Report 05

Yiming Bian

Jun 17th 2022

Major Professor

Arun Somani(arun@iastate.edu)

Committee Members

Henry Duwe (duwe@iastate.edu)
Aditya Ramamoorthy (adityar@iastate.edu)
Alexander Stoytchev (alexs@iastate.edu)
Cindy Yu (cindyyu@iastate.edu)

1 Introduction

In the past two weeks, I was working with Dr. Somani on a paper called "Effective Management of Intense Salt and Pepper Noise Image Classification". This paper concludes the experiments we have done so far on salt and pepper noise image classification. Several insights are

- When the noise is sparse, median filter is a more reliable and preferred choice.
- When the noise is intense, noise-trained models bring surprises. For example, tested on images with salt and pepper noise of intensity 0.5, the best noise-trained model can achieve an accuracy of 85.39% and 78.15% with ResNet-18 and VGG16 architectures, respectively.
- Different CNN models vary in the behavior when processing sparse noise images.

2 Plan for next two weeks

Currently, we are looking for a conference to submit this paper and will do some minor revisions according to the requirements of the conference. In

the mean time, as suggested by Dr. Somani in our meeting yesterday, an experiment to explore the potential benefit of mix training will be done. In this experiment, we mix the noise images to construct the training data. We plan to mix images with noise intensity of 0.1, 0.2, 0.3 in a ratio of $(25\%, 50\%, 25\%)$, $(33\%, 33\%, 33\%)$, $(20\%, 40\%, 40\%)$ and $(40\%, 40\%, 20\%)$. If the results show clear improvement or advantages compared to the current best model, we update the paper with the latest results. The detailed setup, results and analysis will be covered in the next report on July 1st.

After this paper, I will start working on a paper focusing on speckle noise. Moreover, I plan to investigate what causes the different behaviors after the noise training among CNNs. Some architecture designs must contribute to or degrade the training process.

Below, we present the paper "Effective Management of Intense Salt and Pepper Noise Image Classification". If you have any suggestion or question, you are very welcome to send me an email! Thanks.

Effective Management of Intense Salt and Pepper Noise Image Classification

Yiming Bian and Arun K. Somani
Department of Electrical and Computer Engineering
Iowa State University, Ames, Iowa 50010
{ybian, arun}@iastate.edu

***Abstract*—In the past decade, machine learning based classification of high-quality images have made remarkable progress. However, the classification of low-quality images poses new challenges. This work focuses on the improvement of classification accuracy on images with salt and pepper noise. Compared to widely adopted pre-processing techniques such as median filter, our method retrains the model on noise data directly and the noise-trained models show very high resilience. When salt and pepper noise is sparse in an image, classification accuracy of nosie-trained models vary on convolutional neural networks (CNN). VGG16 noise-trained models show comparable accuracy to median filter when the noise intensity is low. On the other hand, ResNet-18 models have a dissimilar performance. But median filter remains reliable and effective. When salt and pepper noise is intense in an image, median filter is outperformed by the proposed noise-trained models with the largest accuracy gap of 11.23% and an average of 6.97%.**

***Index Terms*—salt and pepper noise, noise image classification, noise training, noise-trained CNN models**

I. INTRODUCTION

Image classification is one of the most substantial and comprehensive problems in computer vision. In the past decade, numerous machine learning based techniques were introduced and remarkable progress and achievements have been made.

In 2009, a large database, which contains over 14 million hand-annotated images, designed for visual object recognition research called ImageNet was presented by Li Fei-Fei et al. [1]. The next year, ImageNet project began an annual contest called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [2]. This challenge uses a subset of ImageNet that has $1,000$ non-overlapping classes of objects. As it is commonly acknowledged that the average recognition ability of human has a top-5 error rate of 5%, the winner in 2015, *ResNet* [3], beat human level for the first time with an error rate of 3.6%. The state-of-art model, *Florence* [4], achieved an extraordinary error rate of 0.98%. Behind all these intriguing figures, it is reasonable to conclude that the improvement of classification accuracy on high-quality images is merely a challenge anymore.

Expensive lens and camera systems are manufactured to capture top-quality photos. They incur high cost to store, process and transfer. On the other hand, low-quality images are more ubiquitous. Processing low-quality image, however, poses greater challenges. First, the definition of being low-quality is vague as there are numerous reasons that lead to a flawed image such as taking photos using a cheap webcam, hardware failure during capture, storage, transmission and so on. Many flaws can degrade the quality of an image such as low resolution, noise pixels, over and underexposure etc. There is no panacea for processing all kinds of low-quality images so far and it is hardly possible in the future due to the complicate nature of the problem.

A. Related works

Previous researchers have done embryonic works on stressing noise image processing and proposing experimental solutions. For example, authors in [5] show several networks including VGG16, GoogleNet, VGG-CNN-S and Caffe Reference are susceptible to quality distortions, particularly to blur and noise. In [6], authors investigate the relationship between object recognition and image quality and their results show that the selected algorithms can estimate recognition performance under certain conditions.

Many achievements have been made in biometrics, particularly fingerprint and face recognition. In [7], authors propose a two-stage enhancement scheme that handles low-quality fingerprint images. In [8], authors' investigations indicate that both hand-crafted and deep-learning based face detectors are not robust enough for low-quality images. In [9], authors concluded several techniques to improve the performance of low-quality face recognition (LQFR) such as super-resolution processing, deblurring and learning a relationship between different resolution domains. In [10], authors proposed a framework for recognizing objects in very low resolution images through the collaborative learning of two deep neural networks including image enhancement network and object recognition network.

B. Contributions

In this paper, we focus on one specific kind of noise that degrades the image quality called salt and pepper noise. We explore measures to improve the accuracy of salt and pepper noise image classification for several convolutional neural networks. A pre-processing technique called median filter is widely adopted to remove salt and pepper noise in both greyscale and RGB images. In [11], [12], [13], authors emphasize that median filter is a great general-purposed salt and pepper removal measure. More works, such as [14], [15], [16], focus on optimizing the median filter as determining the median of a set of numbers is an inefficient process. Our

method directly retrains the CNN model on salt and pepper noise data of various intensities. We apply this method on iconic ResNet-18 and VGG16 architectures because ResNet-18 is the first network beating human's recognition level in ILSVRC and VGG16 was proved surprisingly resilient to many distortions as shown in [5]. There is a concept in computer science called GIGO, thus garbage in, garbage out. GIGO describes that flawed input generates useless output. It is true in most cases in machine learning, but if the useful information can be extracted in the flawed input, the output could be meaningful. Our results prove that the noise-trained models obtain solid resilience to salt and pepper noise data and outperform the popular median filter when the noise is intense in the image.

The rest of the paper is organized as follows. Section II provides preliminary knowledge of salt and pepper noise, median filter, deep convolutional networks, image classification and transfer learning. Section III describes how we construct noise data and develop CNN models. In section IV, we present our experiments, results and detailed analysis. Conclusions and future research directions are discussed in section V.

II. BACKGROUND

A. Salt and pepper noise

We focus on salt and pepper noise that degrades the quality of an image in this paper. An image is stored in the form of an array of pixel values and each value, ranging from 0 to 255, denotes the intensity in the current color channel. Two common image color models are greyscale and RGB. In a greyscale image, it has $H \times W$ pixels, where H and W stands for the height and width of the image. Each pixel value reflects the intensity of greyness as 0 being black and 255 being white. On the other hand, since RGB is an additive color model, an RGB image is the product of stacking three color channels and each channel describes how red, green and blue of every pixel. Taking the red channel as an example, the pixel value ranges from 0 to 255 with 0 being black and 255 being red. Therefore, an RGB image is a 3D array with dimension $H \times W \times C$ where C stands for channel and is fixed to 3.

Salt and pepper noise is a type of impulse noise. When the original pixel value is lost and set to an extreme value, thus either 0 or 255, it becomes a salt and pepper noise pixel. When it happens in a grayscale image, the noise appears as either a white or black pixel, thus how the noise gets its name from. Salt and pepper noise can happen in an RGB image as well and the color of a noise pixel is more complicated. Since there are three channels, a pixel is denoted as a tuple with three values, thus (r, g, b). When the noise occurs to a pixel, all three channels lose their values and be set to either 0 or 255. It creates a noise pixel with eight possible colors (i.e. white $(255, 255, 255)$, cyan $(0, 255, 255)$, magenta $(255, 0, 255)$, yellow $(255, 255, 0)$, red $(255, 0, 0)$, green $(0, 255, 0)$, blue $(0, 0, 255)$ and black $(0, 0, 0)$). The salt and pepper noise pattern in the aforementioned two cases are shown in Fig. 1. In Fig. 2, we add salt and pepper noise with intensity equals to 0.1 and 0.5, denoted as SNP_0.1 and SNP_0.5, to an image. This example shows how sparse and intense salt and pepper noise affects the image quality. For simplicity, salt and pepper is referred to as SNP.

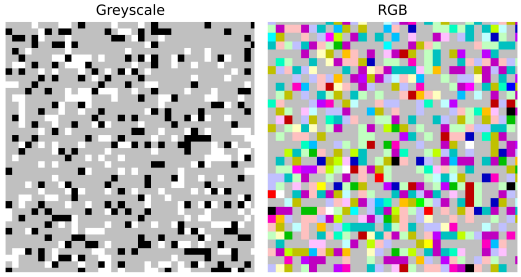

Figure 1: Salt and pepper pixels in greyscale and RGB images

Figure 2: An original image and two salt and pepper noise images of intensity 0.1 and 0.5

B. Median filter

Median filter is one of the most popular noise reduction techniques in digital image processing. For salt and pepper noise, it is particularly effective [17]. To apply a median filter to an image, which is denoted as a 2D array, we need to specify the median filter size, or kernel size, as $k \times k$ where k is an odd number by convention. This kernel traverses the image array from left to right and top to the bottom with a step size of one pixel. It first sweeps horizontally and when the whole row is covered, it moves one pixel down and starts again from the leftmost position. In each move, the kernel covers k^2 pixels and output the median value among them. Fig. 3 explains this process using a small proportion of an image. On the left is a 7×7 pixel area of an input image, SNP noise pixels are marked with a red box and the 3×3 kernel locates at its initial position in this area. The output of the current kernel, 129, is the median of the covered pixel values. After the kernel traversing the whole area, we have the output image on the right and all noise pixels are filtered and removed. The shrinkage of output image size puts a spotlight on the boundary issue of median filtering. There are multiple methods to mitigate this problem and maintain the image size such as padding a circumference using 0s or the pixel values on the boundary. Other schemes may be preferred in particular circumstances.

Compared to other filters such as mean filer, which replaces the pixel value with the mean of its neighbours', median filter shows more robustness in SNP noise removal because

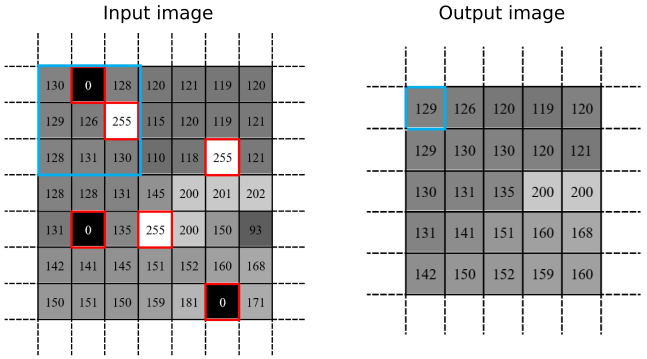

Figure 3: Median filter: an example

unrepresentative pixel values deteriorate the mean value while barely affect the median value. SNP noise pixels have two extremely unrepresentative values as 0 and 255. Moreover, median filter guarantees the output pixels come from the original image and does not create unnatural-color pixels. Fig. 4 shows the restoration ability of median filter of different kernel sizes on images with three levels of SNP noise. When the SNP noise intensity is 0.1, the median filter of kernel size 3 removes almost all the noise pixels and almost perfectly restore it back to the original looking. However, when the noise intensity increases, noise pixels may not be removed entirely with smaller kernels. Tuning up the kernel size does get rid of all the noise pixels but sacrifices sharpness of the image. In other words, restored image becomes more blurry and vague with larger kernel size.

C. Deep convolutional networks and image classification

A concise introduction to deep neural network has been provided in [5]. The big picture of deep learning, convolutional neural network, image understanding with deep convolutional networks were discussed in [18]. In this section, we introduce the structure of two specific neural network architectures that we experiment on: ResNet-18 and VGG16. Both are popular CNN models that achieved top results for image classification in ILSVRC. There are four kinds of basic blocks, or residual blocks in ResNet-18 [3] and skip connections are adopted to address the vanishing or exploding gradient problem. VGG16 has an architecture of sixteen layers including thirteen convolutional layers and three fully connected layers [19]. The layer-wise details of ResNet-18 and VGG16 are presented in Table I and II where k, s, p and OC stand for kernel size, stride, padding and output channels.

D. Transfer learning

Existing CNN models can be modified and retrained to solve particular problems by transfer learning. Transfer learning reuses knowledge from past related tasks to ease the process of learning to perform a new task [20]. A CNN model can be roughly viewed as the combination of a feature extractor and a classifier. When performing image tasks, the feature extractor in CNN transforms pixel values of an image into a suitable

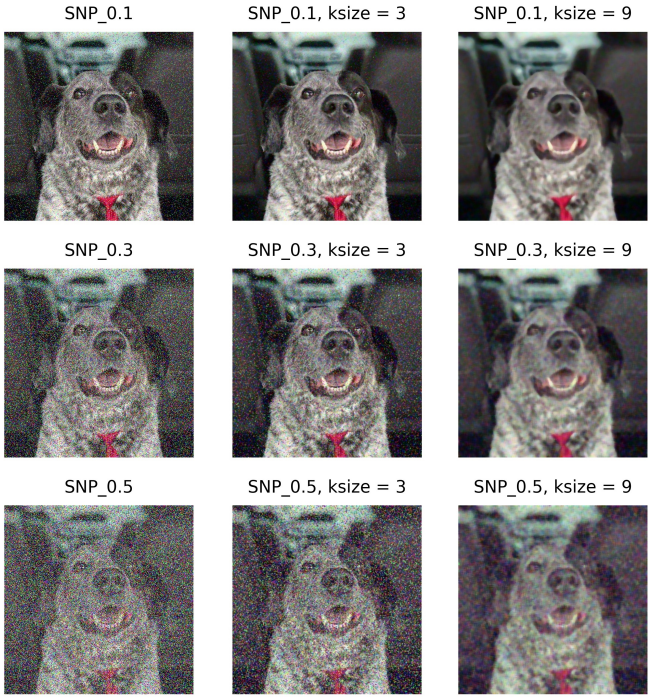

Figure 4: Applying median filter of kernel size 3 and 9 to SNP noise image of intensity 0.1, 0.3 and 0.5

Table I: Layer-wise details of ResNet-18: BB_m_n stands for the n-th appearance of basic block m. Downsampling is performed by BB_2_1, BB_3_1 and BB_4_1 with s=2.

#	Layer		Input size	k	s	p	OC
1	Conv		$224 \times 224 \times 3$	7	2	2	64
-	MaxPool		$112 \times 112 \times 64$	3	2	1	64
2	BB_1_1	Conv	$56 \times 56 \times 64$	3	1	1	64
3		Conv	$56 \times 56 \times 64$	3	1	1	64
4	BB_1_2	Conv	$56 \times 56 \times 64$	3	1	1	64
5		Conv	$56 \times 56 \times 64$	3	1	1	64
6	BB_2_1	Conv	$56 \times 56 \times 64$	3	2	1	128
7		Conv	$28 \times 28 \times 128$	3	1	1	128
8	BB_2_2	Conv	$28 \times 28 \times 128$	3	1	1	128
9		Conv	$28 \times 28 \times 128$	3	1	1	128
10	BB_3_1	Conv	$28 \times 28 \times 128$	3	2	1	256
11		Conv	$14 \times 14 \times 256$	3	1	1	256
12	BB_3_2	Conv	$14 \times 14 \times 256$	3	1	1	256
13		Conv	$14 \times 14 \times 256$	3	1	1	256
14	BB_4_1	Conv	$14 \times 14 \times 256$	3	2	1	512
15		Conv	$7 \times 7 \times 512$	3	1	1	512
16	BB_4_2	Conv	$7 \times 7 \times 512$	3	1	1	512
17		Conv	$7 \times 7 \times 512$	3	1	1	512
-	AvgPool		$7 \times 7 \times 512$	7	7	0	512
18	FC		$1 \times 1 \times 512$	-	-	-	1000

Table II: Layer-wise details of VGG16

#	Layer	Input size	k	s	p	OC
1	Conv	$224 \times 224 \times 3$	3	1	1	64
2	Conv	$224 \times 224 \times 64$	3	1	1	64
-	MaxPool	$224 \times 224 \times 64$	2	2	0	128
3	Conv	$112 \times 112 \times 128$	3	1	1	128
4	Conv	$112 \times 112 \times 128$	3	1	1	128
-	MaxPool	$112 \times 112 \times 128$	2	2	0	256
5	Conv	$56 \times 56 \times 256$	3	1	1	256
6	Conv	$56 \times 56 \times 256$	3	1	1	256
7	Conv	$56 \times 56 \times 256$	3	1	1	256
-	MaxPool	$56 \times 56 \times 256$	2	2	0	512
8	Conv	$28 \times 28 \times 512$	3	1	1	512
9	Conv	$28 \times 28 \times 512$	3	1	1	512
10	Conv	$28 \times 28 \times 512$	3	1	1	512
-	MaxPool	$28 \times 28 \times 512$	2	2	0	512
11	Conv	$14 \times 14 \times 512$	3	1	1	512
12	Conv	$14 \times 14 \times 512$	3	1	1	512
13	Conv	$14 \times 14 \times 512$	3	1	1	512
-	MaxPool	$14 \times 14 \times 512$	2	2	0	512
14	FC	$1 \times 1 \times 25088$	-	-	-	4096
15	FC	$1 \times 1 \times 4096$	-	-	-	4096
16	FC	$1 \times 1 \times 4096$	-	-	-	1000

feature vector so that the classifier could detect and classify patterns in the input image [18]. Since it requires very careful engineering and considerable domain expertise to design a high-quality feature extractor, one could take advantage of existing pretrained CNN models and apply to the similar tasks instead of randomly initializing a model and training it from scratch. Fig. 5 presents the idea of transfer learning. Based on a pretrained CNN model CNN_x, CNN_y shares the same feature extractor architecture but has its own classifier. To tune the model CNN_y, it requires retraining using the data in the specific task.

Feature extraction and fine-tuning are two types of transfer learning. In feature extraction, CNN_y shares the entire feature extractor in CNN_x, including its architecture and weights, thus Feature Extractor_y = Feature Extractor_x. And the weights of its feature extractor do not change during the retraining process. In fine-tuning, Feature Extractor_y is initialized using the weights in Feature Extractor_x and its weights update in the retraining process.

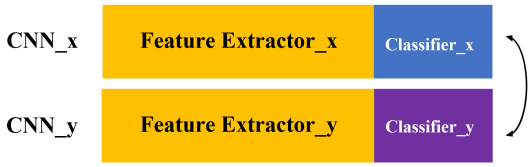

Figure 5: Transfer learning between two CNNs

III. CONSTRUCTING CNN MODELS

To explore the effectiveness of training on noise data and evaluate the noise resilience of noise-trained models, we first generate training, validation and test sets by adding five levels of SNP noise to a subset of ImageNet. Then we retrain multiple CNN models using transfer learning based on pretrained models for both ResNet-18 and VGG16. The classification accuracy of a tested model is defined as one minus its top-1 error. We provide the classification accuracy of each model on various test sets and form our conclusions. In this section, we present the details of constructing noise datasets and developing CNN models.

To prepare datasets, we randomly select five objects, each containing $1,300$ sample images, from ImageNet: goldfinch, hamster, vizsla, upright and pitcher. The proportions of the original sample images for training, validation and test purpose are 80%, 10% and 10%, thus $1,040$ training images, 130 validation images and 130 test images for each object. Next, we add SNP noise of intensity 0.1–0.5 to each original sample image for every purpose and yield fifteen noise sets as shown in Table III.

Table III: Dataset details: In the column named "Dataset Name", the purpose of each set is conveyed by its name. For SNP noise sets, $0.x$ is the noise intensity where x is an integer ranging from 1 to 5. In the column named "Configuration", it shows the number of images and objects in each set. e.g. $1,040 \times 5$ means the set contains 5 objects and each has $1,040$ images, thus a total of $5,200$ images.

Dataset Name	Contents	Configuration
Original_train	Original images	$1,040 \times 5$
Original_val		130×5
Original_test		130×5
SNP_$0.x$_train	Images with SNP_$0.x$ noise	$1,040 \times 5$
SNP_$0.x$_val		130×5
SNP_$0.x$_test		130×5

Table IV shows that based on the pretrained model, six retrained models are developed: RtO, RtN_$0.x$ where x is an integer ranging from 1 to 5. Since the size of our training set $(5,200)$ is not comparable to that of the pretrained training set (over 1.2 million), feature extraction is preferred over fine-tuning to avoid potential overfitting issues. Therefore, all retrained models only have a modified classifier but share the same feature extractor with the pretrained model. For both ResNet-18 and VGG16, the output dimension of the last layer is changed from $1,000$ to 5. In the retraining process, only the weights of the classifier are updated.

Table IV: Pretrained model and retrained model details: "RtO" stands for **Re**trianed on **O**riginal images. A noise-trained model is denoted by"RtN", which is short for **Re**trianed on **N**oise images. All retrained models are developed from the pretrained model using transfer learning. Acronym "OD" is output dimension, thus the number of output channels of the last fully connected layer in the model.

Model Name	Based on	Retrained on	OD
Pretrained	-	-	$1,000$
RtO	Pretrained	Original_train	5
RtN_$0.x$	Pretrained	SNP_$0.x$_train	5

IV. EXPERIMENTS AND RESULTS

We design experiments to show the classification accuracy improvement brought by transfer learning and the performance

comparisons among multiple retrained models. Comprehensive results show the superiority of different models in particular scenarios. Below we present the design details, results and analysis of each experiment.

A. Pretrained model and RtO

In this experiment, we test the classification accuracy of the pretrained model and RtO on all six test sets: Original_test and SNP_0.x_test, where x is an integer ranging from 1 to 5. Differences between two models are the output dimension of classifier and the classifier of RtO is retrained with Original_train.

The accuracy comparison between two models are shown in Fig. 6. We notice that the classification accuracy reduces as the SNP noise intensity increases, which is an expected outcome as neither model has ever seen a noise image. However, the performance of the pretrained model drops sharply, which indicates its more significant lack of robustness on SNP noise data. Benefiting from transfer learning, RtO reduces the possible output to 5 from $1,000$ and gets about 20% accuracy improvement immediately. Another obstacle for the pretrained model to produce the right classification is the interference by similar objects such as the Vizsla and the Great Dane. As a result, RtO is an obvious winner on all test sets compared to the pretrained model and the performance gap tends to increase with the noise intensity: 19.54%, 37.54%, 54.46%, 59.54%, 58.31% and 50.46%.

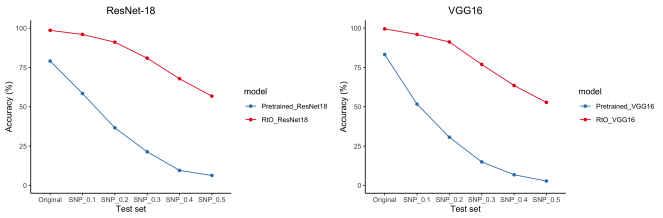

Figure 6: A classification accuracy comparison between the pretrained model and RtO

This experiment shows the great performance leap after applying transfer learning on the pretrained model. Therefore, we take RtO as the baseline model to compare the performance with in the rest of the experiments.

B. RtO, RtO_MF, and RtN

In this experiment, we compare the classification accuracy among RtO and RtN models on multiple test sets. It is divided into three parts and each shows one of the following bullet:

- The accuracy improvement of RtO with and without median filter on noise images
- The performance gain of noise-trained models over RtO
- The comparison between applying median filter and training on noise data in particular scenarios

The selective classification accuracy numbers of ResNet-18 and VGG16 models are presented in Table V and VI. We omitted "_train" in the test set name for simplicity. "RtO_MF" indicates the median filter is applied to the noise test sets before feeding it to RtO. Median filter has four kernel size options: 3, 5, 7 and 9. There are five RtN models and each was retrained on a single intensity of SNP noise data as mentioned in Table IV. Therefore, we list the worst and the best accuracy of RtO_MF and RtN in the tables as well. Below we present the details of each part of this experiment.

Table V: Classification accuracy of ResNet-18 RtO and RtN models, including median filters and RtN models of the worst and best performance

Test set	RtO	RtO_MF		RtN	
		worst	best	worst	best
Original	98.62%	NA	NA	27.85%	75.54%
SNP_0.1	96.00%	96.31%	99.08%	36.46%	97.39%
SNP_0.2	91.08%	94.62%	95.39%	51.69%	95.08%
SNP_0.3	80.92%	88.00%	91.69%	62%	91.85%
SNP_0.4	67.85%	80.00%	83.54%	76%	90.00%
SNP_0.5	56.77%	61.54%	78.15%	72.92%	85.39%

Table VI: Classification accuracy of VGG16 RtO and RtN models, including median filters and RtN models of the worst and best performance

Test set	RtO	RtO_MF		RtN	
		worst	best	worst	best
Original	99.54%	NA	NA	80.00%	99.39%
SNP_0.1	96.00%	97.08%	98.77%	87.69%	98.46%
SNP_0.2	91.23%	93.54%	96.77%	87.39%	96.00%
SNP_0.3	76.92%	90.00%	91.39%	86.62%	91.08%
SNP_0.4	63.54%	75.69%	80.15%	75.49%	83.08%
SNP_0.5	52.77%	62.00%	66.92%	62.46%	78.15%

a) RtO vs. RtO_MF: In the first part of the experiment, we compare the classification accuracy of RtO models before and after applying different median filters to the noise test sets. As mentioned in section II, there is a trade-off between the unfiltered noise pixels and the sharpness of the restored image. Thus, the larger the kernel size is, more noise pixels will be removed but the output image will be more blurry. Previous work [5] proved that blur in an image will create, if not more, comparable difficulties for CNNs to process as noise does. Since all the images are from ImageNet, the average resolution is roughly 480×410 and if the kernel size of the applied median filter is larger than 11, the restored image loses majority of sharpness when the noise intensity is high. Therefore, the potential kernel sizes are carefully selected as 3, 5, 7 and 9.

We take the RtO model, which is retrained on a set of original images, and test its accuracy on five SNP noise sets before and after applying the median filter of each selected kernel size. We record the accuracy changes with the noise intensity in all five scenarios as shown in Fig. 7. When the SNP noise intensity is 0.1, median filter makes little accuracy difference to RtO. Generally, median filter improves the classification accuracy by 9.54%–14.04% for ResNet-18 RtO and 11.35%–15.5% for VGG16 RtO when obvious noise pixels (intensity ≥ 0.3) are observed in an image. Nevertheless, ResNet-18 RtO

is more sensitive to the kernel size change while VGG16 RtO shows more immunity to it.

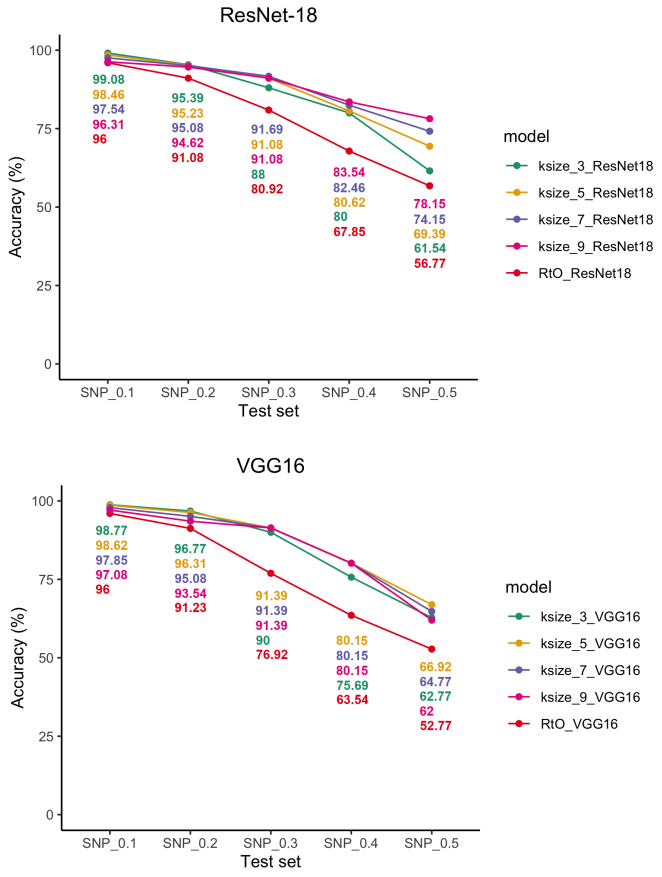

Figure 7: Comparisons of RtO accuracies on noise data with and without applying median filters

b) RtO vs. RtN: In the second part, we compare the classification accuracy among RtN models and the RtO model on all original and noise test sets. As its has been observed that RtO performs poorly on noise test sets, we are interested in how RtN performs on 1) the original test set 2) the noise set of the model's training intensity 3) noise sets other than the model's training intensity. Below we answer all of these questions in the perspective of CNN.

The upper line chart in Fig. 8 shows the accuracy of ResNet-18 RtO and RtN models. All RtN models perform poorly on original images, especially RtN_0.5, its accuracy (27.85%) is merely better than the expected value of a random guess (20%). Only RtN_0.1 outperforms RtO on set SNP_0.1, SNP_0.2 and the rest RtN models show disastrous performance on these two test sets. However, as the noise level increases, more RtN models perform better RtO and they have a decent classification accuracy averaging 89.26% on noise images of their training intensities. When the noise intensity reaches 0.5, RtN models achieve an accuracy ranging from 72.92% to 85.39% compared to RtO's 56.77%. To conclude, all ResNet-18 RtN models have a better classification accuracy as the noise intensity increases and finally all of them outperform the RtO model. They have a very decent performance on the noise images of their training intensities, an approximately satisfying performance when the test noise intensity is slightly deviated but if the test image is much more or less noisy than the data the model has ever seen, the result is unpredictable, usually disappointing.

On the contrary, VGG16 RtN models have quite different behavior as shown in the lower chart in Fig. 8. Every RtN model except RtN_0.5 either has a comparable performance or crushes RtO model in each test case. They all have a performance line that drops steadily and not much fluctuation occurs especially for RtN_0.5 model. Therefore, a claim can be made that RtO can be entirely replaced by RtN_0.3 as the latter model triumphs in all scenarios. Unfortunately, this claim cannot be made easily in the ResNet-18 case.

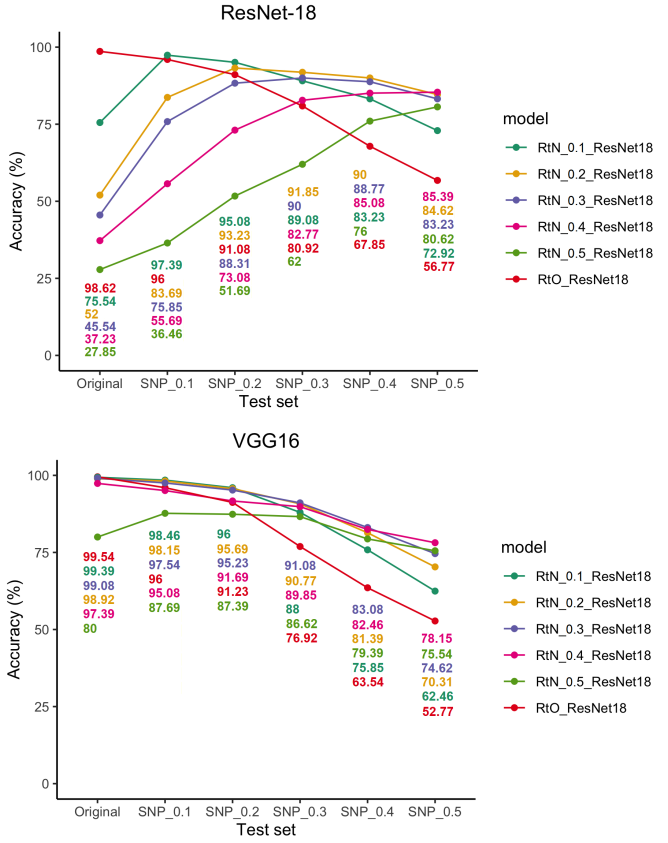

Figure 8: Accuracy comparisons among RtO and RtN models

When the intensity of SNP noise is below 0.3, RtO models produce a solid accuracy as the sparse noise pixels do not degrade the feature extraction process to a large extent. The overall performance of VGG16 RtN models is surprisingly good because noise-trained models have never seen a noise-free image. However, the performance of ResNet-18 RtN models in this scenario is completely unacceptable. When the intensity increases to 0.3 and above, noise pixels start to dominate the image. The performance of RtO drops quickly while noise-trained models begin to show its superiority in

noise image processing for both CNNs as expected. This experiment shows that noise training is more suitable for some CNNs such as VGG16. Because their noise-trained models have a more stable and decent performance. On the contrary, noise training may bring unpredictable outcomes to other CNNs such as ResNet-18. Very careful considerations should be taken before doing noise training on a CNN.

c) RtO_MF vs. RtN: We discussed the advantage of applying median filters and noise training in the first two parts of the experiment. In the last part, we analyze these two methods in terms of their worst and best accuracies on each noise set compared to RtO. Table VII shows the worst and the best median filter and RtN model selections. For median filter, the best kernel size option tends to increase with higher noise intensity. A mixture of unfiltered noise and blur generated by the filter is usually the worst case, which can happen when the kernel size is small and the noise intensity is high. For noise-trained models, the worst one is often extremely-trained, either RtN_0.1 or RtN_0.5. The best accuracy is always achieved by the model that was trained on the noise data with a slightly milder intensity.

Table VII: The best and the worst kernel size and RtN model selections: "SNP" is omitted for noise sets except SNP_0.1 due to the table width limitation. "w" and "b" stand for worst and best respectively. For MF (median filter), the entry is the kernel size. For RtN, the entry is the noise intensity of its training set. Multiple values in an entry indicates a tie.

Model			**SNP_0.1**	**_0.2**	**_0.3**	**_0.4**	**_0.5**
ResNet-18	MF	w	9	9	3	3	3
		b	3	3	7	9	9
	RtN	w	0.5	0.5	0.5	0.5	0.1
		b	0.1	0.1	0.2	0.2	0.4
VGG16	MF	w	9	9	3	3	9
		b	3	3	5, 7, 9	5, 7, 9	5
	RtN	w	0.5	0.5	0.5	0.1	0.1
		b	0.1	0.1	0.3	0.3	0.4

We plot the extreme accuracies of RtO_MF and noise-trained models on each noise set in Fig. 9. Regarding the worst performance of RtO_MF and RtN, the former has a more stable and reasonable accuracy line. Noise-trained models do not catch up until the image is full of noise. However, the best performance lines tell a different story. For both CNNs, noise-trained models produce a comparable accuracy when the noise intensity is below 0.4 and shows clear advantage over RtO_MF when the noise intensity continues to grow. In conclusion, when the noise is sparse in the image, median filter is a more reliable method. When the noise is intense, noise training is a more promising choice to increase the classification accuracy.

V. CONCLUSION

Our experiments substantiate that noise training improves the performance on SNP noise image classification. The noise-trained models show more resilience to SNP noise as the intensity increases. They outperform median filter when the noise pixels dominate the image according to our results. Moreover, the noise-trained models of different CNNs vary in

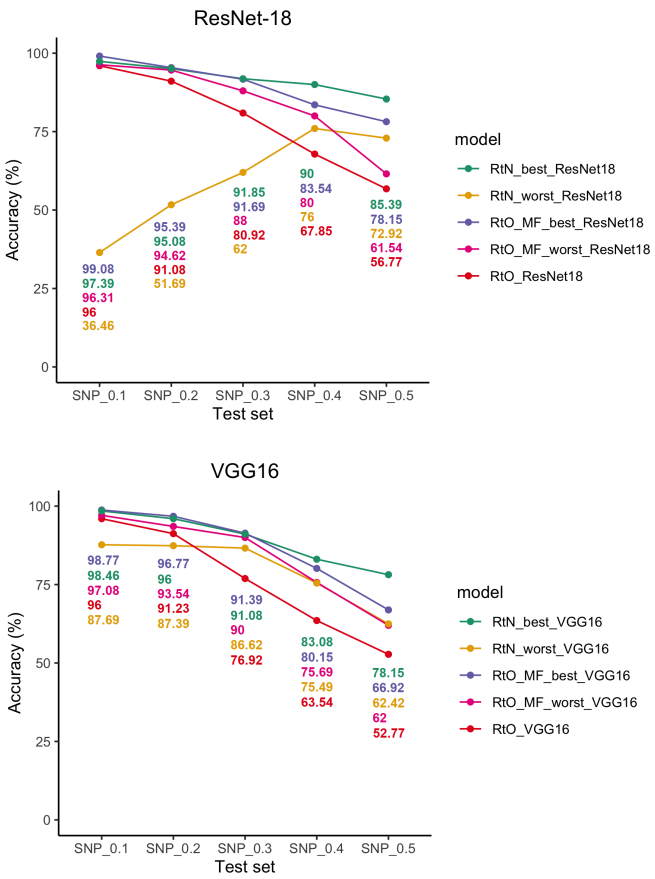

Figure 9: Comparisons among the best and the worst accuracies of RtN and RtO_MF

the classification accuracy on images with sparse noise or free of noise. VGG16 benefits more from noise training as most of its RtN models generate solid accuracy numbers on noise images and little accuracy sacrifice on high-quality images. To conclude, when the noise is sparse, median filter remains reliable, effective and preferred. When the noise is intense, noise training brings more possibilities and surprises.

For future research directions, we will extend this work by adding more objects and increase the output dimension up to $1,000$ and create a complete noise-proof CNN. We will also explore how other CNNs benefit from noise training by bringing in classic and state-of-art architectures such as AlexNet [21], SqueezeNet [22], EfficientNet [23], Mobilenetv2 [24], NFNet-F4 [25] etc. Table VII shows most of the worst RtN models are trained on extreme noise data. Another research idea is that training the model on a mixture of noise data with multiple intensities or a mixture of both original and noise data. It helps forming new concepts such as mix training. We also plan to test if other noise types can be generalized and learned through training such as Gaussian noise and speckle noise. Taking a step further, other image flaws such as partial evidence, low resolution, over and underexposure can be explored for their generalization potentials.

ACKNOWLEDGMENT

THE RESEARCH REPORTED IN THIS PAPER IS PARTIALLY SUPPORTED BY THE PHILIP AND VIRGINIA SPROUL PROFESSORSHIP AT IOWA STATE UNIVERSITY.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (3) (2015) 211–252.

[3] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[4] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, et al., Florence: A new foundation model for computer vision, arXiv preprint arXiv:2111.11432 (2021).

[5] S. Dodge, L. Karam, Understanding how image quality affects deep neural networks, in: 2016 eighth international conference on quality of multimedia experience (QoMEX), IEEE, 2016, pp. 1–6.

[6] D. Temel, J. Lee, G. AlRegib, Cure-or: Challenging unreal and real environments for object recognition, in: 2018 17th IEEE international conference on machine learning and applications (ICMLA), IEEE, 2018, pp. 137–144.

[7] J. Yang, N. Xiong, A. V. Vasilakos, Two-stage enhancement scheme for low-quality fingerprint images by learning from the images, IEEE transactions on human-machine systems 43 (2) (2012) 235–248.

[8] Y. Zhou, D. Liu, T. Huang, Survey of face detection on low-quality images, in: 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018), IEEE, 2018, pp. 769–773.

[9] P. Li, L. Prieto, D. Mery, P. Flynn, Face recognition in low quality images: a survey, arXiv preprint arXiv:1805.11519 (2018).

[10] J. Seo, H. Park, Object recognition in very low resolution images using deep collaborative learning, IEEE Access 7 (2019) 134071–134082.

[11] K. K. V. Toh, H. Ibrahim, M. N. Mahyuddin, Salt-and-pepper noise detection and reduction using fuzzy switching median filter, IEEE Transactions on Consumer Electronics 54 (4) (2008) 1956–1961.

[12] E. J. Leavline, D. A. A. G. Singh, Salt and pepper noise detection and removal in gray scale images: An experimental analysis, International Journal of Signal Processing, Image Processing and Pattern Recognition 6 (5) (2013) 343–352.

[13] B. Wang, Q. Xiang, Fast median filter image processing algorithm and its fpga implementation, Front Signal Process 4 (4) (2020) 88–94.

[14] B. Weiss, Fast median and bilateral filtering, in: ACM SIGGRAPH 2006 Papers, 2006, pp. 519–526.

[15] M.-H. Hsieh, F.-C. Cheng, M.-C. Shie, S.-J. Ruan, Fast and efficient median filter for removing 1–99% levels of salt-and-pepper noise in images, Engineering Applications of Artificial Intelligence 26 (4) (2013) 1333–1338.

[16] B. R. Jana, H. Thotakura, A. Baliyan, M. Sankararao, R. G. Deshmukh, S. R. Karanam, Pixel density based trimmed median filter for removal of noise from surface image, Applied Nanoscience (2021) 1–12.

[17] G. R. Arce, Nonlinear signal processing: a statistical approach, John Wiley & Sons, 2005.

[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[20] L. Yang, S. Hanneke, J. Carbonell, A theory of transfer learning with applications to active learning, Machine learning 90 (2) (2013) 161–189.

[21] A. Krizhevsky, One weird trick for parallelizing convolutional neural networks, arXiv preprint arXiv:1404.5997 (2014).

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016).

[23] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International conference on machine learning, PMLR, 2019, pp. 6105–6114.

[24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[25] A. Brock, S. De, S. L. Smith, K. Simonyan, High-performance large-scale image recognition without normalization, in: International Conference on Machine Learning, PMLR, 2021, pp. 1059–1071.