Connecting low-level image processing and higher-level vision for degraded image classification

Group 15 Project Final Report

Changyue Su
The University of Hong Kong
3035534854
scytina6@connect.hku.hk

Zhui Ding
The University of Hong Kong
3035533044
u3553304@connect.hku.hk

Kehan Long
The University of Hong Kong
3035771323
longkhan@connect.hku.hk

Haozhou Xu
The University of Hong Kong
3035771086
xhz1@connect.hku.hk

Yijiazhen Qi
The University of Hong Kong
3035773096
u3577309@connect.hku.hk

Abstract

Convolutional Neural Networks has shown robustness in image classification tasks. However, commonly used testing datasets in image classification are clean and of high quality. Given that images of real-world scenarios contain corruptions, basic CNN models may lose their robustness. To solve this problem, low-level vision methods that focus on restoring images are a good try. First, we train and test basic CNNs abilities on both regular clean data and corrupted data. Second, we concatenate the filtering idea from low-level vision to the basic CNN models for improving low-quality image classification performance. The experiment results show that improved CNNs with filtering activation function can achieve better results in low-quality image classification than the original CNNs.

1. Introduction

Recent years have witnessed tremendous and robust development in Computer Vision areas, especially in Image Classification [9]. The rapidly emerging image classification techniques have an increasing impact on our life. Some presentational applications are auto-driving, satellite object identification, medical imaging and machine controls. However, most current image classification models only focus on high-level vision, regardless of low-level features [6]. Low-level image processing can play a crucial role in image classification tasks by extracting fundamental image primitives. Connecting low-level image processing to high level vision is promising. An interesting direction is to use low-level vision to enhance image classification on degraded images [5]. Common state-of-the-art image classification methods largely rely on annotated datasets of high-quality images. Their performance on low-quality images is significantly degraded [7]. However, real-world vision data usually suffer from low resolution, noise, occlusion and motion blur. As a result, it is meaningful to tackle low-quality image classification problems. A low-level processing model can be used to improve image quality as well as extract low-level features for future high-level processing [6]. We believe it is feasible to combine low-level image processing with high-level CNN into a framework for enhancing degraded image classification.

In this work, we want to evaluate the performance of some baseline CNN models on both clean and degraded datasets. We further concatenate basic CNN models with low-level vision methods to improve accuracy on low-quality image classification. Experiment results show the lack of robustness in baseline CNNs on corrupted images, and the efficacy of the newly proposed methods.
To summarize, we contribute:

1. implement the three classical image classification models on the selected dataset.
2. verify that classical CNN image classification methods suffer significant degradation on low-quality images.
3. implement a method that combines low-level image processing with image classification networks to achieve better low-quality image classification results.

2. Related Works

2.1. Image classification models

In recent years, Convolutional Neural Networks has become one of the strongest proponents of Deep Learning. One popular application of these Convolutional Networks is Image Classification. Classical image classification models are AlexNet[4], VGG[5] and ResNet[6], respectively. These three methods are commonly used for the classification task. They can achieve good enough results on various datasets.

2.2. Low-quality image classification

Many works have been done to mitigate the drop in accuracy for low-quality image classification [5]. One direction is to transfer this problem into two-stage tasks [3]. First, restore and enhance the degraded images, and then apply image classification. Another method is to make domain adaptation [12]. By using adversarial learning or kernelized training, this approach aims at matching the marginal distribution of low- and high-quality images [5]. A more intuitive solution is fine-tuning the degraded images [6].

2.3. Activation Function

In this work, we focus on improving the performance by changing the activation function in the model. We argue that activation functions can substantially enhance CNN's robustness against corruption if designed with specific parameters [14]. It can be achieved by replacing the widely used ReLU function with LP-ReLU functions, which are functions with filtering help reduce the noise and compact the features. Also, by using the Discrete Cosine Transform (DCT), the low-frequency corruptions in the data can be further reduced [14].

3. Baseline Models

After downloading the Oxford17 flowers dataset, we first separated images into 17 category folders with 80 images each. As the dataset is class-balanced, we skipped the under-sampling /over-sampling step, which is used to resolve the class-imbalance issue. After that, we resized the images to 64*64 and converted them to a torch. Float Tensor of shape (C x H x W) in the range [0.0, 1.0] using {transforms. Resize} and {transforms. ToTensor}. The images in each class were then randomly divided into sub-training sets, sub-validation sets, and sub-test sets with a ratio of 8:1:1, and combined into overall training, validation, and test sets with 1088, 136, and 136 images respectively. We tuned our hyper-parameters on the training and validation sets and assessed the model performance on the test set.

3.1. Baseline Models Implementation

We implemented 3 baseline model architectures, AlexNet [4], VGG [5], and ResNet [6] in our experiments. Since we mainly focus on fine-tuning the baseline models

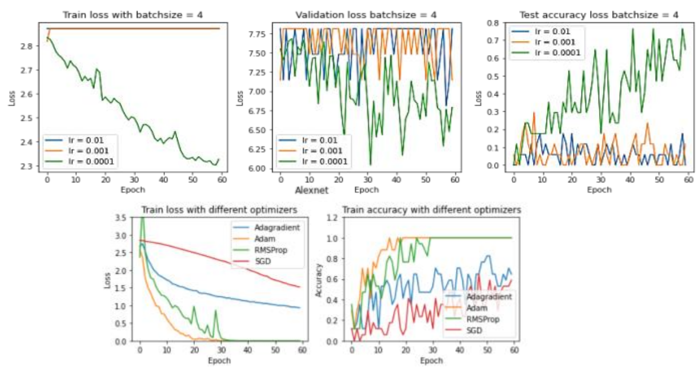


Figure 1. AlexNet
(Above: Learning Rate; Below: Optimizer)

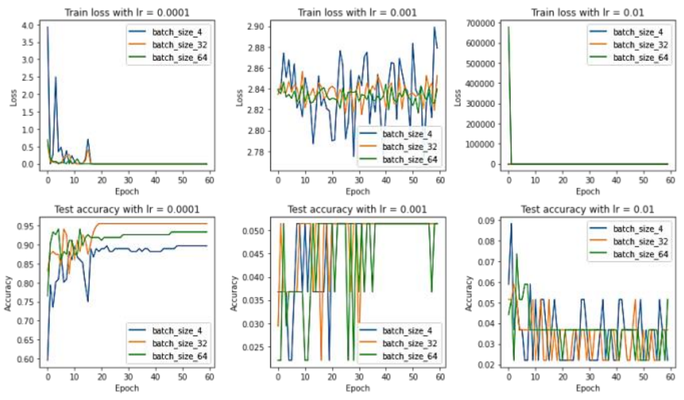


Figure 2. VGG16

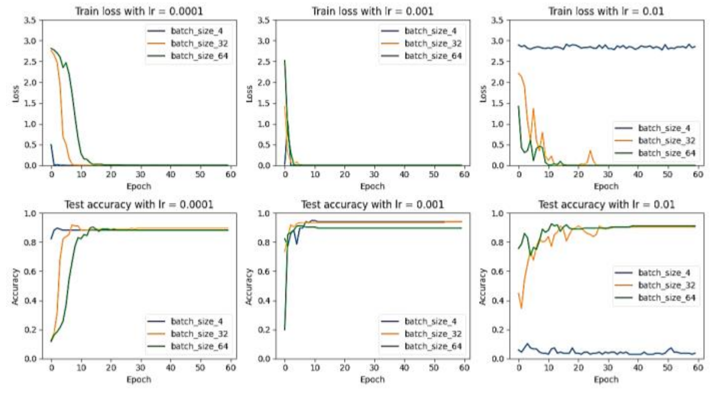


Figure 3. VGG19

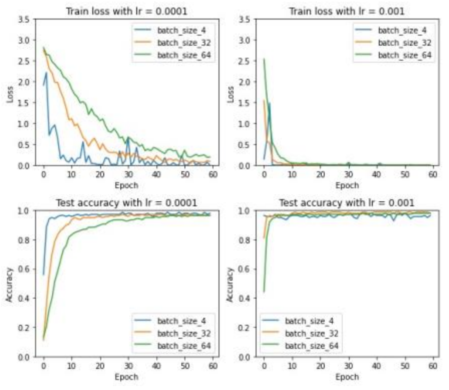


Figure 4. Resnet50

for our new tasks in this part, the models in our experiments are all from PyTorch torchvision pre-trained models. All of the torchvision models have been trained on 1000-class ImageNet dataset. With the torchvision pre-trained weights, we retrain AlexNet, VGG16, VGG19, ResNet34, ResNet50 models and update their parameters for our classification tasks. It is expected to have higher test accuracy and lower training loss by using pre-trained models.

3.2. Training Detail & Experiment Result

We have investigated the influence of **learning rate** and **Optimizers** on the performance of **AlexNet.** For **learning rate**, we train on batch size 4 and Adam optimizer. Compared to the fluctuating pattern in both loss and accuracy in learning rates 1e-2 and 1e-3 (blue & orange line in Figure 1 above), 1e-4 (green line) effectively helps to reduce loss and improve accuracy. For **optimizers**, we trained on Adam, Adagrad, RMSProp, and SGD, with batch size 64 and learning rate 1e-3. The experiment results, as Figure 1 below, show that Adam gets the best performance in reducing the training loss and is the first one to reach convergence, while SGD gets a not-that-good performance.

For **VGG16**, we investigated how well learning rates of 1e-2, 1e-3, and 1e-4 performed on the batch sizes 4, 32, and 64 in epoch 60 on the Adam optimizer (Figure 2). It has been observed that learning rate 1e-4 has the best performance in both loss and accuracy results. Regarding the training and validation losses, learning rates 1e-2 and 1e-3 had terrible performances with an ending loss of around 3, while 1e-4 performed extremely well and had a loss near 0 in the last epoch. In terms of the testing accuracy, we can find 1e-2 fluctuates wildly between 0.02 and 0.03. Though 1e-3 performs better than 1e-2, it still varies between 0.04 and 0.05. 1e-4 did the best job with a rising testing accuracy trend and can reach 0.95 in the last epoch.

For the **VGG19**, we investigate the factors mentioned in the VGG16 model again. We also want to check if the pre-train technique can help improve the training result compared to all parameters initialed to be 0. As shown in Figure 3. For the VGG19 model, a learning rate at 1e-3 level will be the most suitable. It has a higher converging rate and a similar result compared to other rates. Also, a batch size of around 32 will be most suitable for stability and training. Fewer batch sizes with higher learning rates will result in no learning performance. We also tested the non-pretrain model, the value hasn't reached the actual proper parameters after convergency. It reflects the importance of the setting of initial parameters.

For **Resnet34**, our chosen learning rates are 1e-3. Under the learning rate, we train the ResNet50 with batch size 4, 32, and 64 for 60 epochs. The optimization uses stochastic gradient descent (SGD) and the loss function is cross-entropy loss. As shown below, due to the adaptation of pre-trained model, all batch sizes have an accuracy higher than 0.9, which is indeed impressive. ResNet34 achieves better classification results with learning rate 1e-3 and batch size 4. However, batch size 4 had a sudden high loss during epoch 39, which is ten times more than other epochs (0.136679). Besides, although batch size 4 usually shows a better result, it doesn't have a strong trend to converge throughout epochs. Nevertheless, the performance of batch size 4 is still a lot better than batch size 64.

For **Resnet50**, our chosen learning rates are 1e-3 and 1e-4. Under each learning rate, we train the ResNet50 with pre-train weights under batch size 4, 32, and 64 for 60 epochs. The optimizer is stochastic gradient descent (SGD) and the loss function is cross-entropy loss. As shown in Fig. 4, ResNet50 achieves better classification results with

Model	Best (Learning Rate)	Best (Batch Size)	Training Loss	Testing Accuracy
VGG16	1e-4	4	5.36e-07	89.71%
VGG19	1e-3	32	6.45e-06	89.71%
Resnet34	1e-3	4	3.23e-03	98.53%
Resnet50	1e-3	4	3.39e-04	96.32%

Table 1. Baseline Model Performance

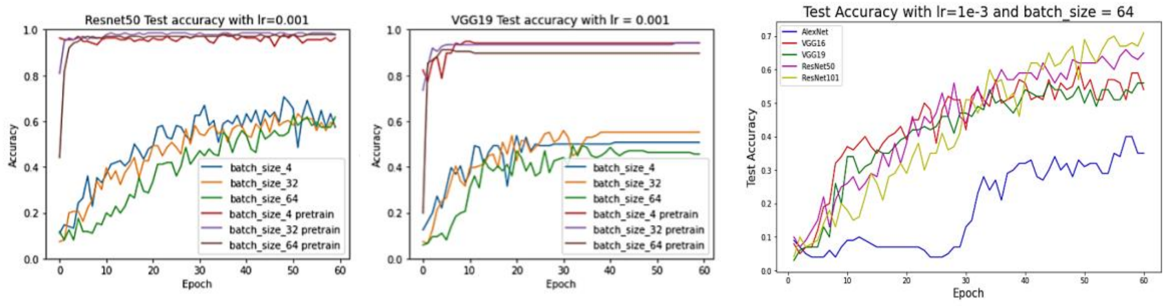


Figure5. Results of no pre-trained model

learning rate 1e-3 and batch size 4 or 32. Training with learning rate 1e-4 and batch size 64 gives the worst results. Since our model contains pre-trained weights, the test accuracy can reach above 90% easily.

In conclusion, we first tuned the hyper-parameters on AlexNet and found that the learning rate 1e-4 and Adam optimizer are the best. However, considering the inferior performance and limited improvement potential of AlexNet, we decided to focus on fine-tuning hyper-parameters for other models (VGG16, VGG19, Resnet34, Resnet50) to find the best baseline model for further improvements. Table 1 above summarizes the best hyper-parameters, training loss, and testing accuracy for all models. It was discovered that ResNet34 and ResNet50 all perform well, and we will investigate how to enhance ResNet50 model on degraded photos in the experiment section.

3.3. Impact of Pre-training

We also tried to investigate the Pytorch pre-train weights' influence on baseline models. Firstly, we randomly initialize the parameters. Then, We train AlexNet, VGG16, VGG19, ResNet50, and ResNet101 with learning rate 1e-3 and batch size 64 for 60 epochs. The results are in Figure 5 right. All the models cannot achieve test accuracy higher than 70% here. AlexNet gives much worse outcomes compared to other models. ResNet50 and ResNet101 get slightly better results in the end. In Figure 5 left and middle, we compare the test accuracy of pre-train models and no pre-train models. VGG19 test accuracy with pre-train weights decline and converge faster than VGG19 test accuracy without pre-train weights. Same for the test accuracy of Resnet50. Additionally, the accuracy pre-train model can achieve is higher than the no-pre-train. We can conclude that the model can receive much more favorable outcomes with the pre-train weights.

4. Improvement

In traditional low-level vision, a way to restore degraded images is filtering [3]. For example, some high-frequency corruptions, like gaussian or speckle noise, will by filtered out by a low-pass filter in classical signal processing [14]. However, directly applying filters before classification may not be a good way. First, traditional filters can drift away corrupted data to different feature space, even the restored results are correct in human vision [5]. Second, two-stage method (low-level stage and middle-level stage) is not efficient enough in many cases. To properly connect low-level vision into classification tasks, we believe changing activation function is a good choice, since activation function share some properties with image signal filters.

Inspired by the low-pass filter activation functions proposed by Hossain et al. [14], we concatenate their activation functions into ResNet50 replacing the widely used ReLU functions. This allows us to get compact features from input to reduce the impact of different noises [14]. We in total test 3 novel activation functions: Leaky-ReLU, LP-ReLU1, and LP-ReLU2. The dataset we choose to train our models is the CIFAR10 dataset, and the test dataset is CIFAR10-C dataset [7].

4.1. Dataset

CIFAR10-C dataset contains 15 algorithmically generated corruptions from noise, blur, weather, and digital categories, along with 4 extra corruptions (Speckle Noise, Gaussian Blur, Spatter, Saturate) to the test images in the CIFAR-10 dataset (See Appendix b) [7]. Each type of corruption has five levels of severity, with 10,000 images in each level.

4.2. Limits of Baseline models with ReLU

Three baseline models have been proven efficient in recognizing flowers in different categories, especially Resnet50 achieved the best performance with an average test accuracy of **96.8%**. However, they did not perform well after we applied the baseline models for the degraded image classification problem on Cifar10-C. We only achieve an average accuracy of **56.88%** for VGG16, **55.85%** for VGG19, and **45.8%** for resnet50, shown as Figure 6. They perform especially poorly on Impulse-noise and Saturated image.

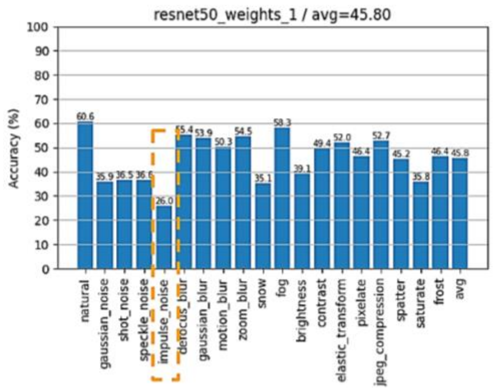


Figure 6. ResNet50 results on CIFAR10-C

4.3. Robustness of new activation functions with ResNet50

We customize all the common ReLUs in ResNet50 to our testing activation functions (LeakyReLU, LP_ReLU1, LP_ReLU2). Then we train ResNet50_LeakyReLU,

Model	Input Size	FLOPs	Number of parameters	Output Size
AlexNet	(3, 64, 64)	98.2433M	61.101M	(17, 1)
VGG16	(3, 32, 32)	437.517M	138.358M	(17, 1)
VGG19	(3, 32, 32)	533.505M	143.667M	(17, 1)
Resnet18	(3, 32, 32)	37.6735M	11.689M	(17, 1)
Resnet34	(3, 32, 32)	75.4862M	21.798M	(17, 1)
Resnet101	(3, 32, 32)	162.098M	44.549M	(17, 1)
Resnet50	(3, 32, 32)	84.06M	23.542M	(17, 1)
Resnet50_LeakyReLU	(3, 32, 32)	84.06M	23.542M	(17, 1)
Resnet50_LP_ReLU1	(3, 32, 32)	<= 84.56M	23.542M	(17, 1)
Resnet50_LP_ReLU2	(3, 32, 32)	<= 85.60M	23.542M	(17, 1)

Table 2. FLOP analysis on our models

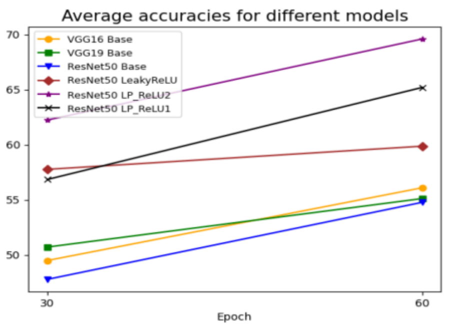


Figure 7. Models' average accuracy on CIFAR10-C

ResNet50_LP_ReLU1, ResNet50_LP_ReLU2 separately under CIFAR10 for 60 epochs, with learning rate 1e-3 and batch size 16. The optimizers are stochastic gradient descent (SGD), and the loss-functions are cross-entropy loss. In the end, we test the models on CIFAR10-C and averaging the results on different severity levels.

Leaky-ReLU is used to refine the "dead neuron" problem caused by normal ReLU functions. The problem is that ReLU blocks all negative input; thus, many neurons will never be activated without specific initialization before training. The leaky ReLU replaces the negative part of ReLU with a small constant slope **α**. In this project, we defined the **α** to be 0.01 as default. This will help activate most neurons and improve the overall performance compared to normal ReLU function.

$$F(x) = \begin{cases} f_1(x) = ax, & x \leq 0 \\ f_2(x) = x, & x > 0 \end{cases} \tag{1}$$

LP_ReLU1 contains a cut-off point (A) and a filtering factor(α). The details of this function are shown in equation 2. LP-ReLU1 has a slope at every point beyond the origin, which is a much-desired property for training any neural network [14]. The experiment results show that LP_ReLU1 better than the ReLU in all corruptions categories, and better than Leaky-ReLU in most of the corruptions categories. However, due to the trainable parameters, LP-ReLU1 takes a large amount of time to train.

$$F(x) = \begin{cases} f_1(x) = 0, & x \leq 0 \\ f_2(x) = x, & x \in (0, A] \\ f_3(x) = A + \alpha(x - A), & x > A \end{cases} \tag{2}$$

LP_ReLU2 contains two cut-off points (A and B, where A < B) and two filtering factors (α and β). The details of this function are shown in equation 3. LP_ReLU2 combines both soft filtering and hard filtering techniques, giving not only enough sparsity in the center for weak features, but also a compact feature space limiting feature shifts [14]. This separate method provides sparsity for correcting low-frequency corruption, as well as compactness for mitigate high-frequency corruptions [14]. As shown in Figure 7 and Appendix a, the experiment results show that LP_ReLU2 is better than other activation function over all corruptions categories. However, models with LP_ReLU2 are slower to train, given that this activation function contains trainable parameters.

$$F(x) = \begin{cases} f_1(x) = 0, & x \leq 0 \\ f_2(x) = x, & x \in (0, A] \\ f_3(x) = A + \alpha(x - A), & x \in (A, B] \\ f_4(x) = \max(f_3(x)) + \beta(x - \max(f_3(x))), & x > B \end{cases} \tag{3}$$

4.4. Complexity (FLOP)

Table 2 shows the number of coefficients and FLOPs calculated from our experimental data on Flower17 with input size (3, 32, 32) and output size (17, 1). Due to the structure of the activation equation, we cannot calculate the exact FLOPs for each time, but we can calculate the upper bound of the FLOPs. According to the data in the table, the number of parameters and FLOPs does not increase much, but as our previous experimental data suggested, the improved resnet50 has a significant improvement in the recognition accuracy of low-quality images.

5. Limitations and Future work

Despite the improvements, there are still several limitations and difficulties in our training. There are several technical problems during training. LP-ReLUs requires plenty of time. How to implement it is another task.

Another problem is that the training process is too time-consuming. Generally, it takes around 2 hours to train 30 epochs. LP-ReLUs are very complex activations functions, especially both have trainable parameters. Due to the long training time and tight deadline, some possible training processes are cancelled.

As shown in the result, we find that the change of activation functions does not improve the performance as much as we expected. In some of the corruption categories, the accuracy between our model and the baseline model is quite close, even slightly lower than the baseline model. This may be because our implementation isn't as mature as the pre-designed baseline models, as mentioned above. Besides, the pre-trained weights show a noticeable influence in the model, which is not expected by us, and we could investigate deeper into that element.

To further develop our discovery, we could implement and test other activation functions. It will be more convincing if we can have more implementation results. To further examine the improvement, more real-world degraded images data could be tested. Due to the significant difference between different neural network models, we could test the LP-ReLUs on more models. Besides, we could train our models more epochs to get a more precise experiment result.

6. Conclusion

In this paper, we tried several models and activation functions on the flower dataset. We started with several baseline models AlexNet, Resnet18, Resnet34, Resnet50, VGG16 and VGG19. We changed the learning rate, training epochs and the batch size. The best baseline model is ResNet50. Since Resnet50 performs the best, we further develop Resnet 50 by changing the activation functions. The activation function of the baseline model is ReLU. We tested Leaky-ReLU, LP-ReLU1 and LP-ReLu2. LP-ReLU1 and LP-ReLU2 are activation functions using trainable parameters and filtering techniques. LP-ReLU2 adopts soft filtering and hard filtering. From our experiments, we find that the LP-ReLu2 performs the best, while LP-ReLu1 is also better than Leaky-ReLU in most tests. Thus, the project can conclude that the improvement in training epochs, learning rate, batch size will impact the model's performance, while the proper choice of an effective activation function also shows a noticeable influence.

References

[1] Qiufu Li, Linlin Shen, Sheng Guo, and Zhihui Lai. WaveCNet: Wavelet Integrated CNNs to Suppress Aliasing Effect for Noise-Robust Image Classification.arXiv:2107.13335v1 , 2021.

[2] Haozhe Liu, Haoqian Wu, Weicheng Xie, Feng Liu, and Linlin Shen. Group-wise Inhibition based Feature Regularization for Robust Classification.arXiv:2103.02152, 2021.

[3] Ding Liu, Bihan Wen, Xianming Liu, Zhangyang Wang, and Thomas S. Huang. When Image Denoising Meets High-Level Vision Tasks: A Deep Learning Approach. arXiv:1706.04284, 2017. 2, 4

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Communications of the ACM* 60.6 (2017): 84-90. 1, 2

[5] Yang Wang, Yang Cao, Zheng -Jun Zha, Jing Zhang and Zhiwei Xiong. Deep Degradation Prior for Low-Quality Image Classification. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11046-11055, 2020. 1, 2, 4

[6] Steven Diamond, Vincent Sitzmann, Frank Julca-Aguilar, Stephen Boyd, Gordon Wetzstein, Felix Heide. Dirty Pixels: Towards End-to-End Image Processing and Perception.arXiv:1701.06487, 2021. 1, 2

[7] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261, 2019. 1, 3, 4

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[9] Wei Zhang, Shengnan Hu, Kan Liu, and Zhengjun Zha. Learning compact appearance representation for video-based person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(8):2442–2452, 2018. 1

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recog- nition, pages 770–778, 2016.

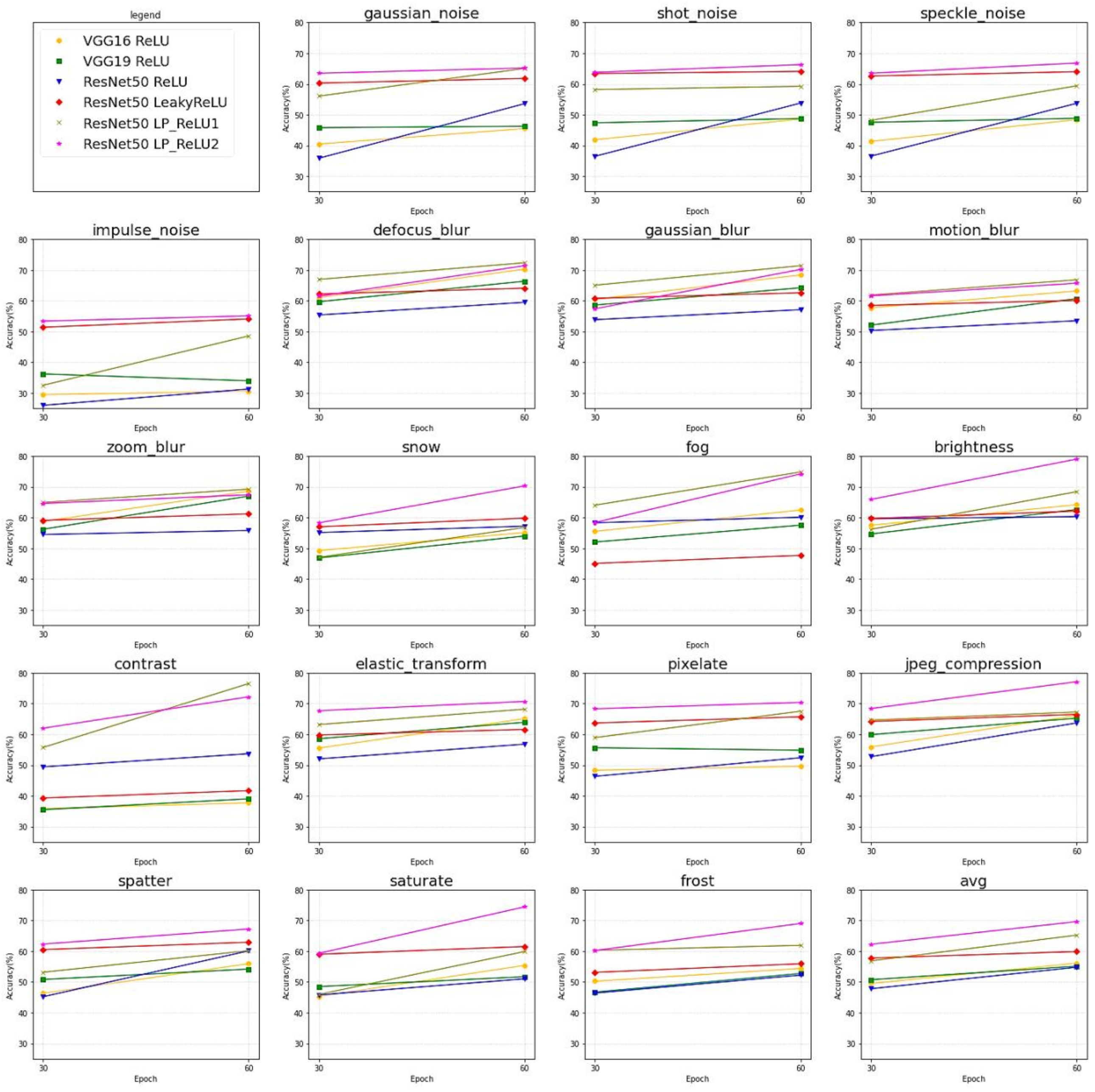
[12] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2009. 2

[13] Maria-Elena Nilsback and Andrew Zisserman. 17 category flower dataset, 2006.

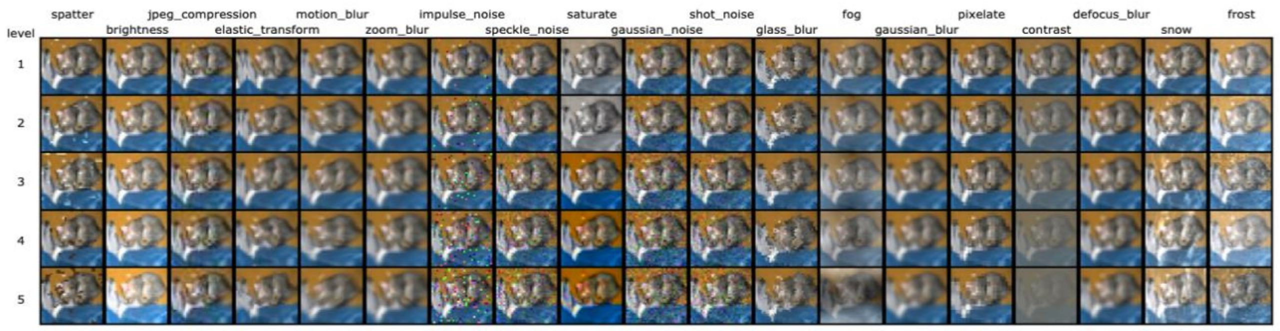
[14] Hossain, Md Tahmid, Shyh Wei Teng, Ferdous Sohel, and Guojun Lu. "Robust Image Classification Using a Low-Pass Activation Function and DCT Augmentation." IEEE Access 9 (2021): 86460–74. https://doi.org/10.1109/access.2021.3089598. 2, 4, 5

Appendix

Our training details and results can be found on GitHub. We also include all the codes and useful tutorials for data preprocessing and building model architectures. Further instructions of how to achieve our results will be provided in the future.



Appendix a. Different models' testing accuracy across all the common corruptions



Appendix b. CIFAR-10-C Dataset Example Images