction in L11 is more than L7 and L9. Also, L3 acts poorly compared with the other networks.

Quantitative results: We have tested the above networks by generating $1263 \times 150 = 189,450$ images. For each network in our experiment, we counted the number of the samples that each network has produced the highest PSNR. Table 2 shows the results. We observe that L11 has the highest number of hits by producing the best result in 52 % of the noisy samples. In addition, its output is within the top-2 PSNRs in the case of 76 % of the samples. The bottom-right plot in Fig. 3 shows the distribution of hits of each network, separately. We observe that L11 is able to produce the highest PSNR in noisy images with any magnitude of noise. This is consistent with the results shown in Fig. 2 in which

we illustrated that the network is able to produce accurate results for slightly degraded (the red and green rectangles) as well as the strongly degraded (blue rectangle) images. Moreover, we observe that the network proposed in [15] is only able to produce accurate results for strongly degraded images. In other words, if the image is not noisy, this network might discard some of the important details of the object in the restored image.

Table 2. Number of times that each network has produced the output with highest PSNR.

Network	Top-1	Top-2
L11	95739	146854
L9	5184	32874
L7	36395	117268
L5	6759	24903
L3	39548	47250
Dong [15]	5825	9751

Based on these results, L11 is the better choice for restoring images compared with L3, L5, L7 and L9 since it produces images with higher PSNR and also its value of K is small. Although [15] posses smallest K among other network, it has also the worst restoration power compared with other networks. Consequently, we pick L11 as the restoration model for the task of classification in following experiments.

Classification of traffic signs: We have previously proposed a ConvNet for classifying traffic signs which is able to classify 99.51 % of the test samples in the GTSRB dataset correctly using an ensemble of 5 ConvNets [21]. Also the best accuracy of a single ConvNet is 99.05 % in this work. After training the restoration network, we attach the input of the classification ConvNet to the output of the restoration ConvNet. Then, the layers of the restoration ConvNet are frozen and the classification network is fine-tuned using noisy samples generated by the same module as in Fig. 2. Note that, we do not train the classification ConvNet from scratch. Instead, we use the pre-trained model from our previous work to initialize the weights of the classification network.

We finetuned the classification ConvNet using the mini-batch size 50 and 15000 iterations. After fusing the classification ConvNet with the restoration network, classification accuracy of the single ConvNet was increased to 99.37 % after 6500 iterations. We also noticed that the accuracy was converged after 15000 iterations. Table 3 shows the results.

We observe that the classification accuracy increases 0.47 % which is equal to 59 samples in the test set. Note that the winner of the GTSRB competition [22] classified 99.46 % of the test samples using an ensemble of 25 ConvNets where a

Table 3. Classification of the traffic signs before and after attaching the restoration network.

Network	Top-1(%)	Top-2(%)	Top-5(%)
Single network	99.05	99.69	99.81
Restoration+single network	99.52	99.80	99.85
Single network in [22]	98.80	NA	NA

single network consisted of 1.5 M parameters. However, harnessing our classification ConvNet with the proposed restoration network, we could correctly classify 99.52 % of the test samples. Our fused network consists of 1.5 M parameters. Taking into account the fact that [22] have achieved a comparable result to our network using an ensemble of 25 network, our fused ConvNet is approximately 25 faster than the ensemble in [22]. Yet, it is also more accurate. We analyzed the samples that are incorrectly classified using our single network and they are correctly classified after fusing the classification ConvNet with the restoration network. Figure 4 shows some of these samples.

Fig. 4. Some of the samples that are incorrectly classified using the single network but they are correctly classified in the fused network (best viewed in color and electronically). (Color figure online)

Looking at the noisy sample inside the red and green rectangles, we observe that the restoration network has removed some artifacts from the pictograph area while it has improved the quality of the pictographs. Similarly, the edges of the sample inside the yellow rectangle is noisy due to excessive background illumination. The restoration network reduces this effect and produces an image that is correctly classified by the network. Last but not the least, the background noise has been reduced in the sample indicated by the blue rectangle.

Analyzing Stability: In order to analyze the stability of the fused network, we considered four different ConvNets including the original network before and after attaching the restoration network, the original network finetuned using a training set that is augmented with many noisy samples and the original network that applies a Gaussian smoothing in the first layer.

Next, to empirically study the stability of the ConvNets against noise, the following procedure is conducted. First, we pick the test images from the original

Table 4. Accuracy of the ConvNets obtained by degrading the *correctly classified test images* in the original datasets using a Gaussian noise with various values of σ.

Network	Accuracy (%) for different values of σ											
	1	2	4	8	10	15	20	25	30	35	40	Overall
Original	99.0	98.9	98.9	98.3	97.8	96.5	93.4	90.3	87.0	83.7	80.6	93.2
Noisy augmented	100.0	99.9	99.8	99.3	99.0	98.3	96.8	95.0	92.9	90.7	88.4	96.4
Smoothing	99.9	99.9	99.8	99.3	98.9	97.7	95.2	92.6	89.6	86.5	83.6	94.9
Restore+original	100.0	100.0	99.9	99.8	99.7	99.4	98.8	98.1	97.1	95.9	94.4	98.5

datasets which are *correctly classified* by each ConvNets. Then, 150 noisy images are generated for each $\sigma \in \{1, 2, 4, 8, 10, 15, 20, 25, 30, 35, 40\}$. In other words, 1650 noisy images are generated for each of correctly classified test images from the original datasets. We only consider the correctly classified samples since we are interested in finding how noise can change the class of the correctly classified samples. In other words, it might not be logical to add noise to a misclassified sample to see whether or not it is misclassified by the ConvNet. Table 4 shows the accuracy of the ConvNets per each value of σ.

It is clear that the network equipped with the restoration network produces better results with a significant margin to other methods. This is mainly due to the same reasons that we mentioned on Fig. 4. The restoration network is able to recover important parts of the degraded image but simple noise removal techniques such as smoothing is not able to do a complex restoration. In addition, we observe that augmenting the dataset with noisy images is advantageous. But, it is still prone to samples that are strongly degraded.

5 Conclusion

In this paper, we proposed a ConvNet to accurately restore degraded images. Our ConvNet is different from rest of the networks proposed for image denoising in the sense that it is deeper and it utilizes a shortcut connection to find a pattern which is added with the noisy image in order to restore the clean image. This is different from other networks in the sense that they try to directly restore the clean image. Our experiments showed that the proposed ConvNet is able to accurately restore strong and slightly degraded images. Restoring slightly degraded images is not trivial since ConvNets are highly non-linear functions and a slight change in the input can cause a significant change in the output. We showed this problem using a previously proposed ConvNet. However, because of the shortcut connection, our ConvNet is able to deal with small magnitude noises. Furthermore, we attached our network to the classification ConvNet proposed for the GTSRB dataset and showed that the accuracy of the same ConvNet increase from 99.05 % to 99.52 %. Finally, we empirically analyzed stability of the fused classification ConvNet and showed that it produces more accurate results compared with other methods.

Acknowledgements. Hamed H. Aghdam and Elnaz J. Heravi are grateful for the supports granted by Generalitat de Catalunya's Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) and University Rovira i Virgili through the FI-DGR 2015 fellowship and the Marti Franques fellowship, respectively.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105. Curran Associates, Inc. (2012)
2. Szegedy, C., Reed, S., Sermanet, P., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: arXiv preprint, pp. 1–12 (2014). arXiv:1409.4842
3. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representation (ICLR), pp. 1–13 (2015)
4. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks (2015). arXiv:1505.00387
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: arXiv prepring (2015). arXiv:1506.01497
6. Szegedy, C., Zaremba, W., Sutskever, I.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR), pp. 1–10 (2014)
7. Aghdam, H.H., Heravi, E.J., Puig, D.: Analyzing the stability of convolutional neural networks against image degradation. In: Proceedings of the 11th International Conference on Computer Vision Theory and Applications (2016)
8. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 1st IEEE European Symposium on Security and Privacy, Saarbrucken. IEEE (2016)
9. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representation (ICLR), pp. 1–11 (2015)
10. Gu, S., Rigazio, L.: Towards deep neural network architectures robust to adversarial examples (2013), pp. 1–9 (2014). arXiv:1412.5068
11. Sabour, S., Cao, Y., Faghri, F., Fleet, D.J.: Adversarial manipulation of deep representations. arXiv preprint, pp. 1–10 (2015). arXiv:1511.05122
12. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. In: Number 2014, pp. 1–14 (2015). arXiv:1502.02590
13. Jain, V., Seung, S.: Natural image denoising with convolutional networks. In: NIPS 21, pp. 769–776. Curran Associates, Inc. (2009)
14. Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 633–640 (2013)
15. Dong, C., Loy, C.C., He, K.: Image super-resolution using deep convolutional networks. arXiv preprint 8828(c), pp. 1–14 (2014)
16. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint, p. 10 (2013)
17. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning, vol. 30 (2013)
18. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., Eecs, U.C.B.: Caffe: convolutional architecture for fast feature embedding. In: ACM Conference on Multimedia (2014)

19. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), vol. 9, pp. 249–256 (2010)
20. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. **32**, 323–332 (2012)
21. Aghdam, H.H., Heravi, E.J., Puig, D.: Recognizing traffic signs using a practical deep neural network. In: Reis, L.P., Moreira, A.P., Lima, P.U., Montano, L., Muñoz-Martinez, V. (eds.) AISC 2006. AISC, vol. 417, pp. 399–410. Springer, Heidelberg (2016). doi:10.1007/978-3-319-27146-0_31
22. Cirean, D., Meier, U., Masci, J., Schmidhuber, J.: Multi-column deep neural network for traffic sign classification. Neural Netw. **32**, 333–338 (2012)