CS182 CV Project Final Report

Weijia Zeng, Xiaoyi Zhu

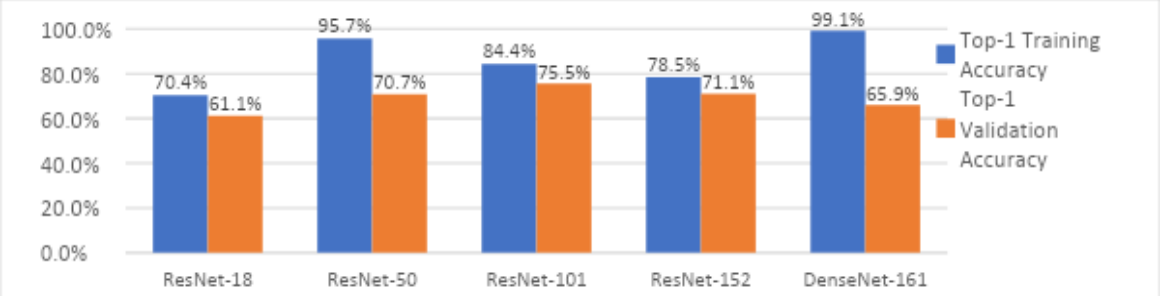
Abstract

In this work, we aim to develop a robust image classification system for dataset with real world perturbations and adversarial examples. We made structural adjustments to the pretrained model for the low-resolution characteristics of the project dataset. We also train our Networks on image-level augmented data and apply data preprocess techniques such as local median filter on adversarial images. Our final model makes predictions based on the ensemble of three specialized Networks to increase the robustness of the system.

1. Baseline Model Selection

1.1 Pretrained Model

Given that our data set is similar to the ImageNet dataset, and pytorch's TORCHVISION.MODELS module integrates many pretrained models, we started by investigating the model families ResNet, and DenseNet. Due to computation power restriction, ResNet-50 was trained for 10 Epochs while ResNet-101, ResNet-152, and DensetNet-161 were only trained for 5 Epochs.

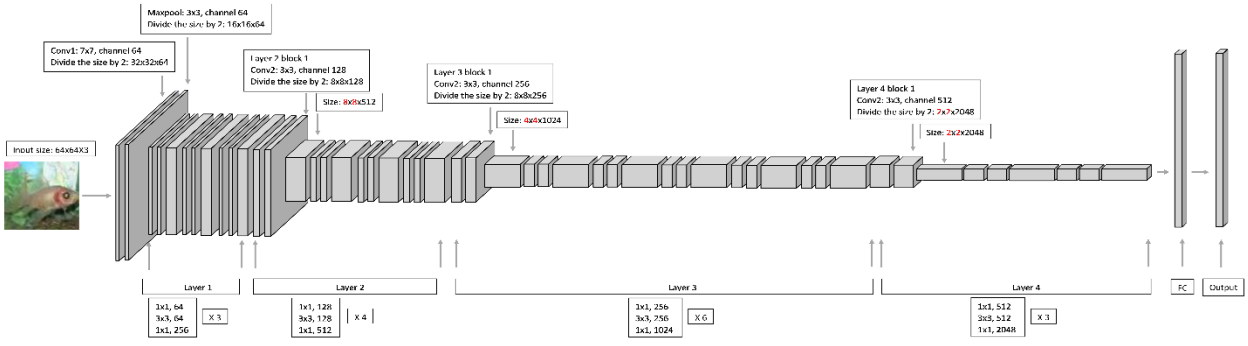


Figure(1): Pretrained Model Performances

Although most pretrained models can achieve decent performance, we have observed a phenomenon that in most of the learning curves, the accuracy of the validation has appeared early saturation. We hypothesized that **the resolution of tiny ImageNet(64x64x3) is much smaller than that of ImageNet(224x224x3), and most of the pretrained models reduce the output size of the layer prematurely, which causes the network unable to fully extract and learn the effective features of the image, thus resulting in overfitting**.

1.2 ResNet-50 Modification

Due to limited time and computation resources, we decided to modify the ResNet-50's architecture. As illustrated as the following picture, if we input a picture with a size of 64x64x3 into the original ResNet-50 network. The size would be decreased by a factor of 2 immediately to 32x32x64 on the first convolution layer, and the size would keep decreasing at each layer. When it reaches layer 3, the size only remains at 4x4, and 2x2 on layer 4. Hence it is hard for the network to learn spatial information with such a narrow size even though the layer is deep enough.



Figure(2) Original ResNet50 architecture

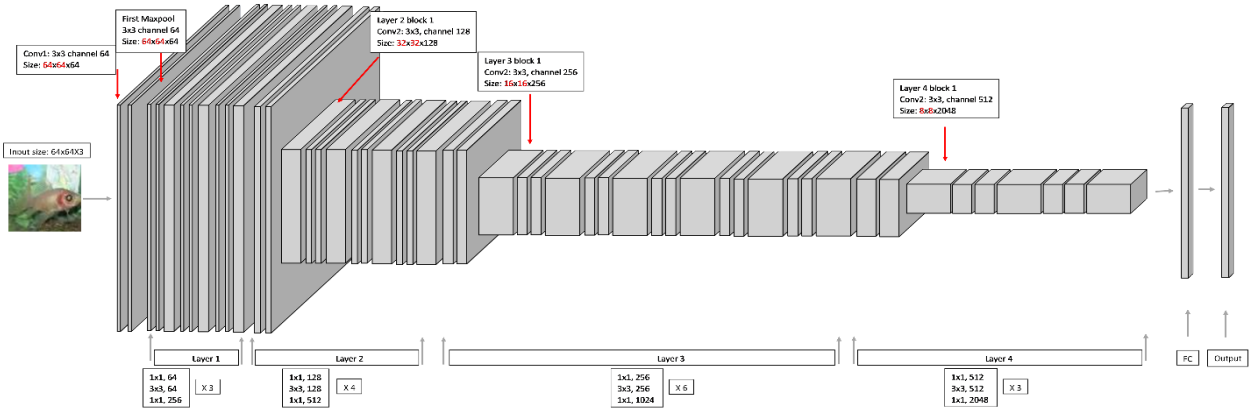
To solve the problem, we tried three ways to adjust the ResNet50's architecture. Since we only use Method 3 in our final ensemble, we focus on discussing method 3 here and move the

descriptions of model 1 & 2 to Appendix (a)(b).

1.2.1 Method 3

We made two modifications to the network's architecture:

1. Change the first conv1 layer from kernel size 7, stride 2 and padding 3 to kernel size 3, stride 1 and padding 1.
2. Change the first Maxpool layer from kernel size 3, stride 2 and padding 1 to kernel size 3, stride 1 and padding 1.



Figure(3): Modified ResNet-50 architecture Method 3

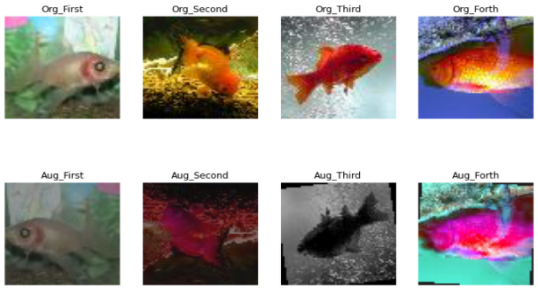
Under this architecture, the size of each layer is relatively large, which ensures that relatively rich information is given to the network to learn, and the Maxpool layer of each layer is retained to learn abstract information. This architecture obtained the highest validation accuracy performance, and we decided to choose this structure as our baseline model

Model	Top-1 Training Acc	Top-1 Validation Acc
Original ResNet-50	95.0%	70.7%
Method 3	91.1%	74.5%

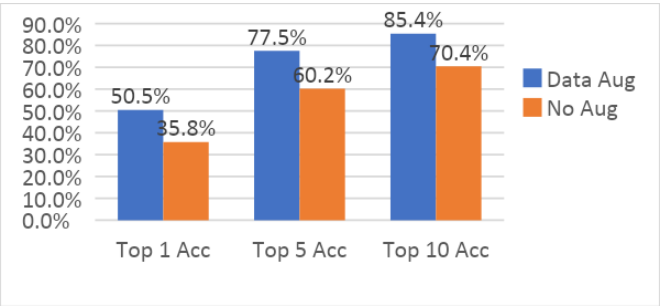
Table(1): Performances of ResNet-50 and modified ResNet-50 method 3

2. Image Level Perturbation

Image level perturbations are real world perturbations of images. They are not model specific adversarial attacks. We augmented our dataset by a few common perturbations to account for image level perturbation. The perturbations we used are RandomRotation, Vertical/Horizontal Flip, Grayscale, and ColorJitter.



Figure(1): row 1 are four original images ; row 2 are the four perturbed images



Figure(2) Performances of our baseline model with / without data aug trained 5 epochs

Figure(2) compares performance of the baseline model training with data augmentation and without data augmentation evaluating on the augmentation validation set. The data augmented model performs better than the non augmented model by 15% in top-1 accuracy.

3. Pixel Level Perturbation

Pixel level perturbation is the generation of adversarial examples, where some adversary crafts the input image so that the target classifier will produce an incorrect output. We used Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM) to evaluate model robustness.

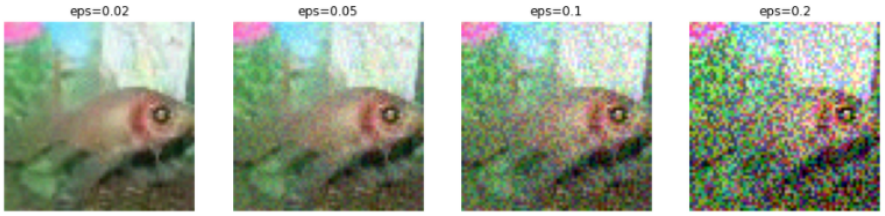
3.1 Fast Gradient Sign Method (FGSM)

FGSM explores the gradients of a neural network and builds adversarial images based on the gradient. FGSM first computes the gradient of the neural network's loss function with respect to the input, then uses gradient's sign to construct an adversarial image that maximizes the loss. The equation for FGSM is:

$$X_{adv} = X + \epsilon sign(\nabla_x J(\theta, x, y))$$

where ϵ represents how much adversarial noise to add, J is the loss function, and $\theta$is our neural network.

We tried four values of ϵ and decides to use ϵ = 0.02 in training and validation process as it is the largest value that will fool human eyes and yet corrupt our neural network.



Figure(3): FGSM attack on our baseline model with different ϵ

3.2 Basic Iterative Method (BIM)

Basic Iterative Method is an extension of FGSM, where we apply FGSM multiple times with a small step size. After each step, we clip pixel values of intermediate results to ensure that they are in an α-neighborhood of the original image.

$$X_{adv} = X, X_{adv} = Clip_{X,\alpha}\{X_{adv} + \epsilon sign(\nabla_X J(X_{adv}, y)\}$$

Similar to FGSM, we decides to use ϵ = 0.02 for BIM attack.

4. Image Preprocess

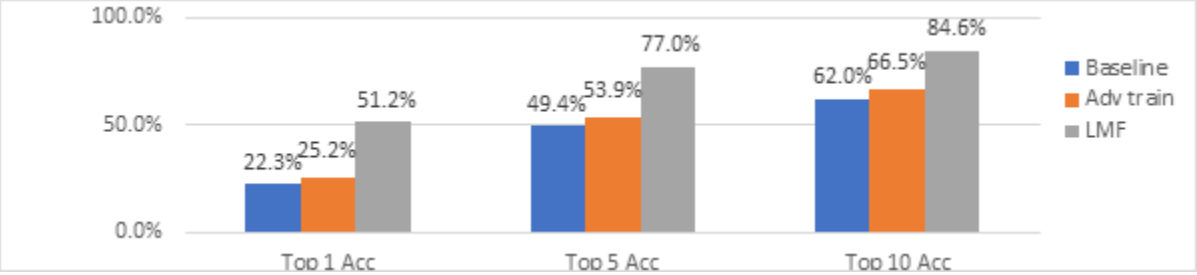
To deal with adversarial images,, we explore the idea of image preprocess, which transforms the input image before feeding it into the neural network. Our group hypothesized that **spatial structure will be important for low resolution images**. Lambros, et. al(1) mentioned that spatial resolution describes how detailed an object can be represented by the image. Therefore we want to **reduce the effect of adversarial attack while preserving image spatial structure as much as possible**. We explore one spatial smooth(3) method local median filter and compare it with standard adversarial training on FGSM attacks. We then propose a weighted version of the local median filter (WLMF). Since we didn't include WLMF in our final ensemble model, we put the section of WLMF into Appendix(b).

4.1 Local Median Filter (LMF)

The Local Median Filter is defined as

$$X_i = median\{\forall j \in \Omega(i): X_j\}$$

Where Ω(i) represents a local region of the image and the local median operation is performed for each channel separately. Xie, et.al(2) mentioned that Median filters are good at removing salt-and-pepper noise and other similar noises while preserving image spatial structure.
We applied LMF to our baseline model and compared it with adversarial training on a FGSM applied validation set.



Figure(4): Performances of adversarial training model and LMF model on FGSM validation set trainrf 5 Epochs

The result in Figure(4) shows that the LMF model outperforms the adversarial training model. This result matches our expectation. We learned in class that adversarial training can improve model robustness against a specific adversarial attack but may suffer a low prediction accuracy. On the other hand, LMF is not attack specified, meaning that it can be used to defend against any attack, while preserving model accuracy.

5. Ensemble and Validation Result

Our final model is an ensemble of three different models: modified ResNet50 (data augmented), LMF pretrained ResNet101, pretrained ResNet101(data augmented) and each with weight 0.5, 0.3, 0.2. We test our model on a validation set with data augmentation, 30% FGSM attack and 30% BIM attack and has a top-1 accuracy of 51.3%, top-5 accuracy of 78.5%, and top-10 accuracy of 85.8%.

6. Lesson Learned

- For convolutional networks, the shallow layer size is critical to the learning ability of the network, our method 2 and method 3 modified ResNet-50 only differentiated in the size of the first layer block, but the performances are quite different.
- Pooling Layer is critical to avoid overfitting. Our method 1 and method 3 modified ResNet-50 are very similar in terms of size at each layer except method 1 without the first Max Pooling layer, which results in serious overfitting.
- For image level perturbations, we can apply simple technique-data augmentation- to our training set and the improvement in accuracy is significant.
- Dealing with adversarial examples is more trickier; where adversarial training makes a trade-off between model robustness and accuracy and the improvement is not very significant. Advanced techniques (e.g LMF) works better and more general than adversarial training.

7. Contributions

Xiaoyi Zhu wrote about 50 percent of the Final Project Report, including the Abstract, Baseline Model Selections, and wrote the Pytorch code for baseline model modification and test, model training, validation test, model Ensemble and the test submission. Xiaoyi Zhu did about 50 percent of the Final Project.

Weijia Zeng wrote about 50 percent of the Final Project Report, including Image Level Perturbation, Pixel Level Perturbation, Local Median Filter, and Ensemble and Validation Result. Weijia Zeng explored different kinds of adversarial attack and some defensive methods using the Adversarial Robustness Toolbox. Weijia also did adversarial training and analysis between different defensive techniques. Weijia Zeng did about 50 percent of the Final Project.

Reference:

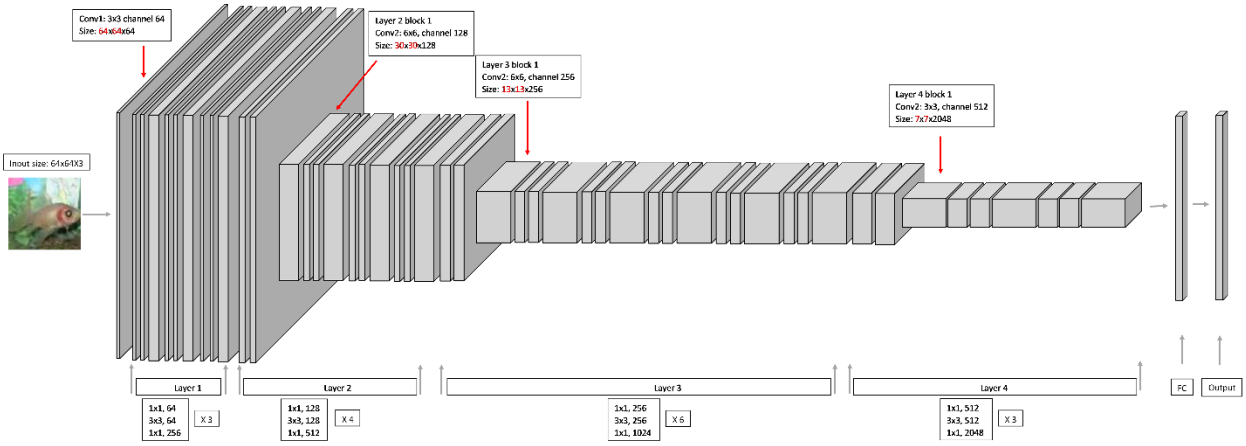
1. Lambros S. Athanasiou, Dimitrios I.Fotiadis, Lampros K. Michalis. Propagation of Segmentation and Imaging System Errors. Atherosclerotic Plaque Characterization Methods Based on Coronary Imaging: page 151 - 166, 2017
2. Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, Kaiming He. Feature Denoising for Improving Adversarial Robustness. arXiv:1812.03411v2 [cs.CV], 25 Mar 2019
3. Weilin Xu, David Evans, Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. arXiv:1704.01155v2 [cs.CV] 5 Dec 2017

Appendix

A ResNet-50 Modification Method 1

Modifications to the network's architecture:

1. Change the first conv1 layer from kernel size 7, stride 2 and padding 3 to kernel size 3, stride 1 and padding 1. (Output size:(N, 64, 64, 64), N is the batch size)
2. Cancel the first Maxpool layer.
3. Change the downsample layer at layer 2, block 1 to a Conv2d layer with kernel size 6, stride 2 and padding 0. (Output size:(N, 512, 30,30))
4. Change the conv2 layer at layer 3, block 1 from kernel size 3, stride 2 and padding 3 to kernel size 6, stride 2 and padding 0.
5. Downsample layer at layer 3, block 1 from kernel size 1, stride 2 and padding 0 to kernel size 6, stride 2 and padding 0. (Output size:(N, 1024, 13, 13))



Figure(1): Modified ResNet-50 architecture Method 1

Under this architecture, the Network maintains the full input size(64x64) throughout the whole layer 1. But canceling the Maxpool layer before layer 1 reduces the Network's ability to learn the abstract features, which finally cause overfitting.

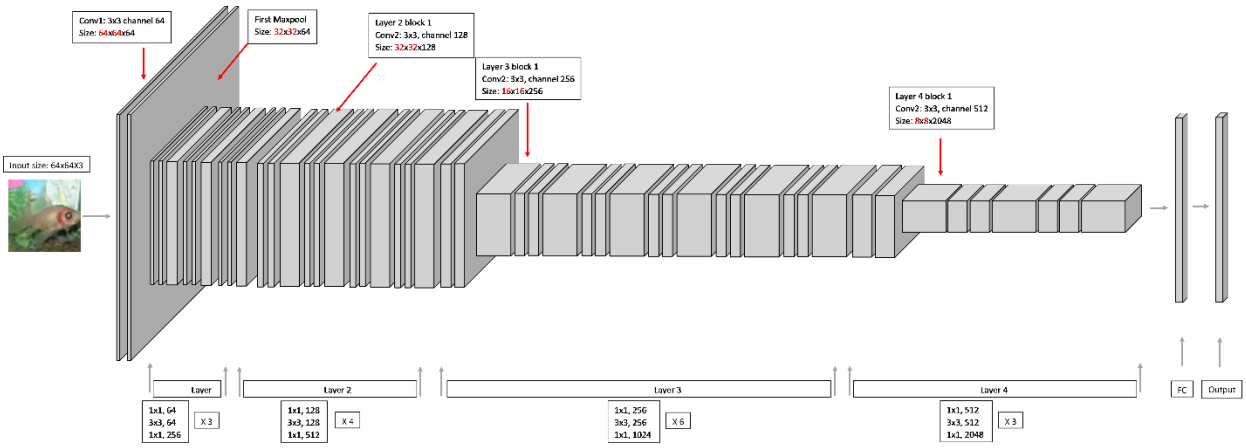
Model	Top-1 Training Acc	Top-1 Validation Acc
Original ResNet-50	95.0%	70.7%
Method 1	96.7%	42.2%

Table(1): Performances of ResNet-50 and modified ResNet-50 method 1

B ResNet-50 Modification Method 2

Modifications to the network's architecture:

1. Change the first conv1 layer from kernel size 7, stride 2 and padding 3 to kernel size 3, stride 1 and padding 1. (Output size:(N, 64, 64, 64), N is the batch size)
2. Keep the first Maxpool layer. (Output size:(N, 64, 32, 32))
3. Change the conv2 layer at layer 2, block 1 from kernel size 3, stride 2 and padding 1 to kernel size 3, stride 1 and padding 1.
4. Change downsample layer at layer 2, block 1 from kernel size 1, stride 2 and padding 0 to kernel size 3, stride 1 and padding 1. (Output size:(N, 512, 32, 32))



Figure(2): Modified ResNet-50 architecture Method 2

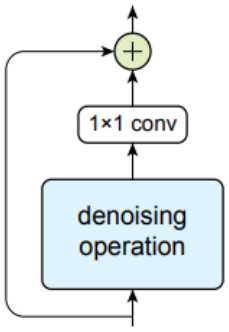
Under this architecture, the input size was reduced before layer 1 as the original ResNet-50, but remains the input size after layer 1 to layer 2, which makes the latter layers have a relatively larger size. But it turns out the input size before layer 1 is important for the network to learn the features as the performance of this architecture is way worse than method 3.

Model	Top-1 Training Acc	Top-1 Validation Acc
Original ResNet-50	95.0%	70.7%
Method 1	76.0%	59.0%

Table(2): Performances of ResNet-50 and modified ResNet-50 method 2

C Weighted Local Median Filter

Inspired by the denoising network structure by XIe, et. al (2),

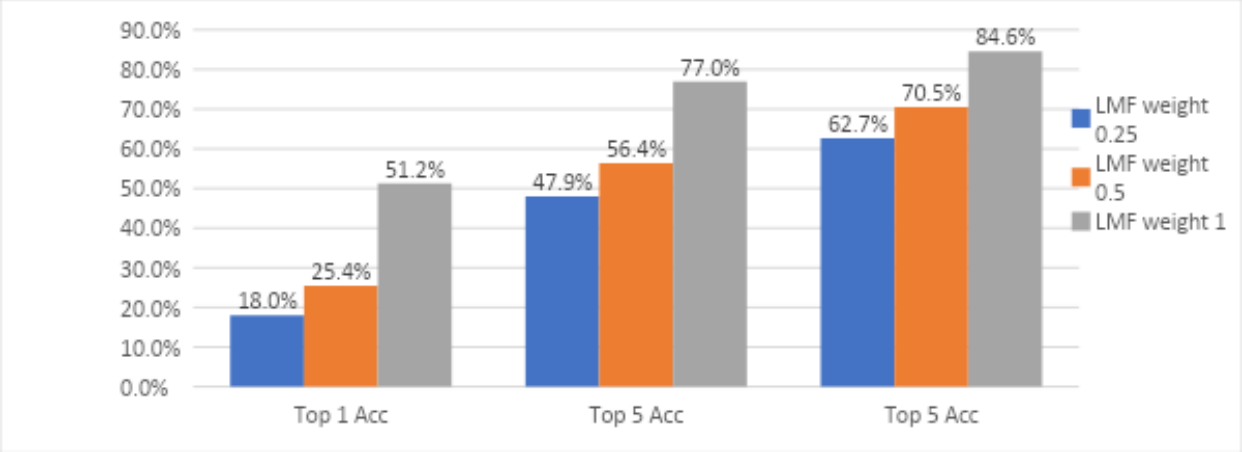


Figure(3): the generic denoise layer proposed by Xie, et.al in (4)

we propose a weighted version of local median Filter:

$$X_{weighted} = (1 - \alpha)X_{original} + \alpha X_{LMF}, \alpha \in [0, 1]$$

Since the original input image has good spatial structure and the smoothed image has less adversarial effect, combining the two images may help to preserve spatial structure while reducing adversarial effect.



Figure(3): different weights of LMF model trained 5 Eopchs

To our surprise The result above shows that the solo median filter method has the best performance. We think this is because the noise added by the original image outperforms the improvement of spatial structure.