Improving Convolutional Neural Network Robustness to Adversarial Images Through Image Filtering

Natalie Elizabeth Bogda

Dr. Amir Sadovnik

Min H. Kao Department of Electrical Engineering and Computer Science

June 10, 2020

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline

- Introduction
- Background
- Support Vector Machine Defense
- Datasets and Networks
- Defending Against Adversarial Examples
- Conclusions

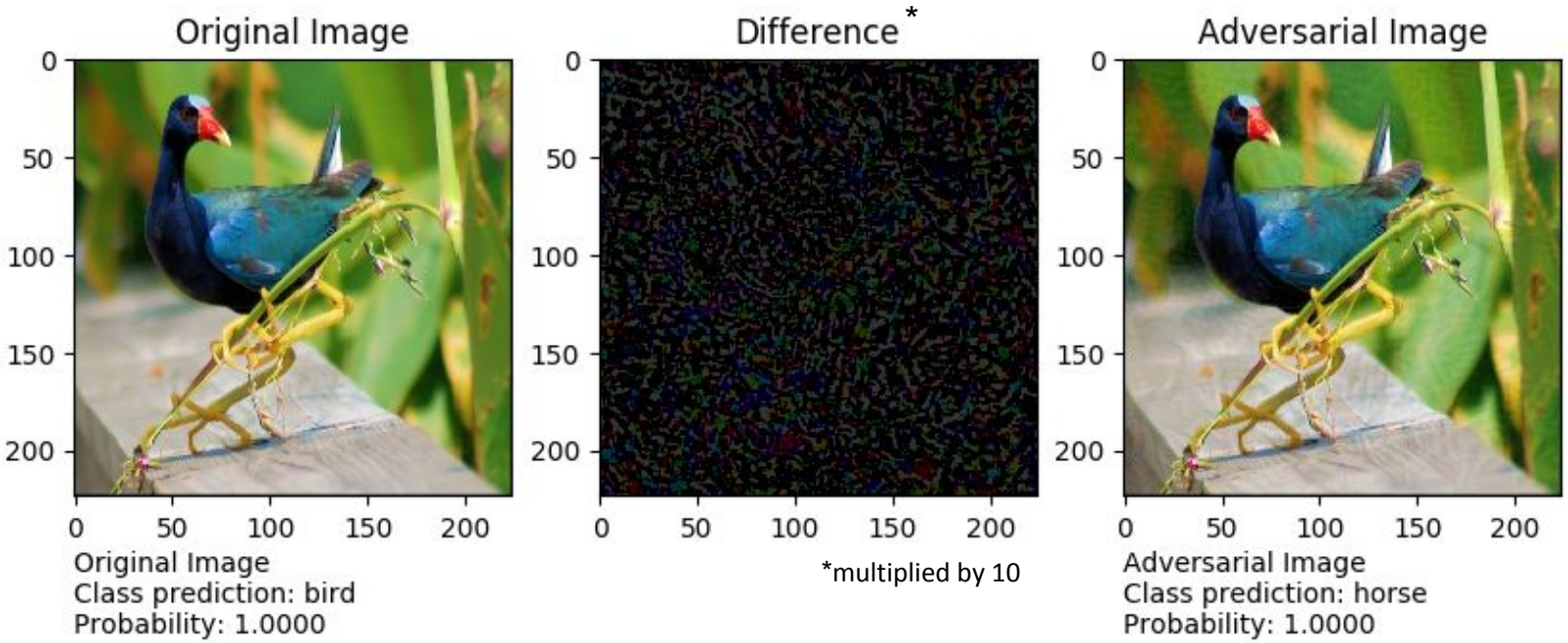
Introduction

Introduction

- Convolutional neural networks (CNNs) have the capability to classify images with extreme accuracy.
  - Current state-of-the-art (a CNN named *AmoebaNet-A*) correctly classifies a 1.2 million image dataset with 1,000 classes with 96% accuracy (Real, 2019).
- However they are are vulnerable to ***adversarial examples***.
  - These cause an otherwise accurate CNN's accuracy score to plummet.
- Indicates a "blind spot" in how CNNs "see" images (Szegedy, 2013).

What is an adversarial example?

- An image with pixels that have been carefully manipulated such that a classifier misclassifies it with a high confidence score.
- The image looks unaffected to the human viewer.



Motivation

- Adversarial images pose a threat to systems that rely on CNNs to process and analyze data.
- Examples of systems:
  - Content filter on website.
  - System that scans handwritten digits.
  - Camera on self-driving car.

Stop sign misclassified as Speed Limit 45 sign after careful placement of stickers (Eykholt, 2018)

Image altered with adversarial noise and printed out is misclassified as a safe by the Tensorflow Camera Demo app (Kurakin, 2016)

Hypothesis

- Adversarial images are a result of ***pixel manipulation.***
- Therefore, ***additional pixel manipulation (filtering)*** that masks adversarial noise should ***reduce adversariality.***
  - However, filtering the images will cause reduction in classifier accuracy on clean images.

Goal of Thesis

To improve adversarial robustness while retaining original accuracy.

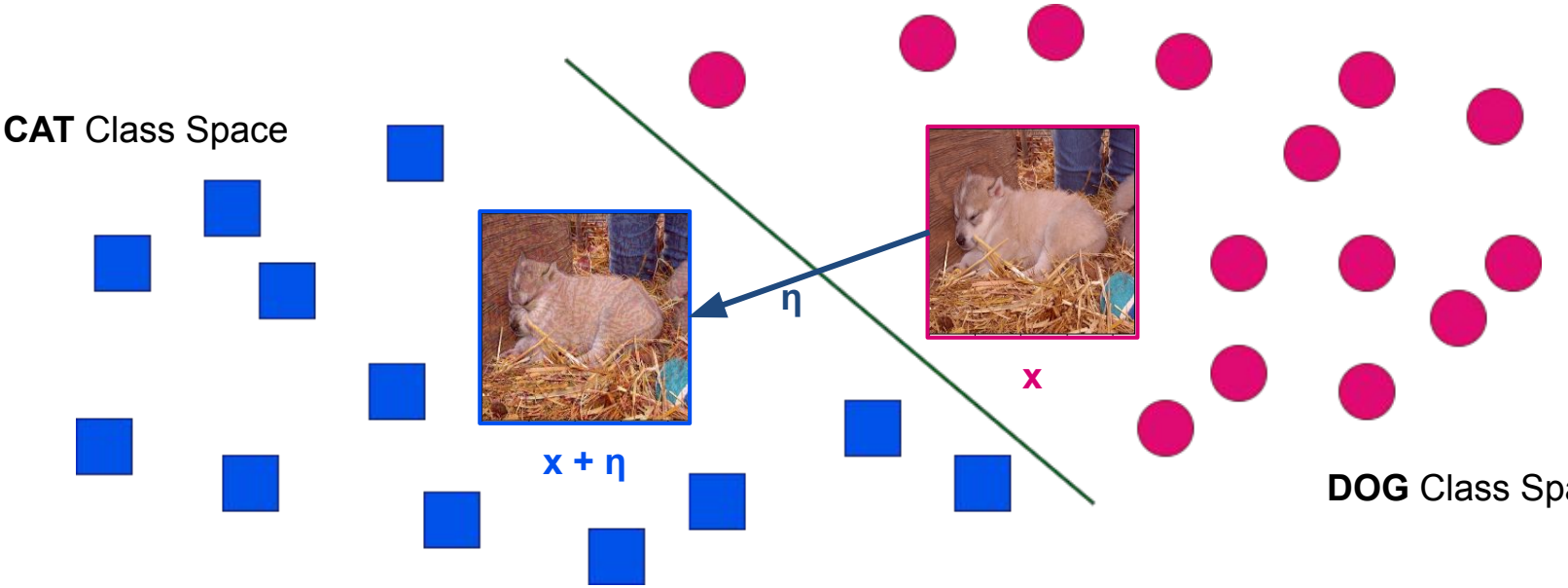
Background

What causes adversarial examples?

- ***The linear components in a CNN*** (Goodfellow, 2014).
- Just a few small “wiggles” in high dimensional input multiplies to create a large change in the output.
- These wiggles are referred to as the perturbation **η**
  - **x’ = x + η**

Intuition

- **Small changes in the input add up to one large change in the output.**
  - Given an image, how can I “wiggle” (add perturbation **η** to) every pixel in such a way to increase the score of an incorrect class?



To how compute η

- Many different ways to compute.
- This thesis uses **two** such methods:
  - **Fast Gradient Sign Method (FGSM)**
  - **Projected Gradient Descent (PGD)**
- In the following slides, the perturbation **η** will be indicated with a red box.

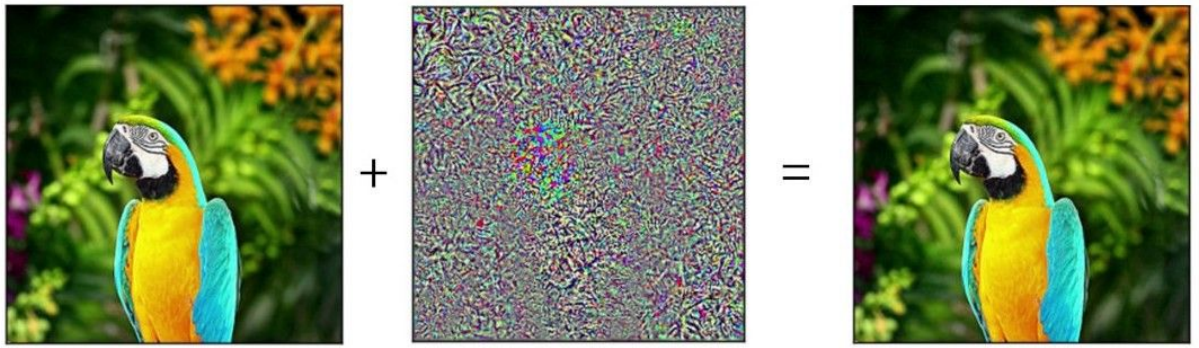
Important Terminology

- **Epsilon (ε)** refers to adversarial strength.
  - Larger epsilon = stronger adversary

Fast Gradient Sign Method

$$x' = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y))$$

- y is the targets associated with **x**.
- J(**Θ**, **x**, y) is the cost function.
- ∇_x is the gradient.
- ϵ is the amount to add to the sign matrix.



X

97.3% macaw

sign(∇_X J(θ, X, Y))

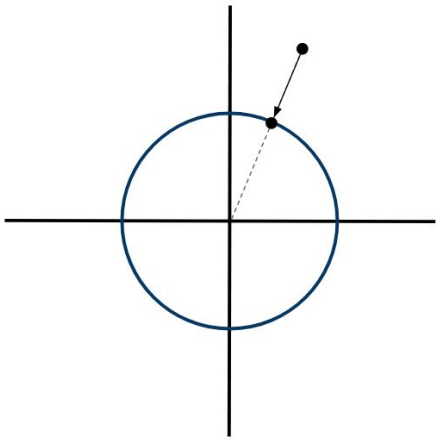
X + ϵ.sign(∇_X J(θ, X, Y))

88.9% bookcase

Projected Gradient Descent

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha \cdot sign(\nabla_x J(\theta, x, y))$$

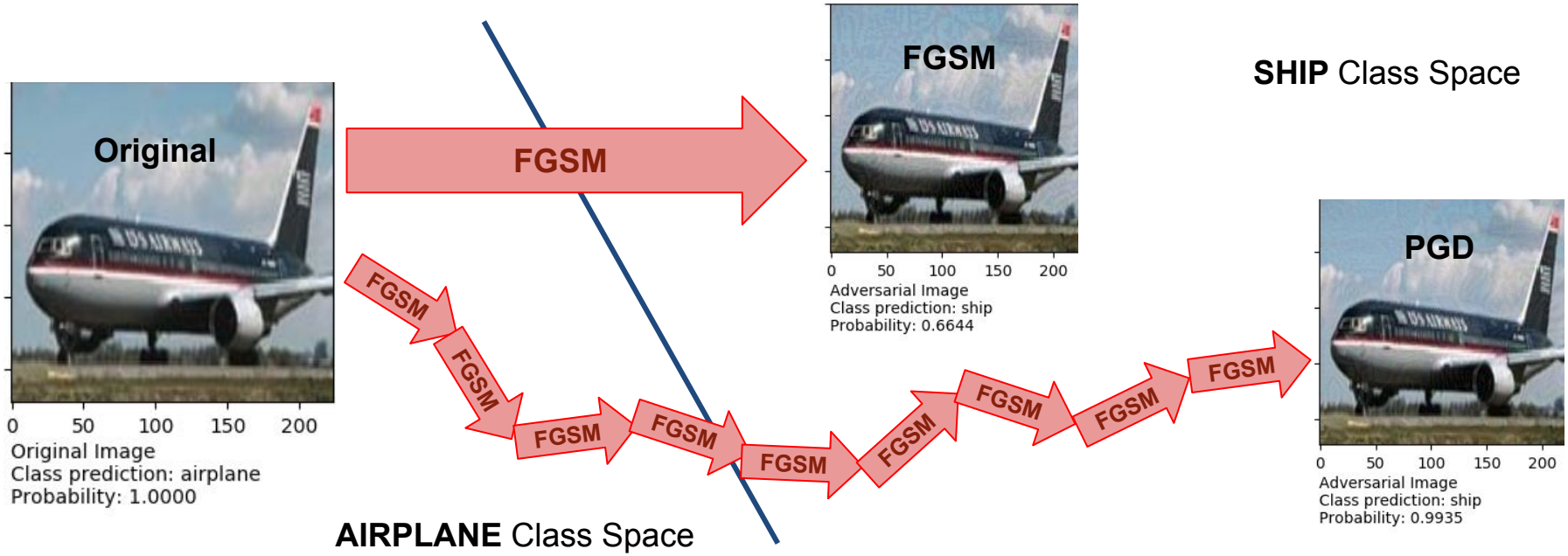
- t is number of gradient steps.
- $\mathbf{x}^t$ is adversarial example after t steps.
- α is ϵ /t where ϵ is same as in FGSM.
  - ϵ is the amount to add to the sign matrix.
- S is allowed perturbation distance from **x**
- $\mathbf{\Pi}_{\mathbf{x}+S}$ is the clamping factor (the projection back into the L-ball)

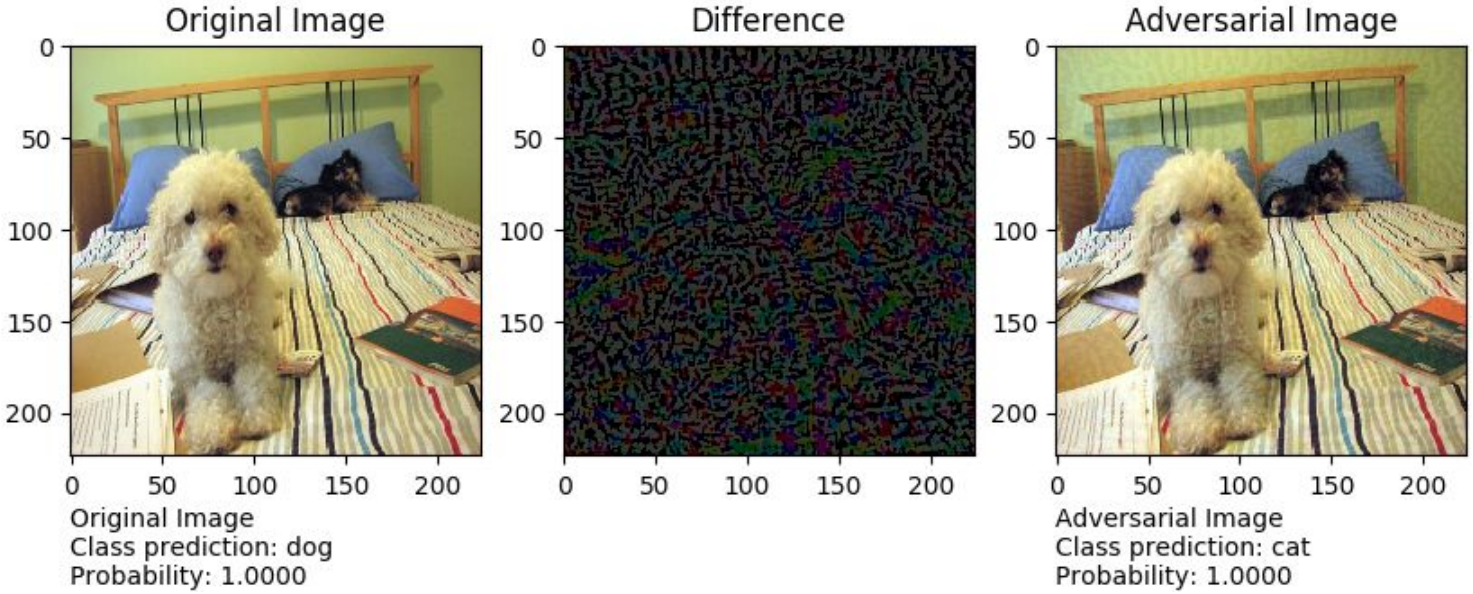


Projection of perturbation back onto L ball.

FGSM vs PGD

- Both methods aim to optimize the input pixels to create maximum loss using the sign of the gradients from the network.
- FGSM does so in a “one-shot” way, while PGD computes this iteratively.
  - Recomputes the gradient at each step.
  - PGD is simply an iterative, constrained version of FGSM.

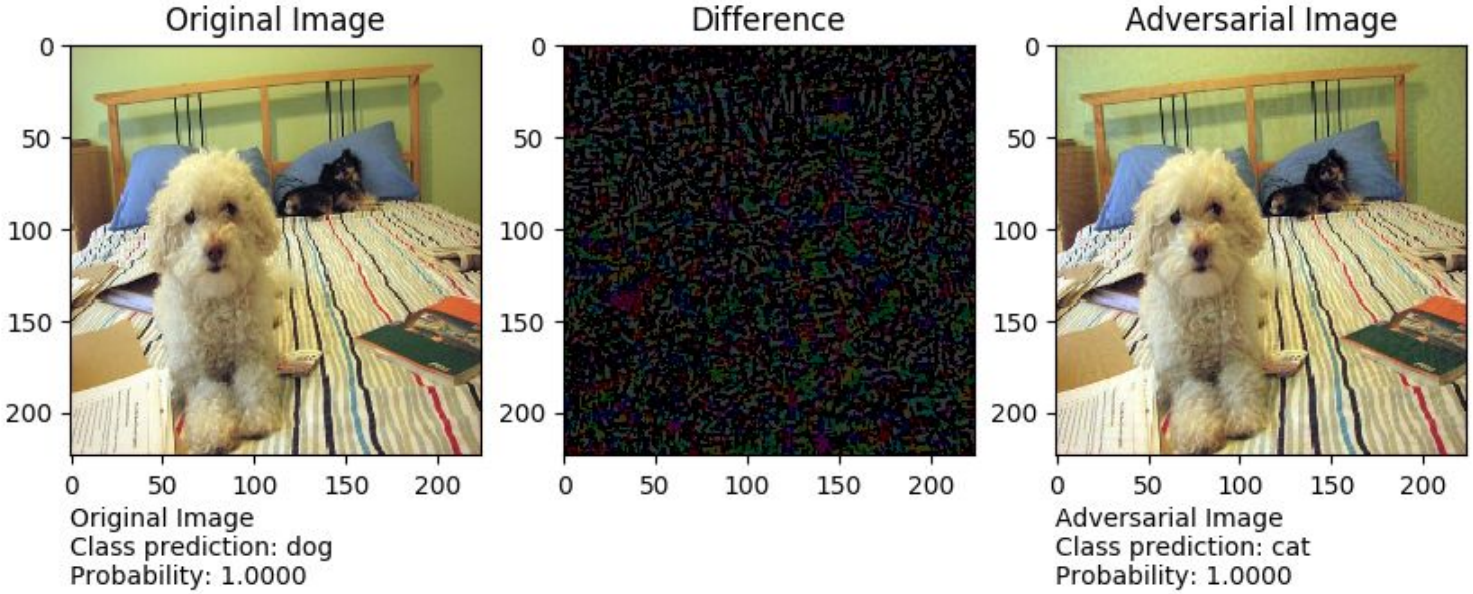




FGSM, ϵ = 5

Noise Comparison

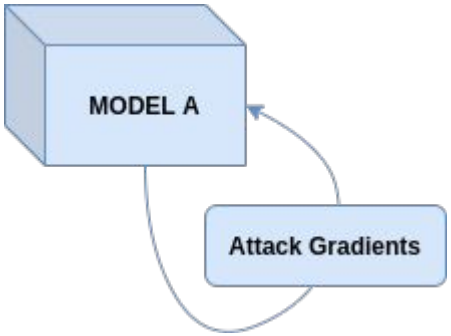
Differences multiplied by 10



PGD, ϵ = 5

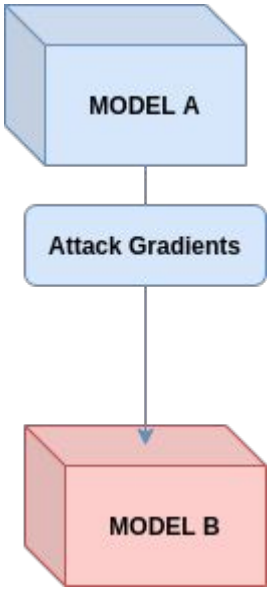
Attack Types

White Box



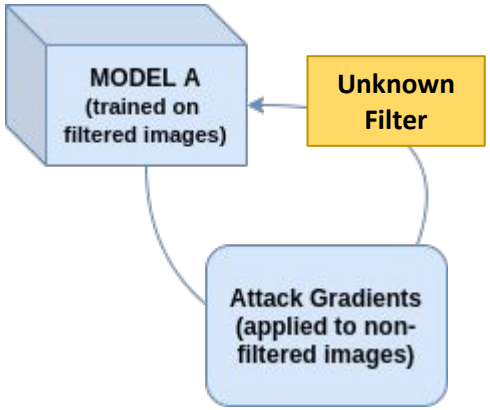
Attack computed using model's own gradients.

Black Box



Attack computed using different model's gradients.

Gray Box



Attack computed using filter-image-trained model's own gradients, but attack applied to clean images (assume filter unknown).

Filtering Defenses Used

Small Images

Threshold

Gaussian

Median

Bilateral

Resize

Big Images

ROF

Chambolle

Non-local Means

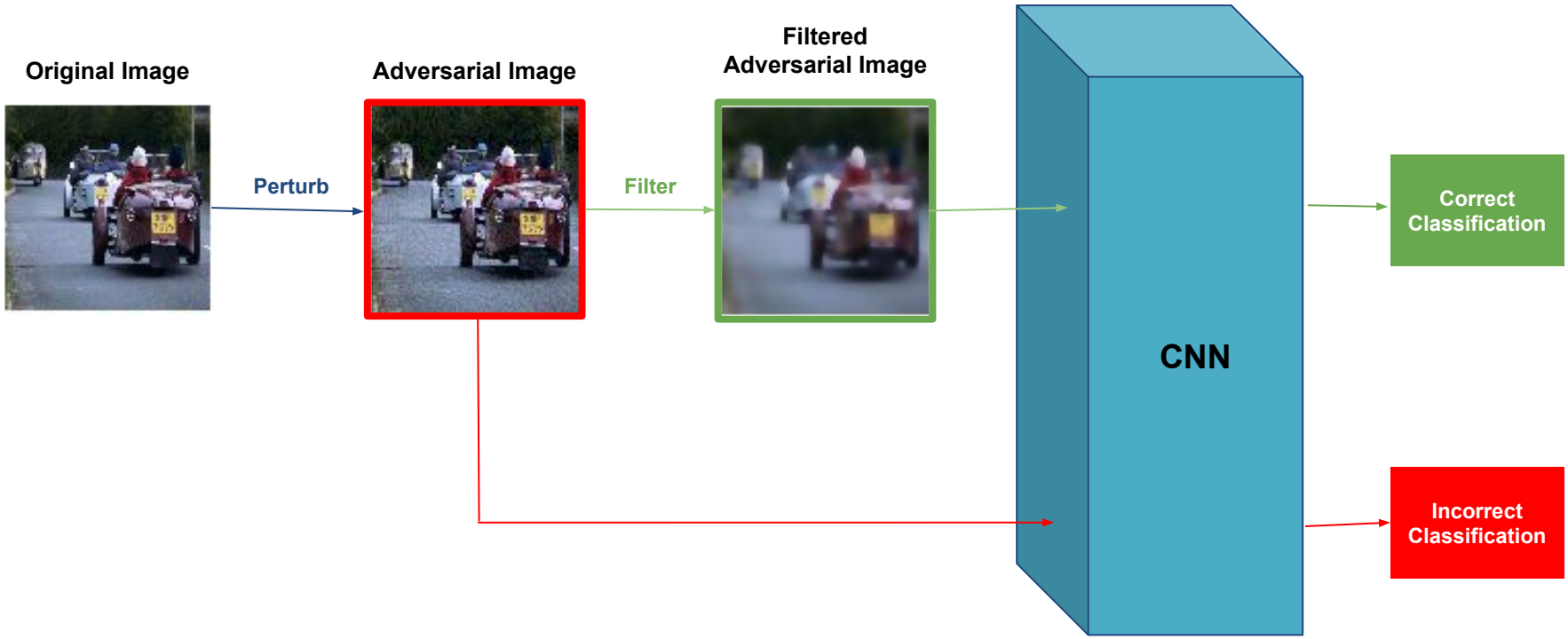
Anisotropic Diffusion

Median + Gaussian

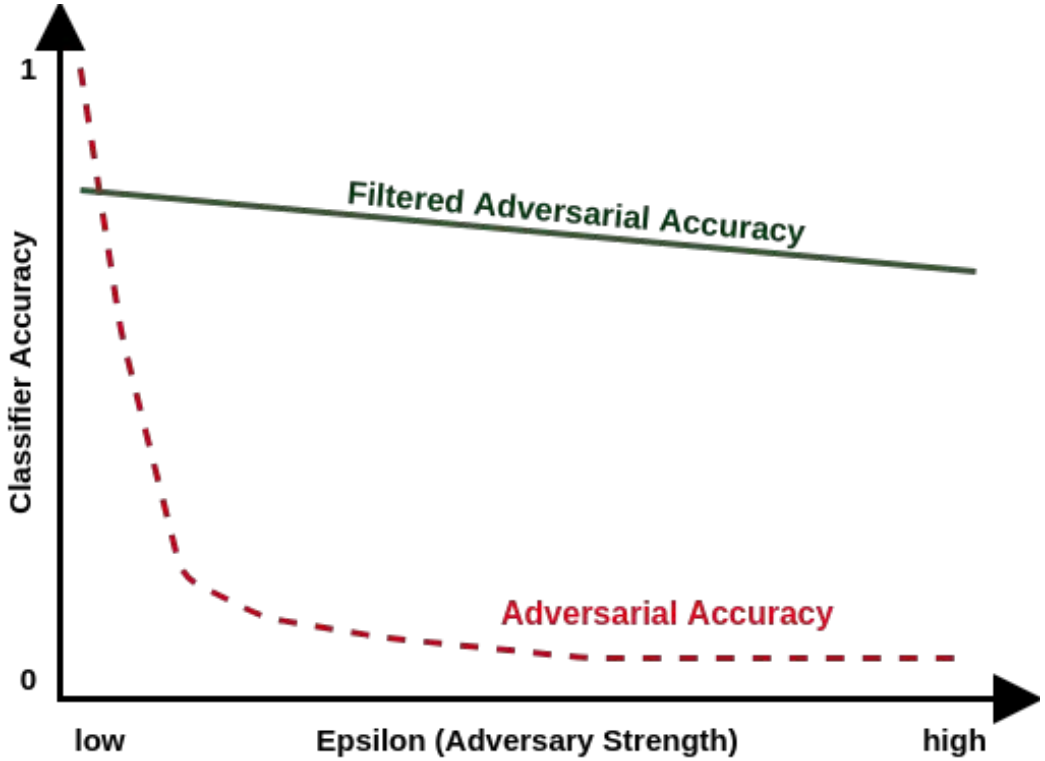
Why does filtering help?

- This can be explained by removal of non-robust features, which are vulnerable to adversarial noise (Ilyas, 2019).
  - Models rely on these to generalize.
  - Adversarial examples take advantage of these vulnerable features to alter an image.

Pipeline



Desired Outcome

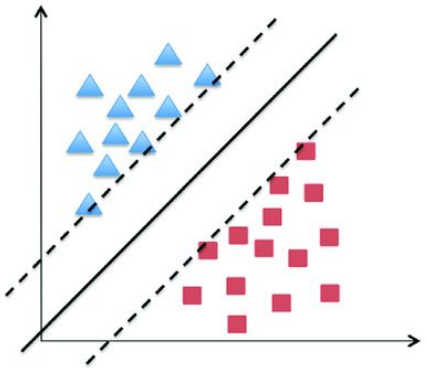


As adversarial strength grows, adversarial images processed with the filtering techniques should continue being correctly classified.

Support Vector Machine Defense

Proof of Concept

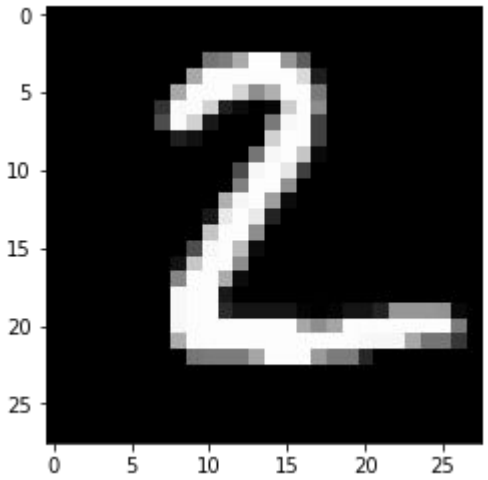
- Create and defend against adversarial images on a Support Vector Machines (SVM).
- Quick review of SVM
  - Classifies by finding the best decision boundary between classes.
  - Seeks to maximize the margin between classes.

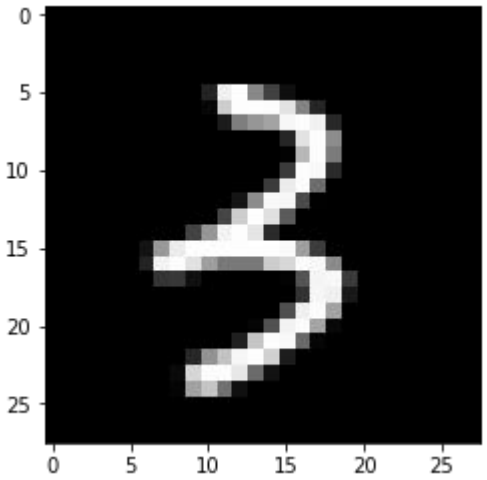


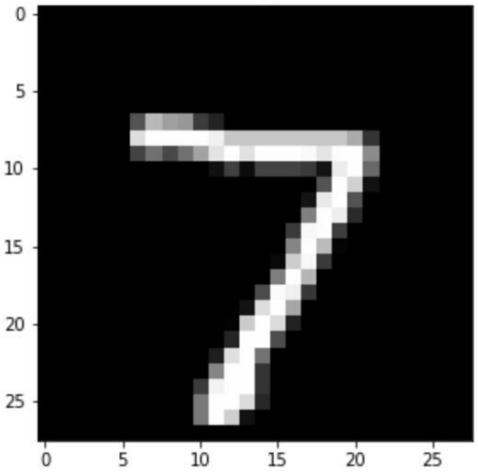
Example of SVM

Data Set

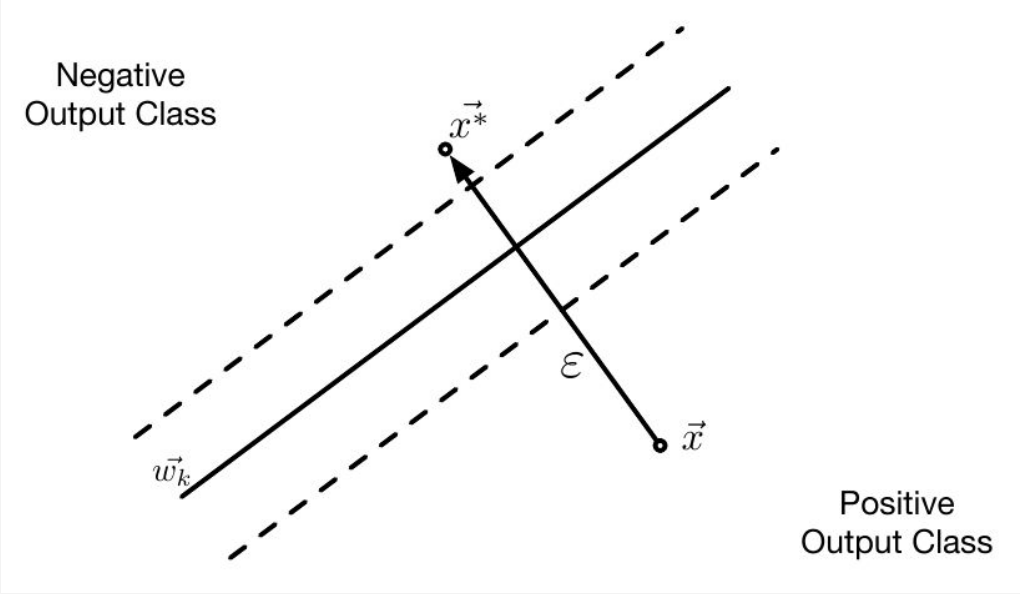
- Use the 2, 3, and 7 class images from MNIST
- 6,500 train samples per class
- 500 test samples per class







Crafting Adversarial Examples



Visualization of how to push an image **x** into another class **x*** when pushed by an amount ϵ over the linear decision boundary $\mathbf{w_k}$

$$\vec{x^*} = \vec{x} - \varepsilon \cdot \frac{\vec{w[k]}}{||\vec{w_k}||}$$

x* = adversarial image

x = input image from class k

ε = input variation (how far to push)

w[k] = weight vector of class k

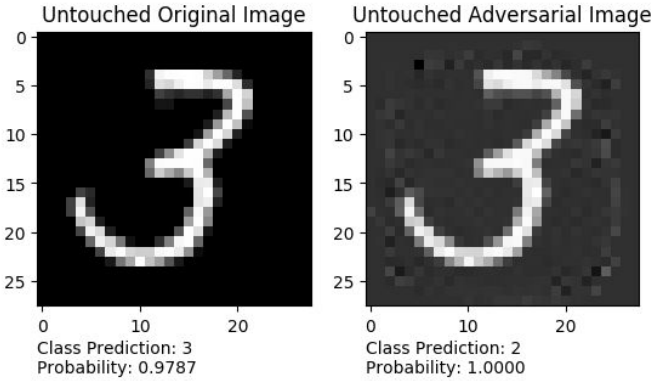
$||\mathbf{w_k}||$ = Frobenius norm

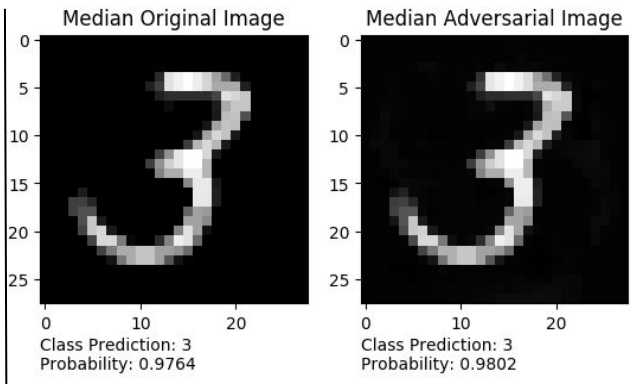
Defenses Used

- Thresholding
- Gaussian filtering
- Median filtering
- Bilateral filtering
- Resizing

Filtering Results

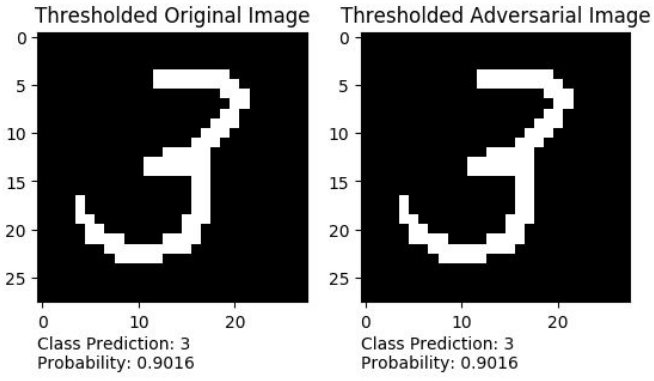
Epsilon: 200

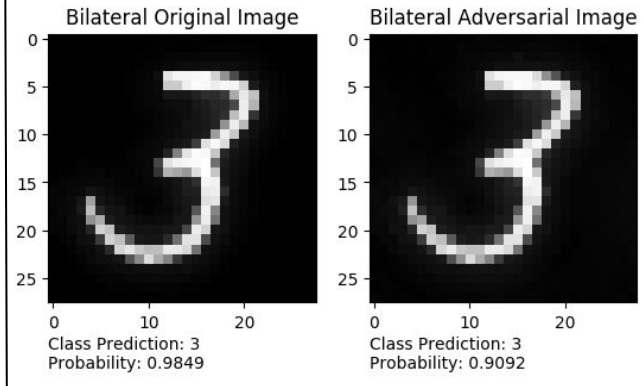




Kernel size: 3 x 3

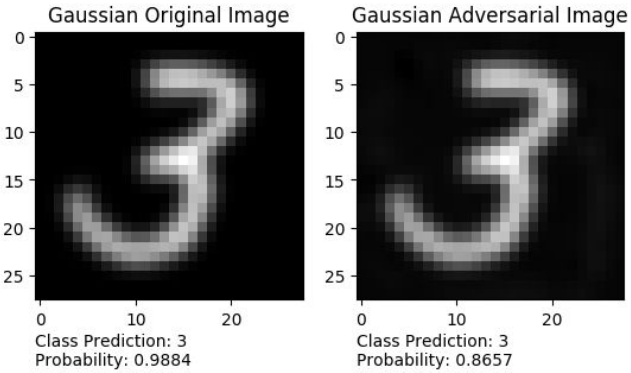
Threshold: 100





Kernel size: 9 x 9,
Range & Domain
Kernel: 100

Sigma: 5,
Kernel size: 5 x 5



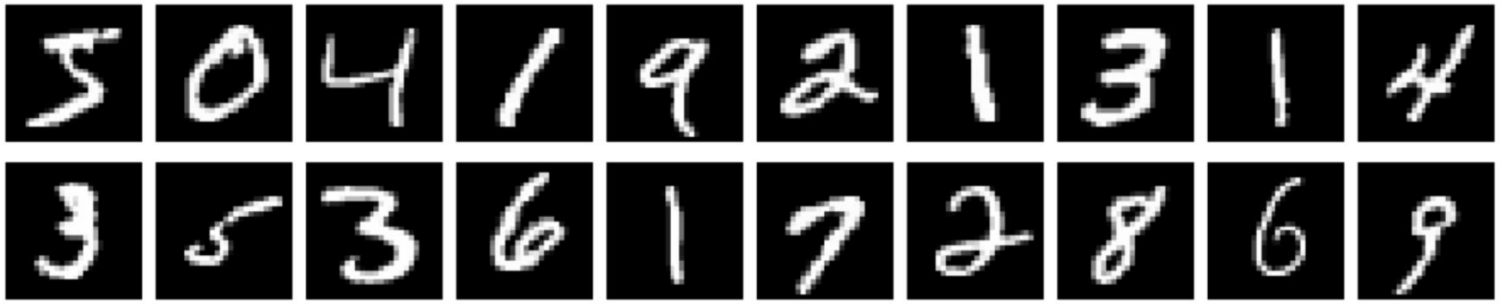
Downsize: 16 x 16

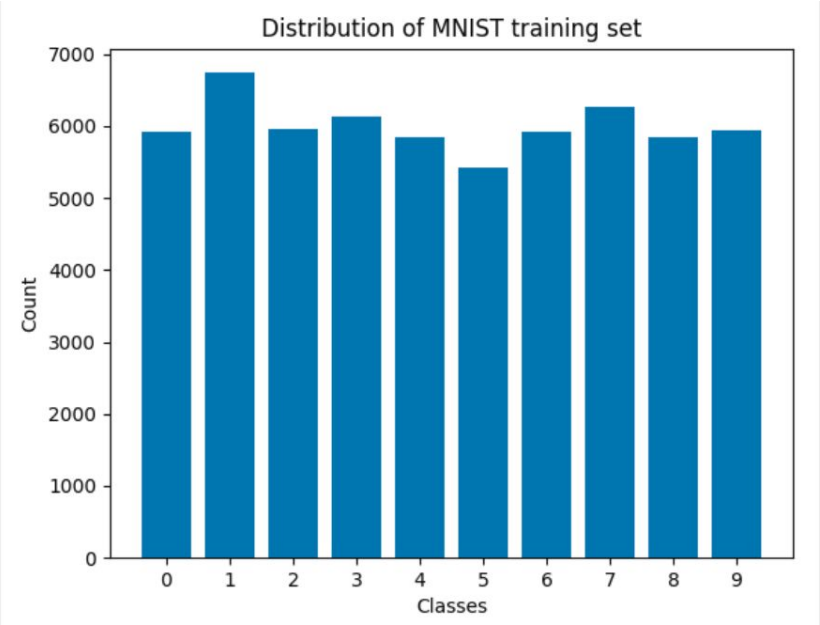
Conclusion

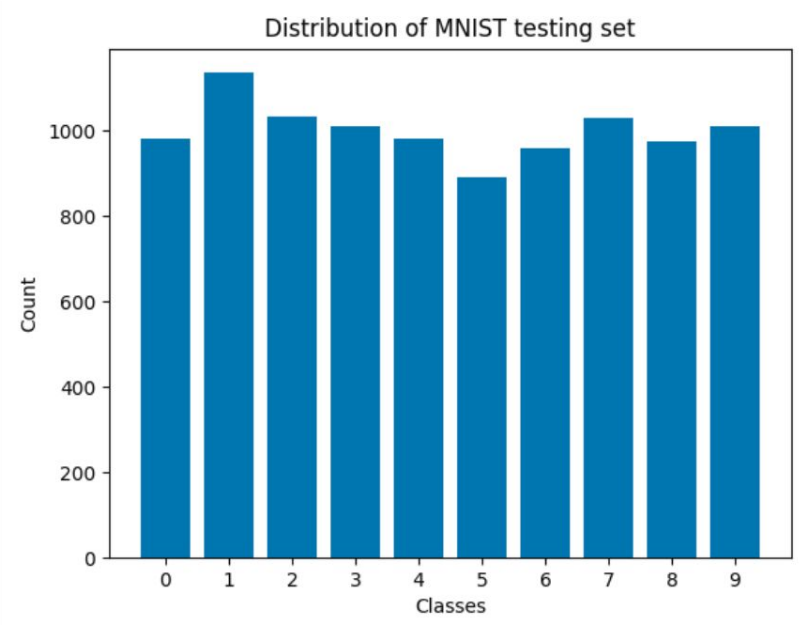
	Original Accuracy	Adversarial Accuracy
Unaltered	0.968	0.00065
Threshold	0.958	0.947
Gaussian	0.967	0.816
Median	0.966	0.896
Bilateral	0.967	0.868
Resize	0.956	0.738

Data Sets & Networks

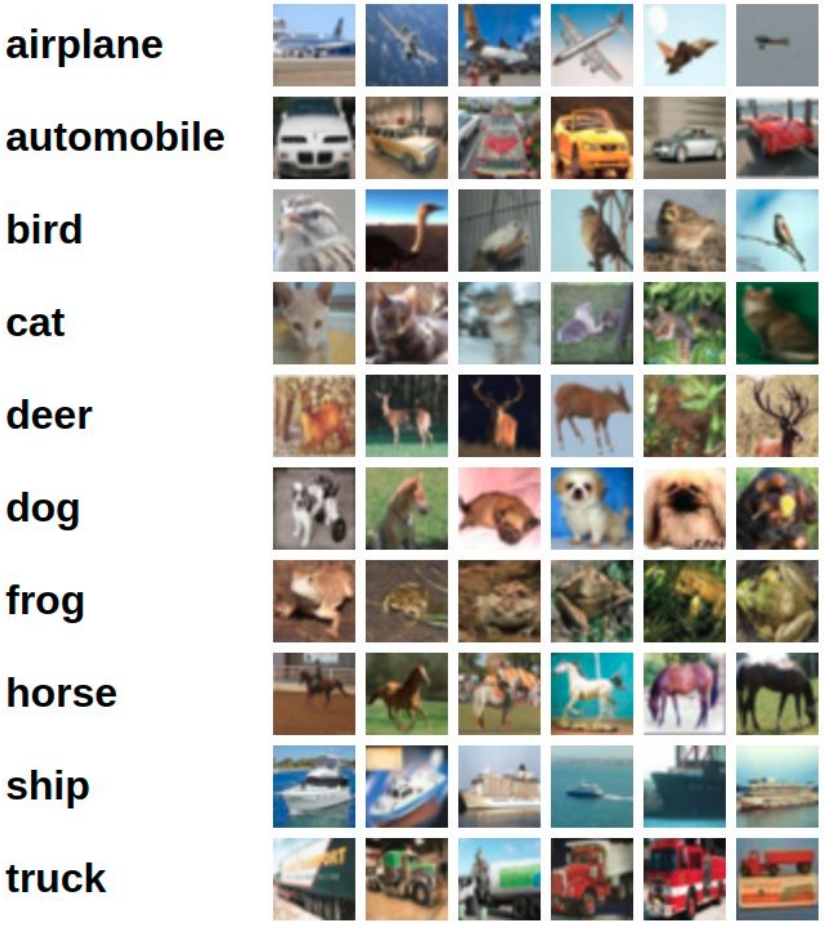
MNIST Dataset of Handwritten Digits

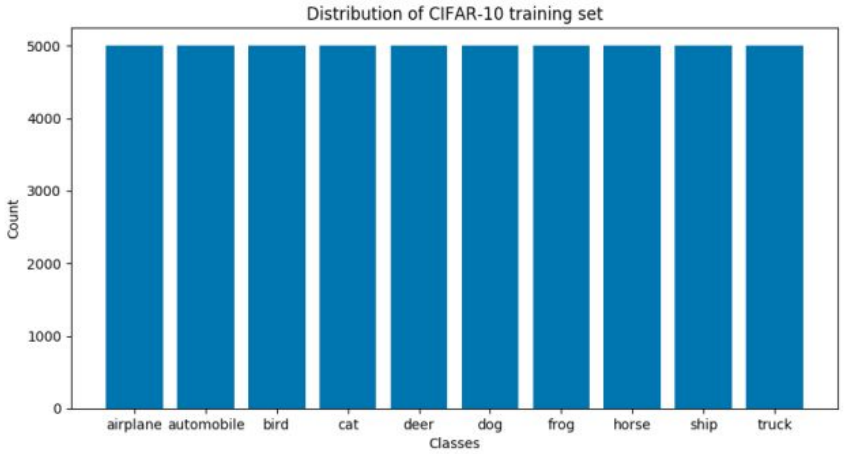


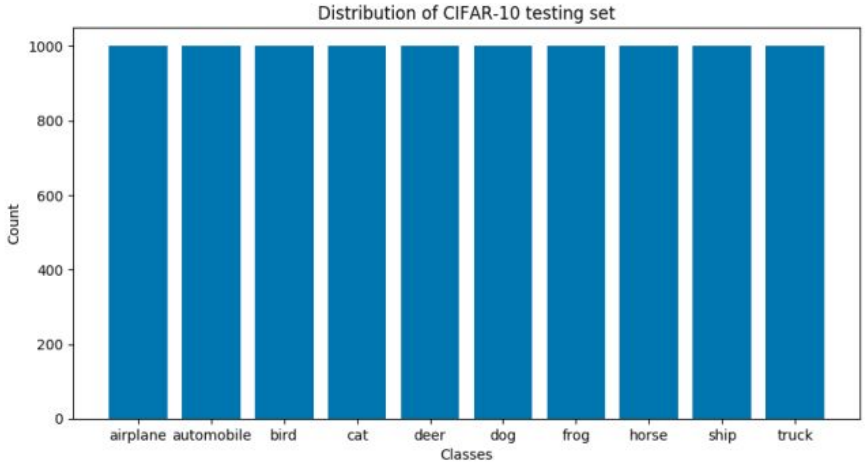




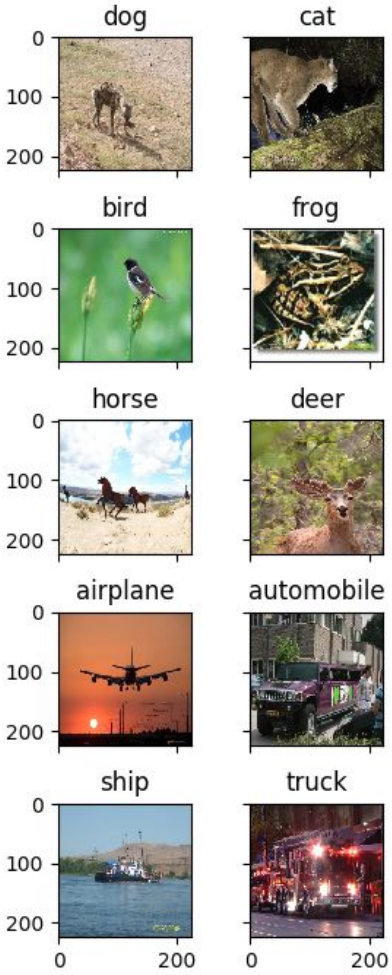
CIFAR-10





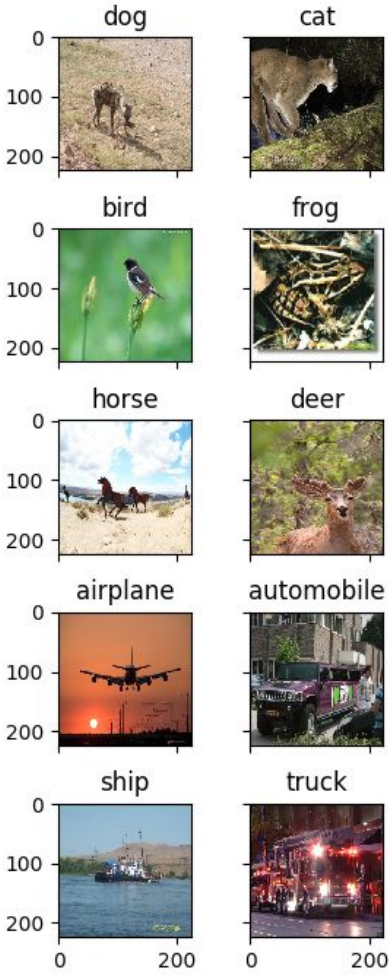


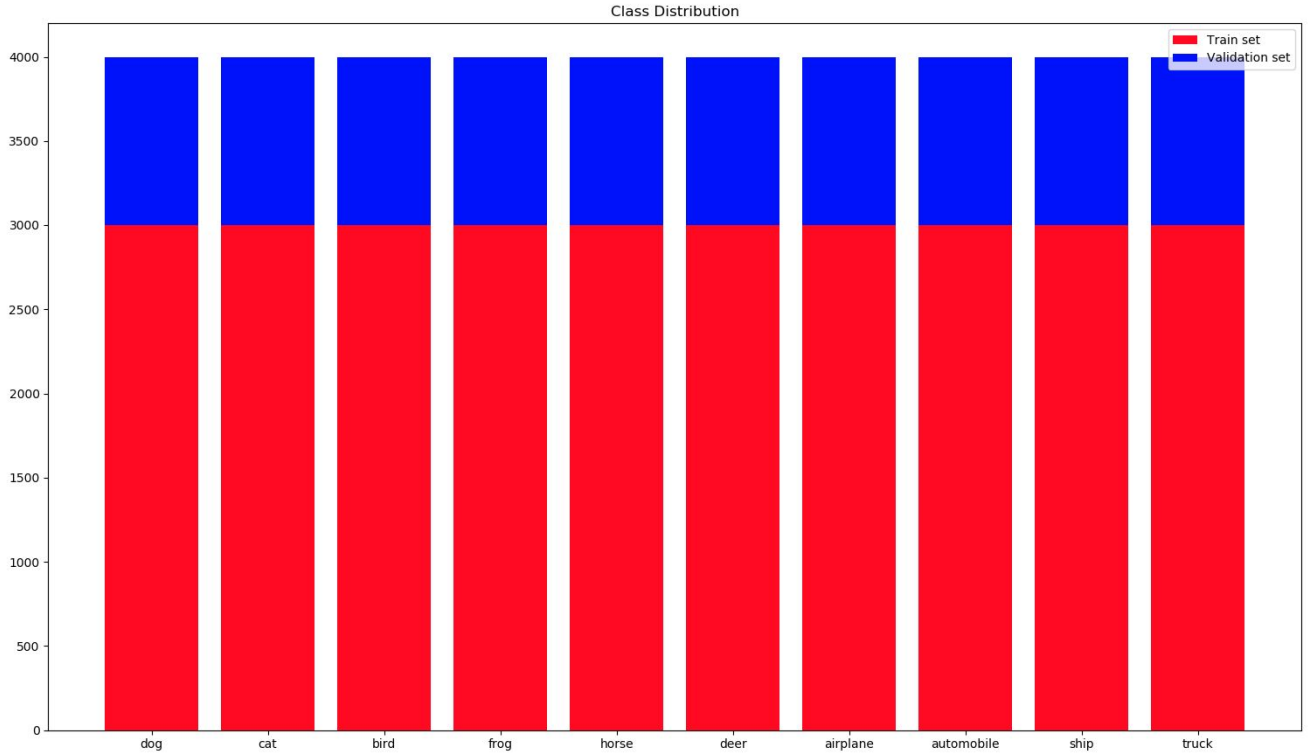
CINIC-10 Large



- Custom dataset of 224 by 224 images gathered by cross-referencing CINIC-10's source file and ImageNet's URLS.
- Retrieved ~80,000 train images and ~40,000 test images.
- Had to reduce set size later to 30,000 train images and 10,000 test images.

CINIC-10 Large

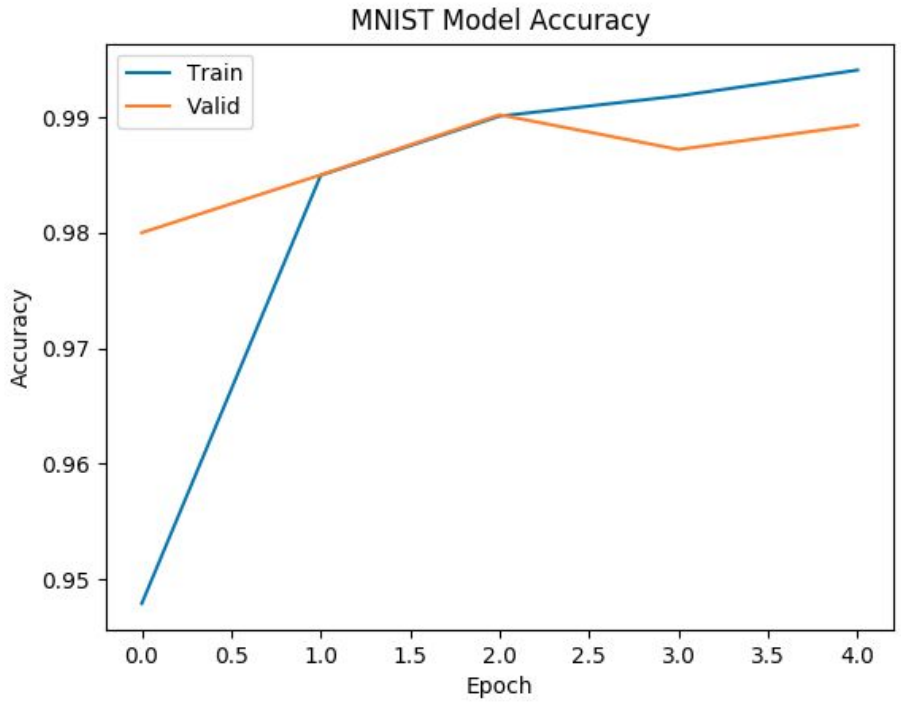




Class distribution after data set reduction.

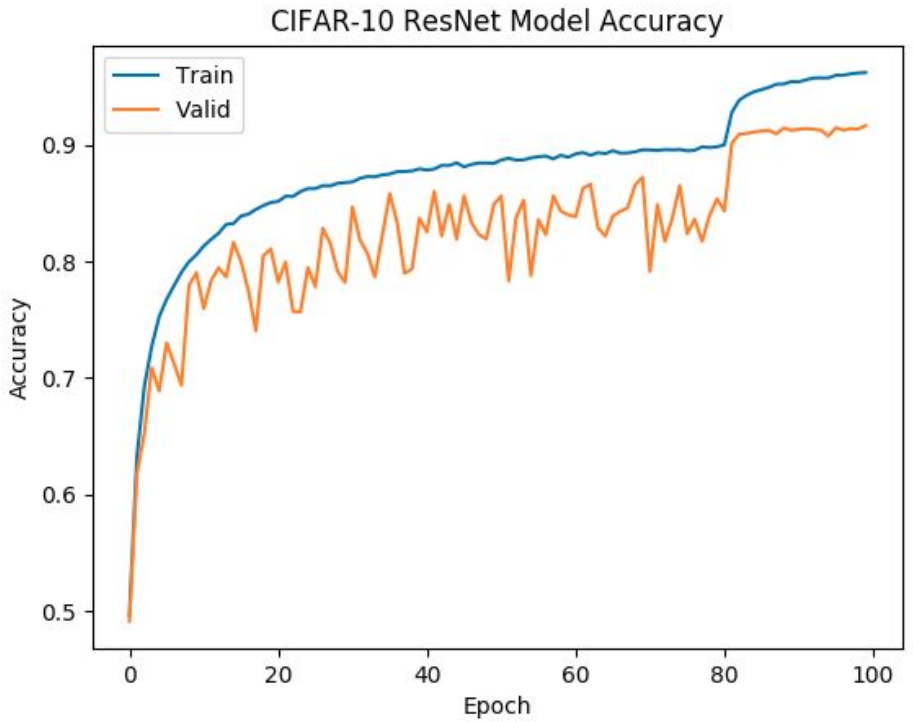
MNIST Network

- Simple network, achieved 99% accuracy on the testing set after just a couple of epochs.



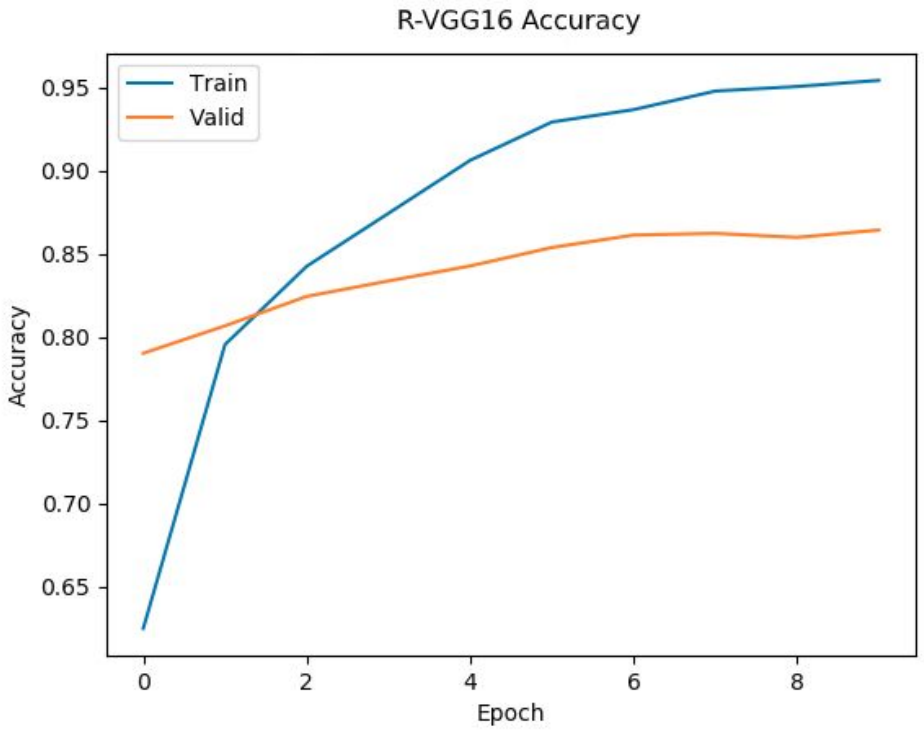
CIFAR-10 Network

- Use a residual neural network (ResNet) to achieve high accuracy on the testing set.
- Achieves 91% accuracy on testing set at around 100 epochs.

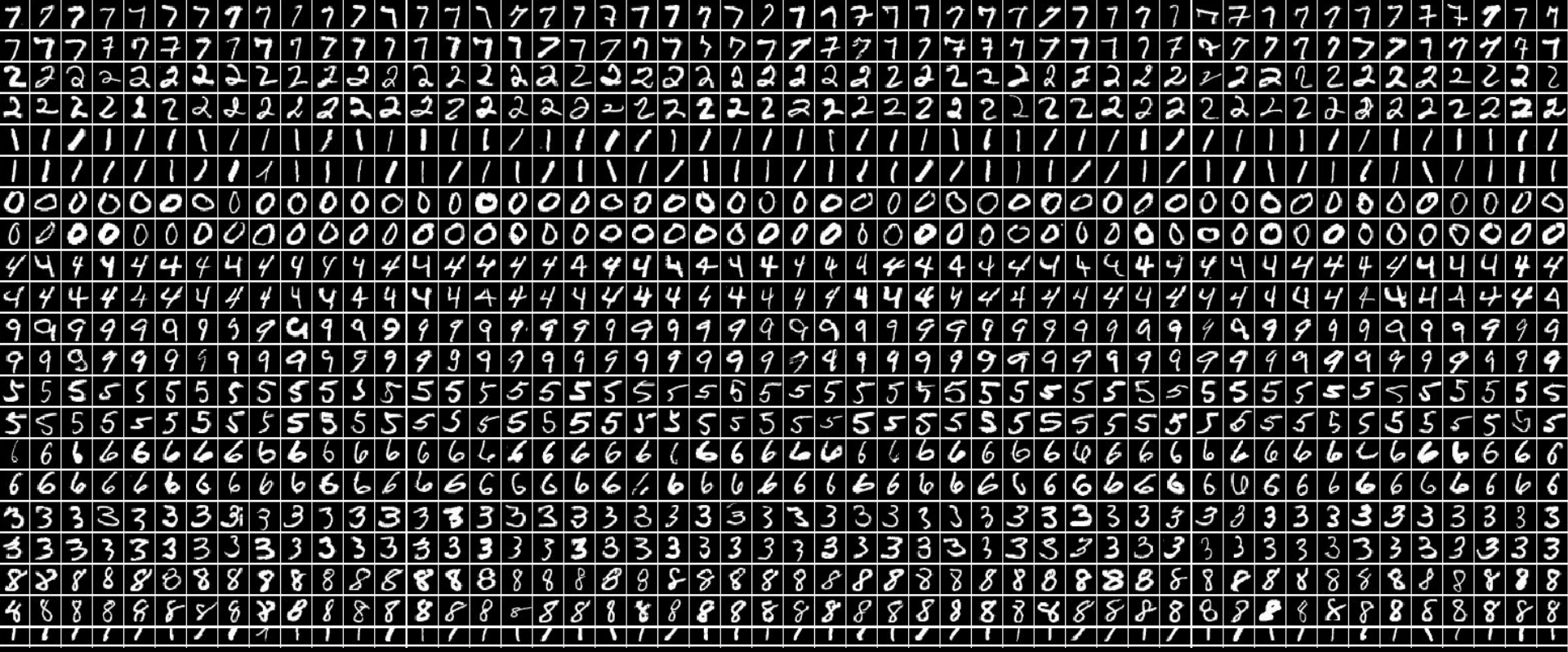


CINIC-10 Large Network

- Transfer learning with a VGG16 model trained on ImageNet, plus **lots** of regularization.
  - VGG16 out of the box did not produce enough gradients.
- Will be referred to as R-VGG16 (Regularized VGG16).
- Achieves 85% accuracy at 10 epochs.



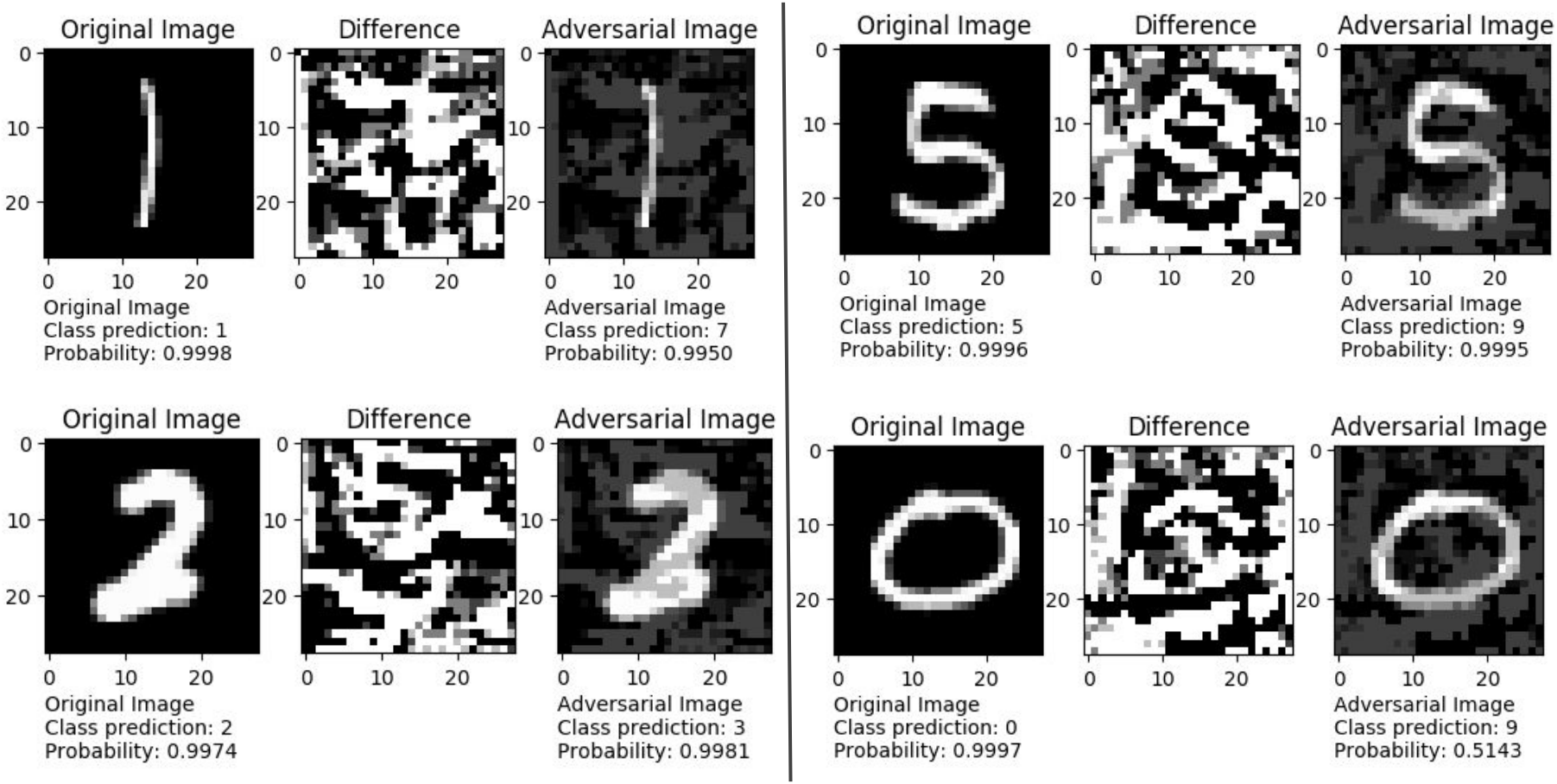
Defending Against Adversarial Examples



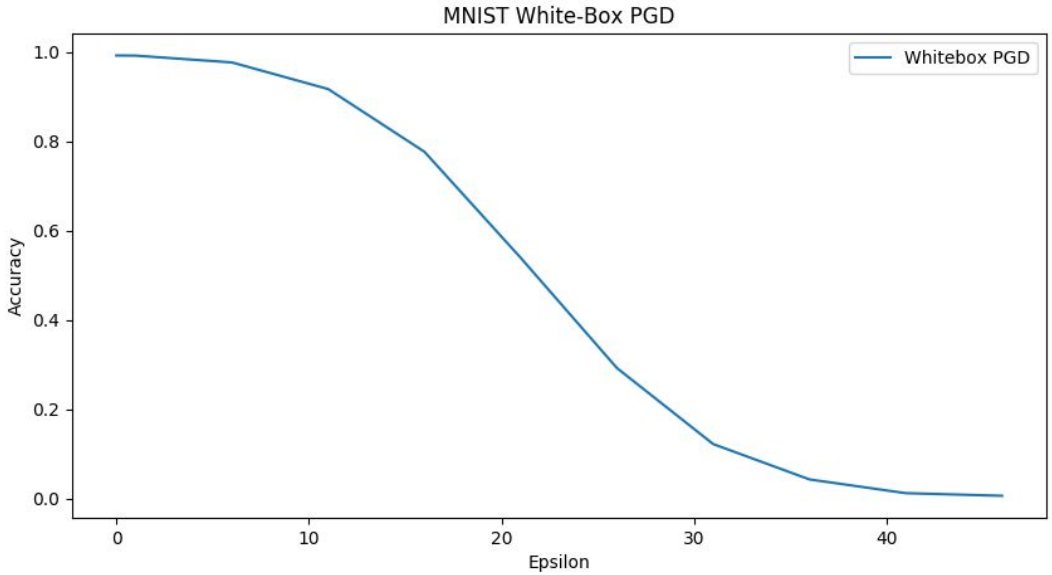
Defending

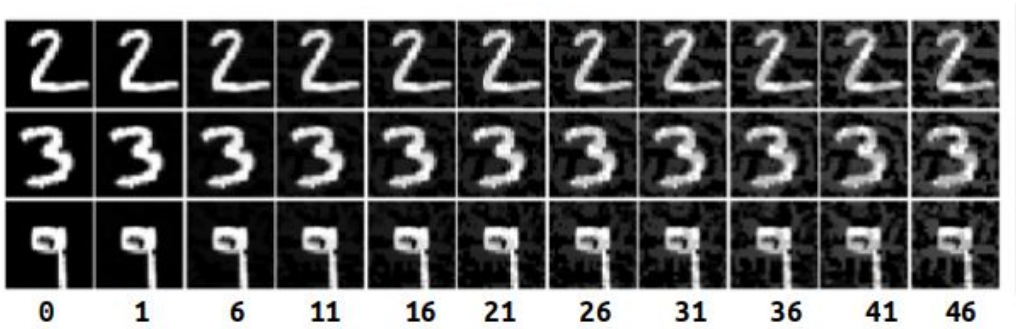
MNIST

MNIST White-box PGD



MNIST White-box PGD



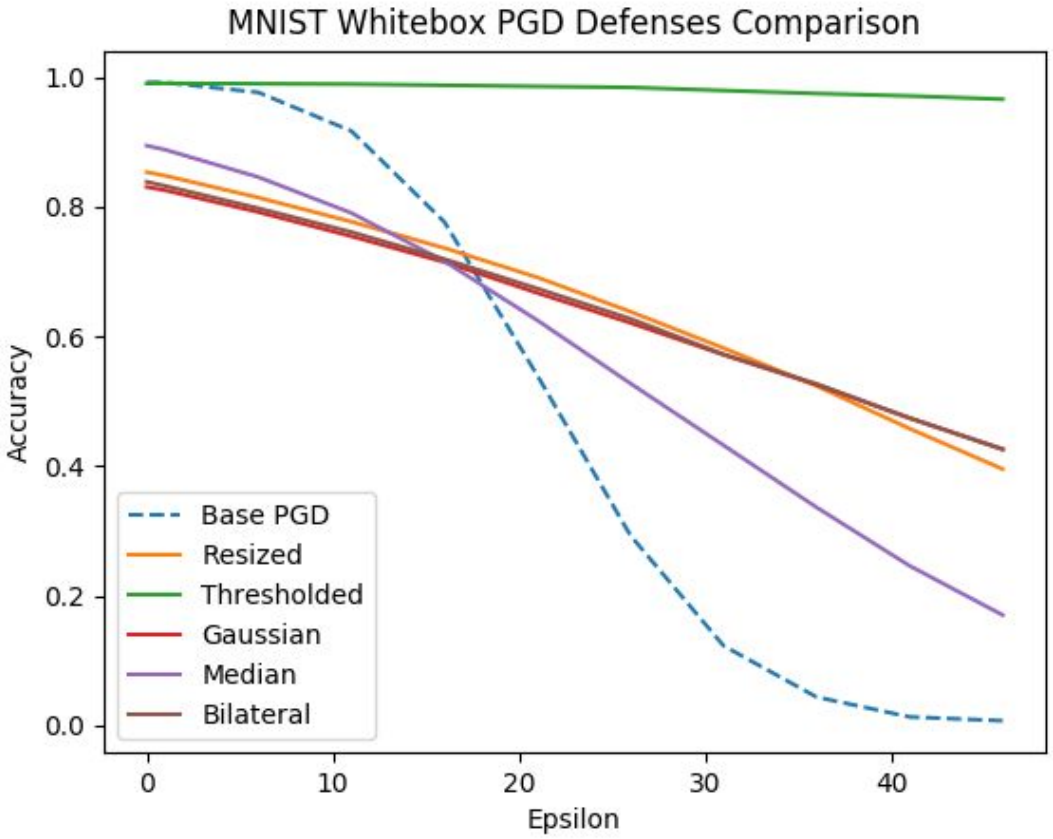


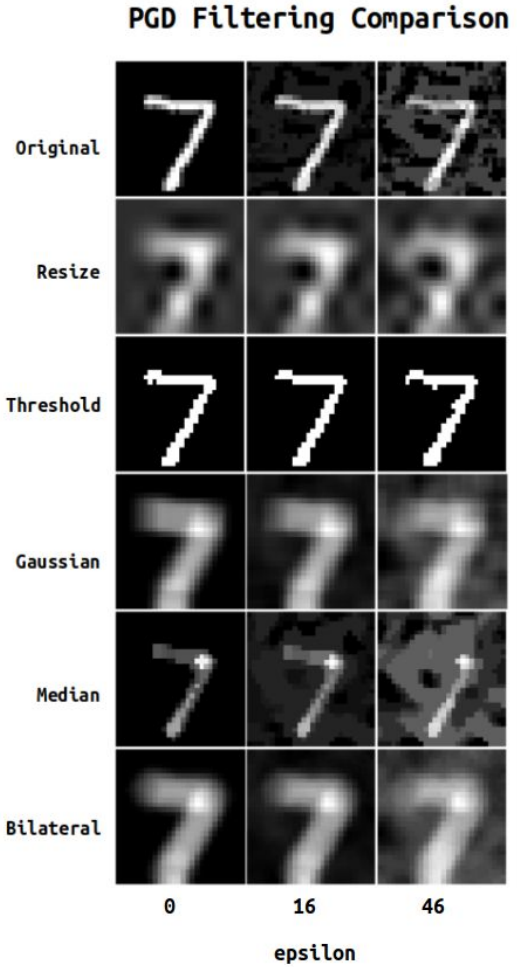
MNIST Filtering Defenses

- **Resizing**
  - Downsample to 7 by 7
- **Thresholding**
  - Binarize at value of 100
- **Gaussian filtering**
  - Kernel size of 7 by 7 with sigma of 10
- **Median filtering**
  - Kernel size of 5 by 5
- **Bilateral filtering**
  - Kernel size of 9 by 9, range and domain of 100.

Note: All values hand-tuned based on accuracy scores.

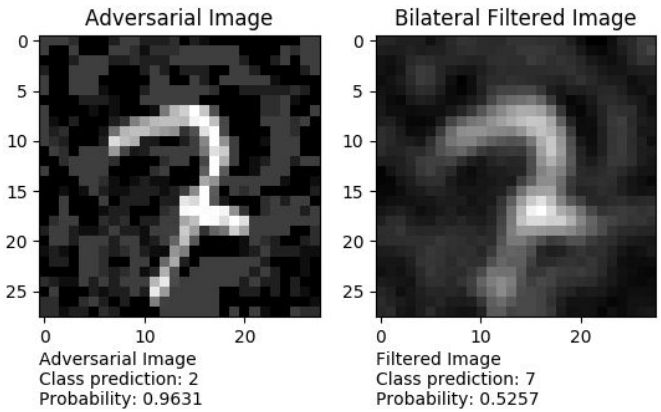
MNIST White-box PGD Defenses



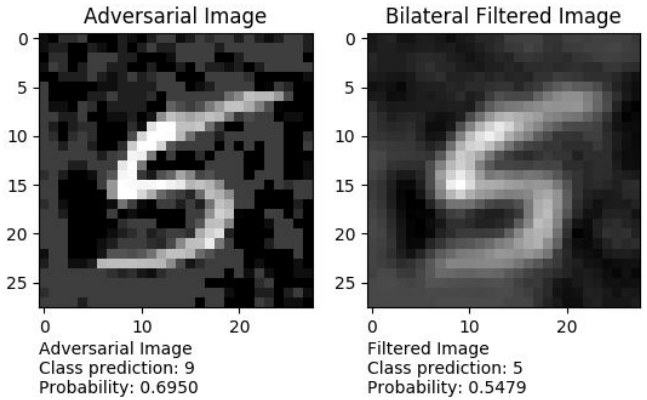


Bilateral Defense in Action

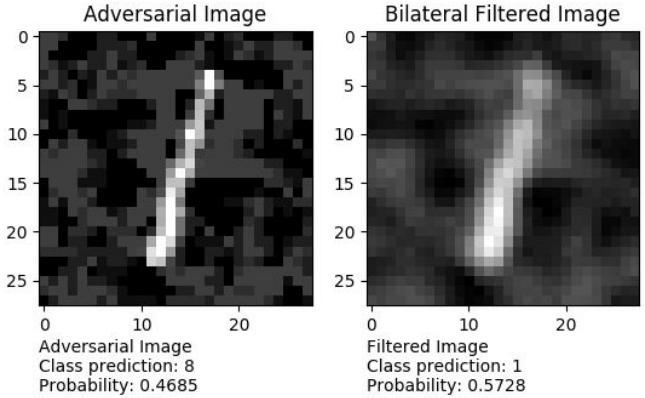
PGD 7 Example



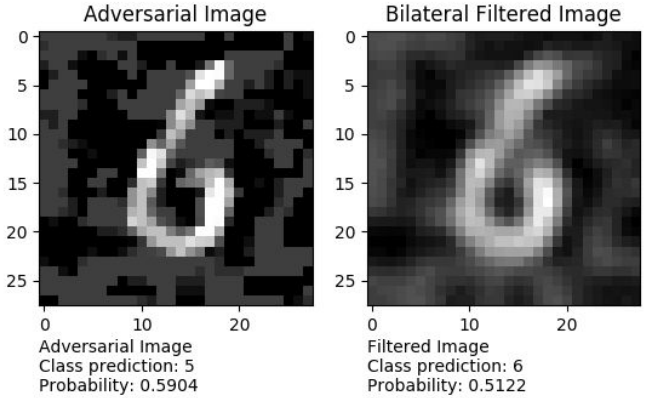
PGD 5 Example



PGD 1 Example

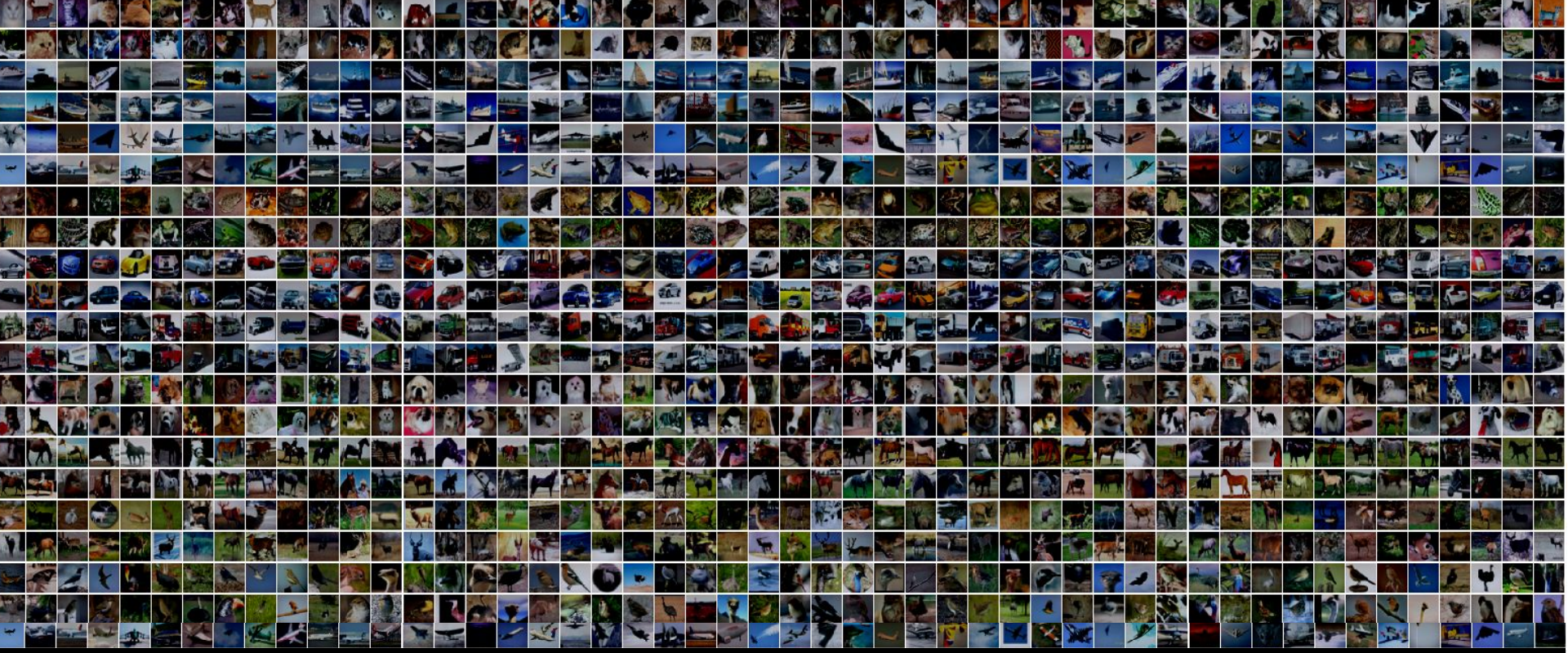


PGD 6 Example



MNIST Conclusion

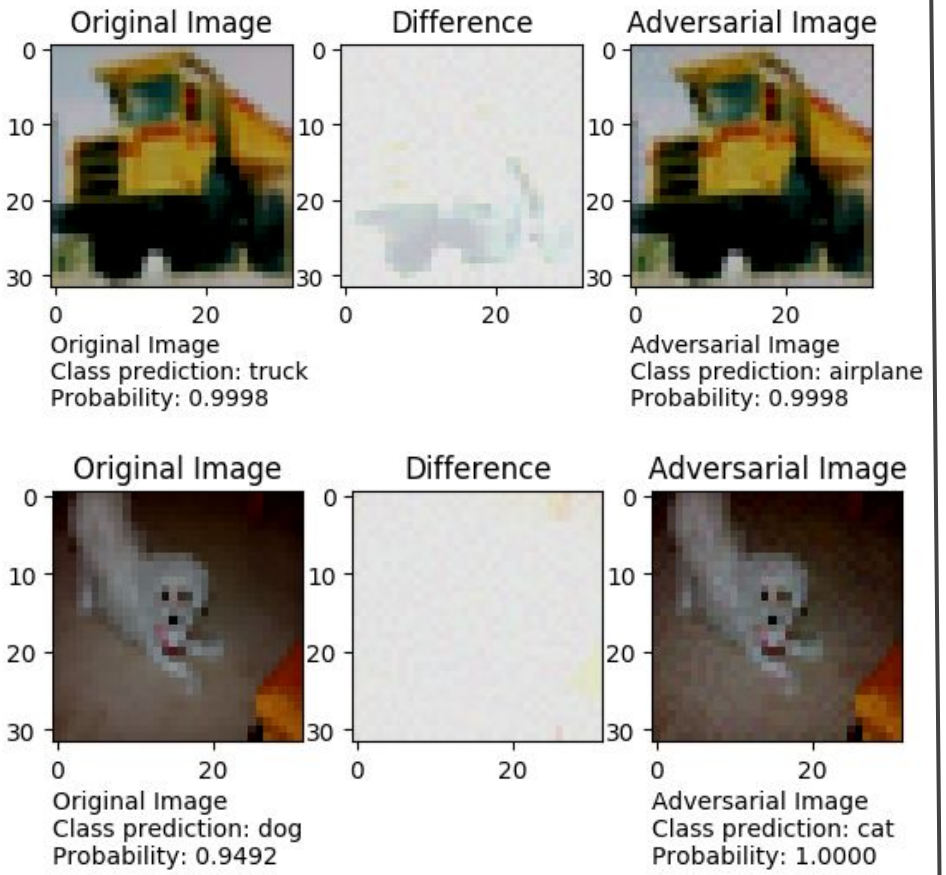
- **Filtering was effective at increasing adversarial accuracy!**
- Best performing filters:
  - Bilateral
  - Resize
- Bilateral filter increased accuracy from 0.6% to 43% on the strongest adversary.
- However, MNIST's images are too simple and not realistic.
  - They require a significant amount of adversarial perturbation.
- Needed to move to larger images.

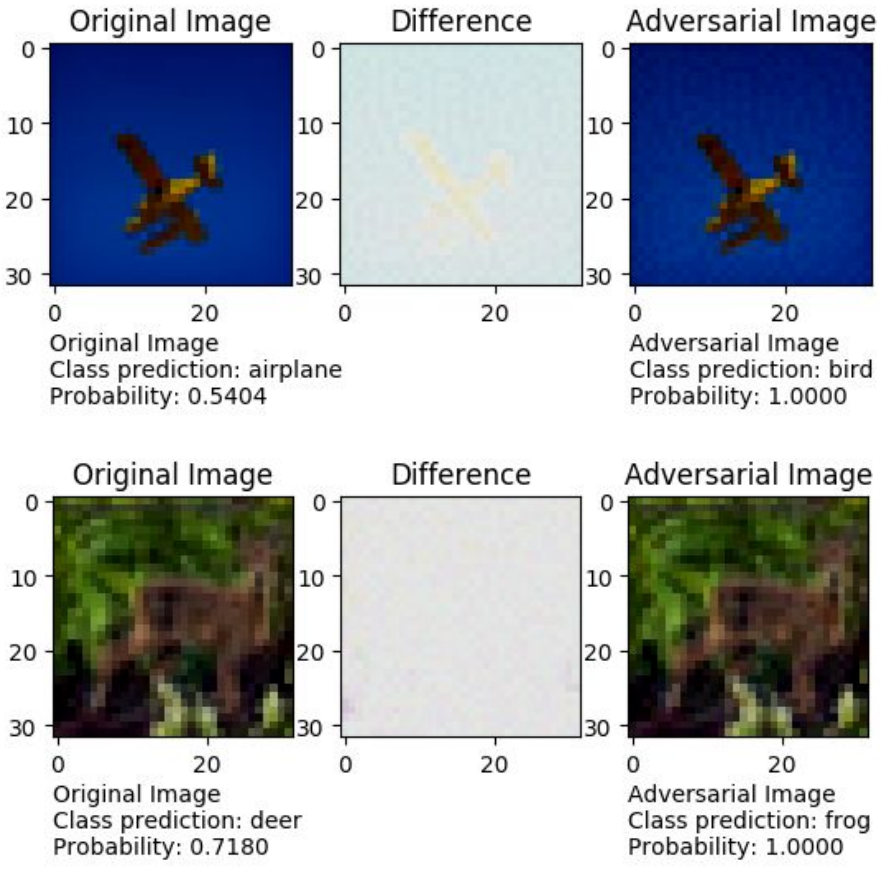


Defending

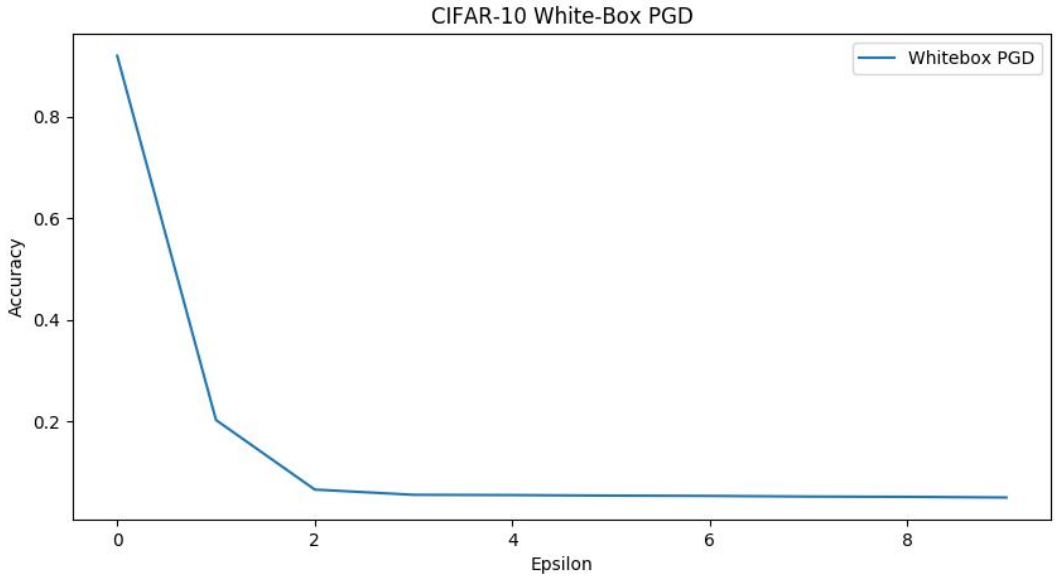
CIFAR-10

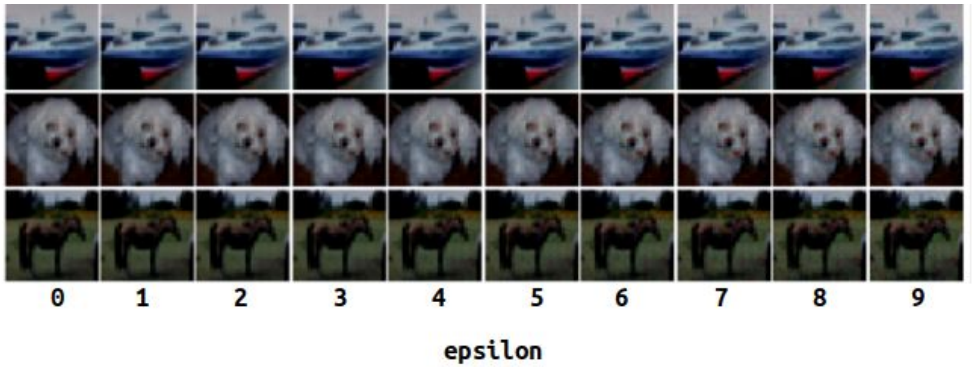
CIFAR-10 White-box PGD





CIFAR-10 White-box PGD



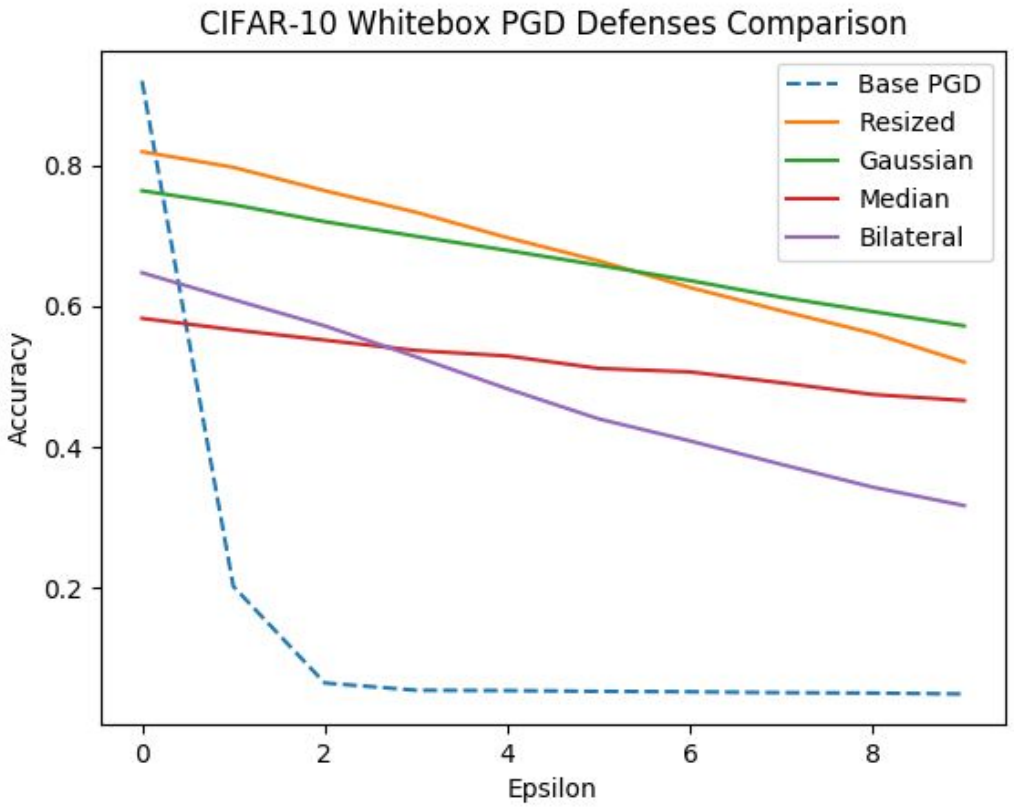


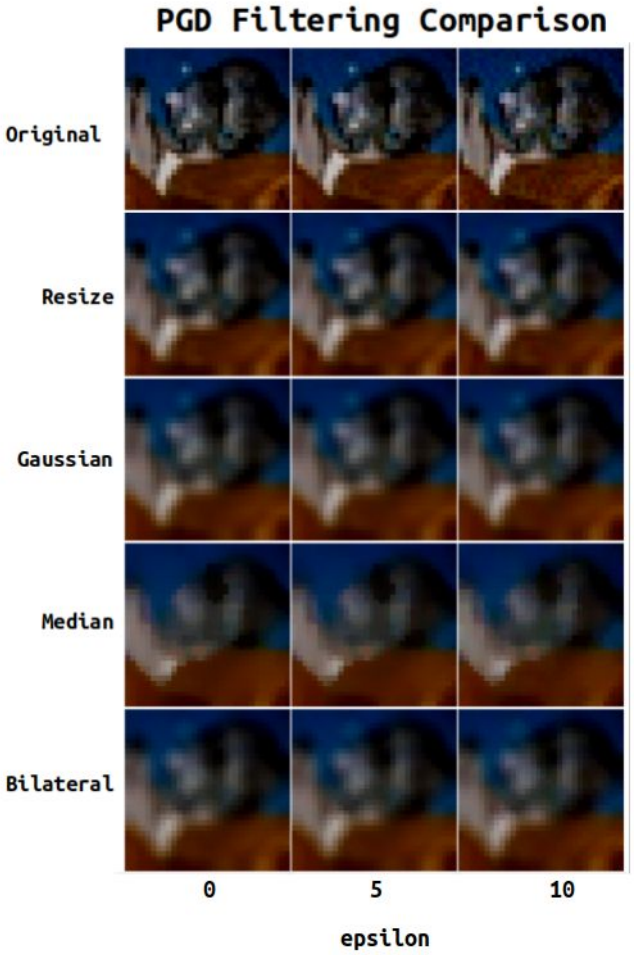
CIFAR-10 Defenses

- **Resizing**
  - Downsample to 17 by 17
- **Gaussian filtering**
  - Kernel size of 5 by 5 with sigma of 1
- **Median filtering**
  - Kernel size of 5 by 5
- **Bilateral filtering**
  - Kernel size of 5 by 5, range and domain kernels of 100

Note: All values hand-tuned based on accuracy scores.

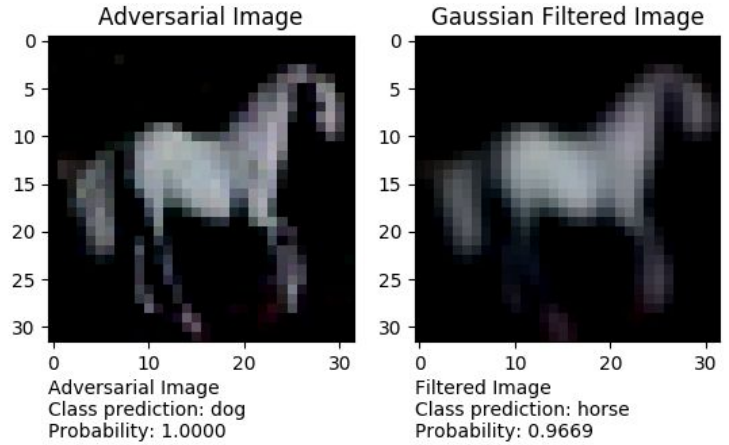
CIFAR-10 White-box PGD Defenses



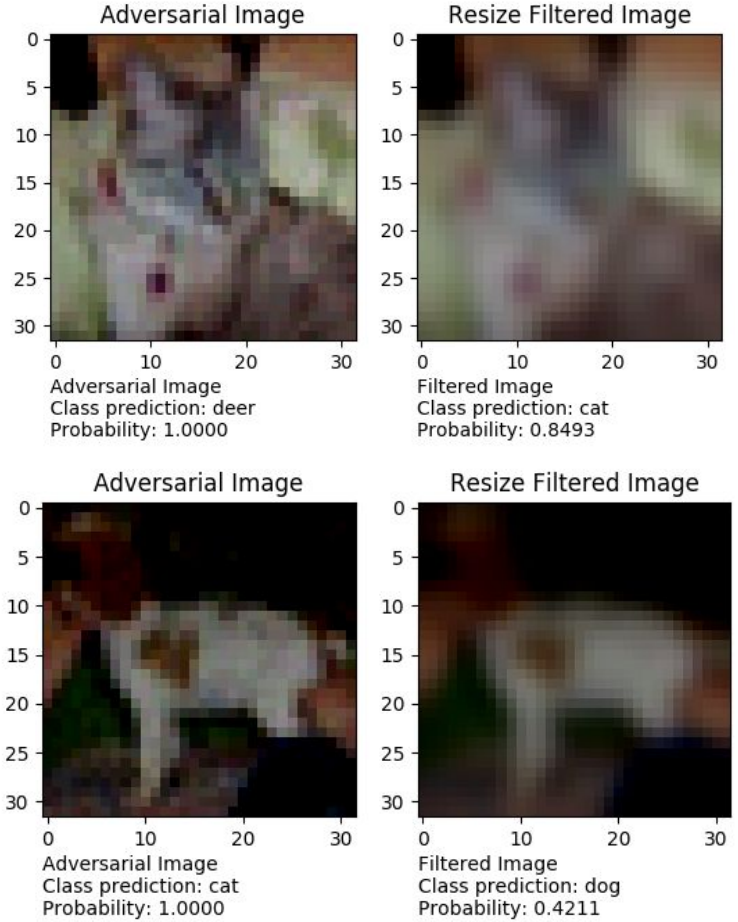


Defense in Action

Gaussian

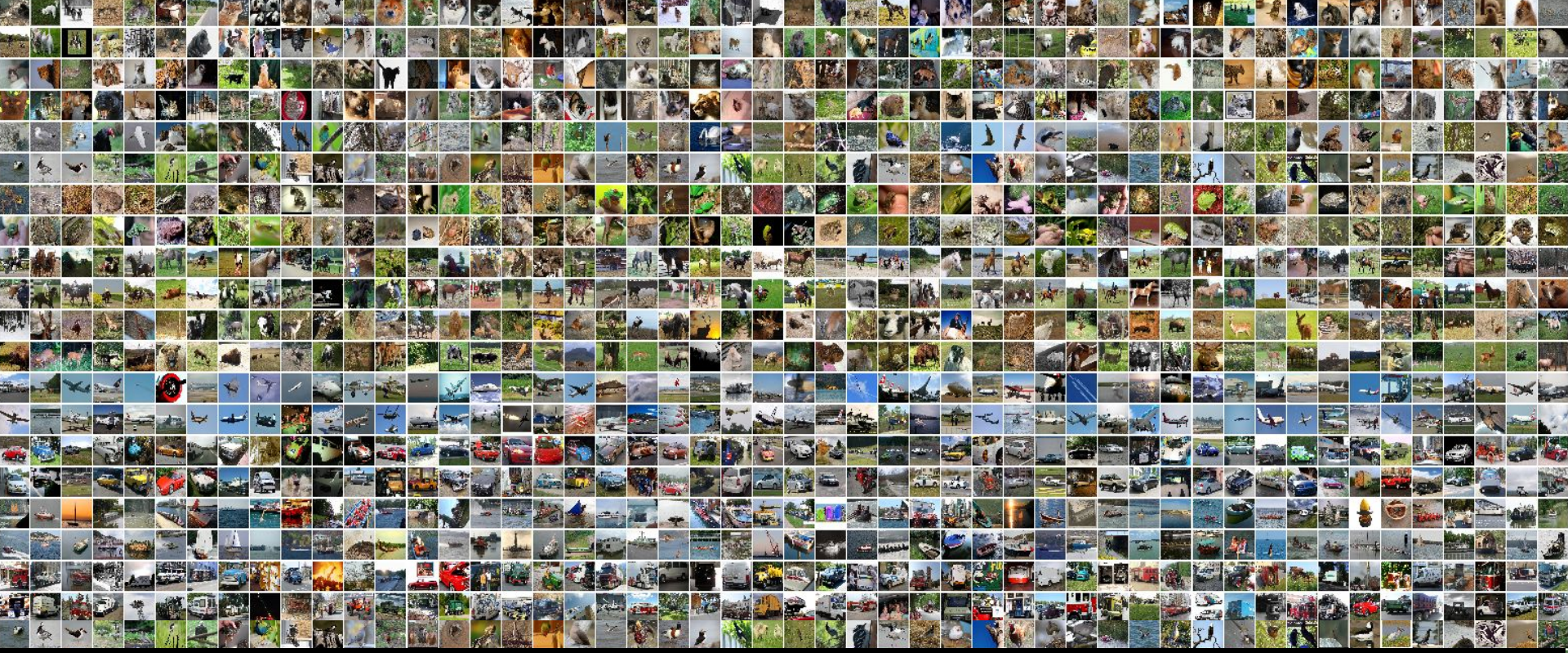


Resize



CIFAR-10 Conclusions

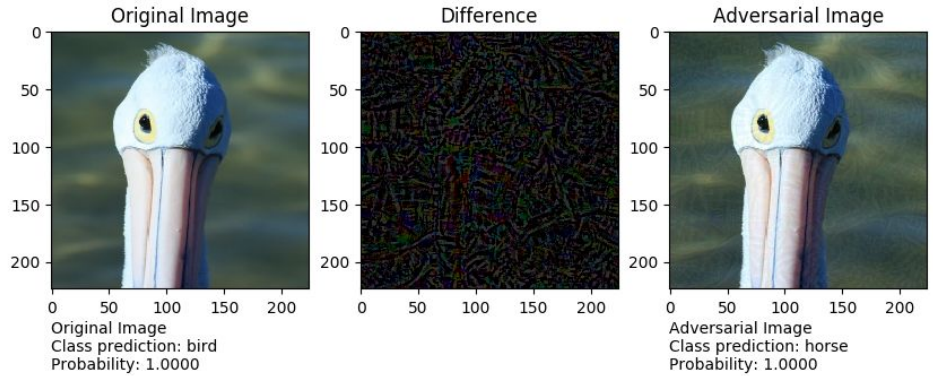
- **The filtering increased accuracy!**
- Best performing filters:
  - Gaussian
  - Resizing
- The Gaussian filter increased accuracy from ~8% to ~60% on the strongest attack.
- The images are very small, however, so to have a wider range of experimentation, it was time to move onto the big image data set.

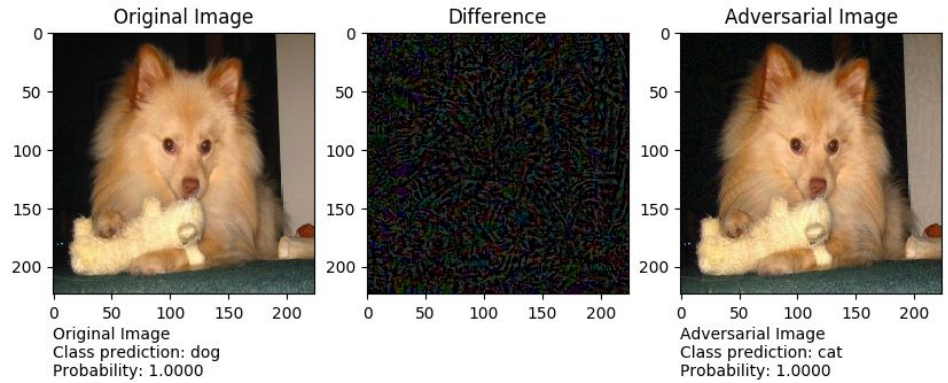


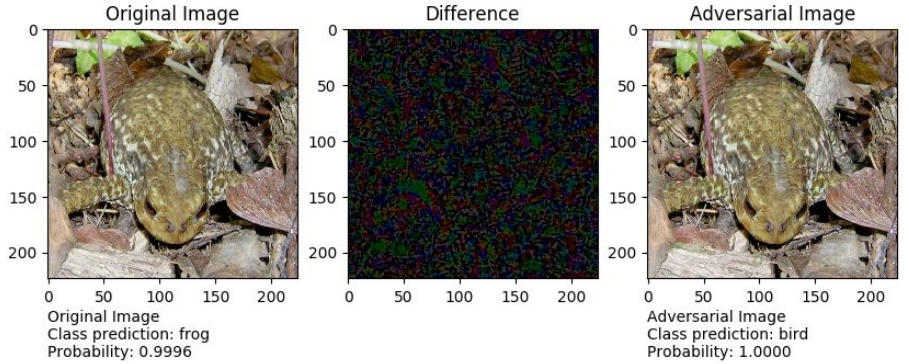
Defending

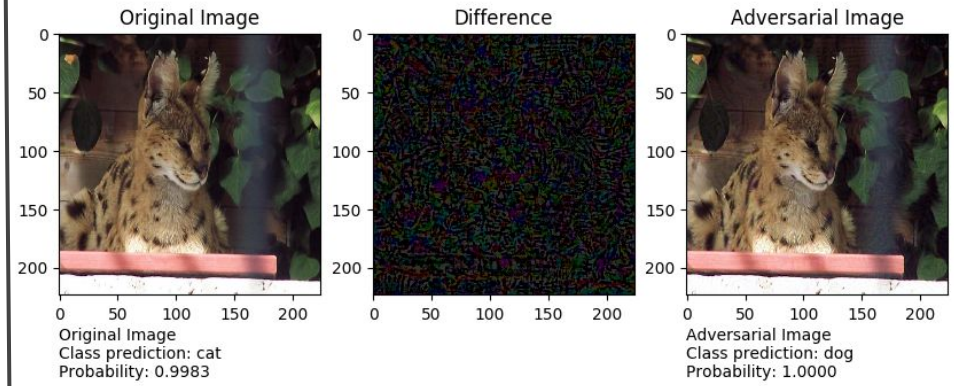
CINIC-10 Large

CINIC-10 Large PGD

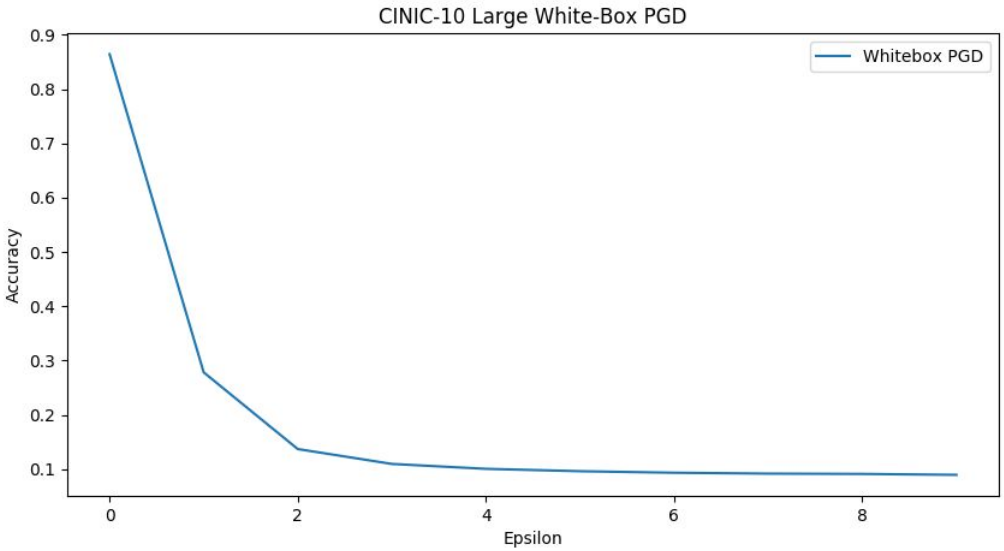


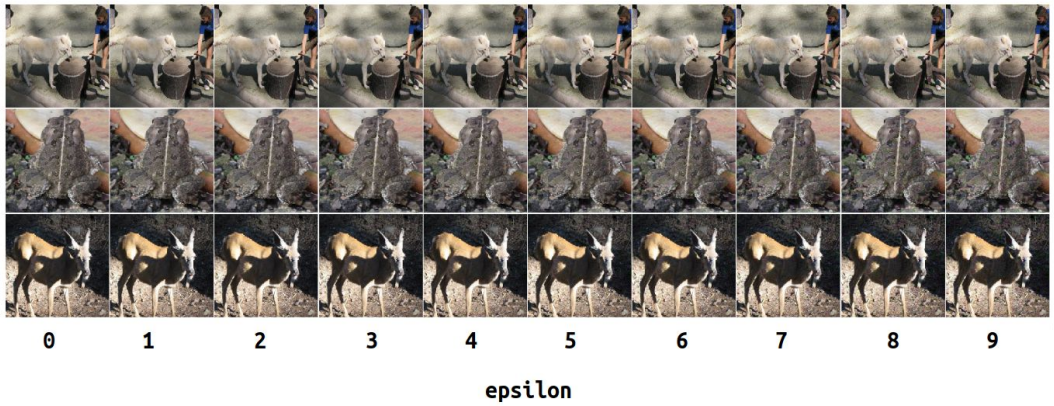






CINIC-10 Large PGD



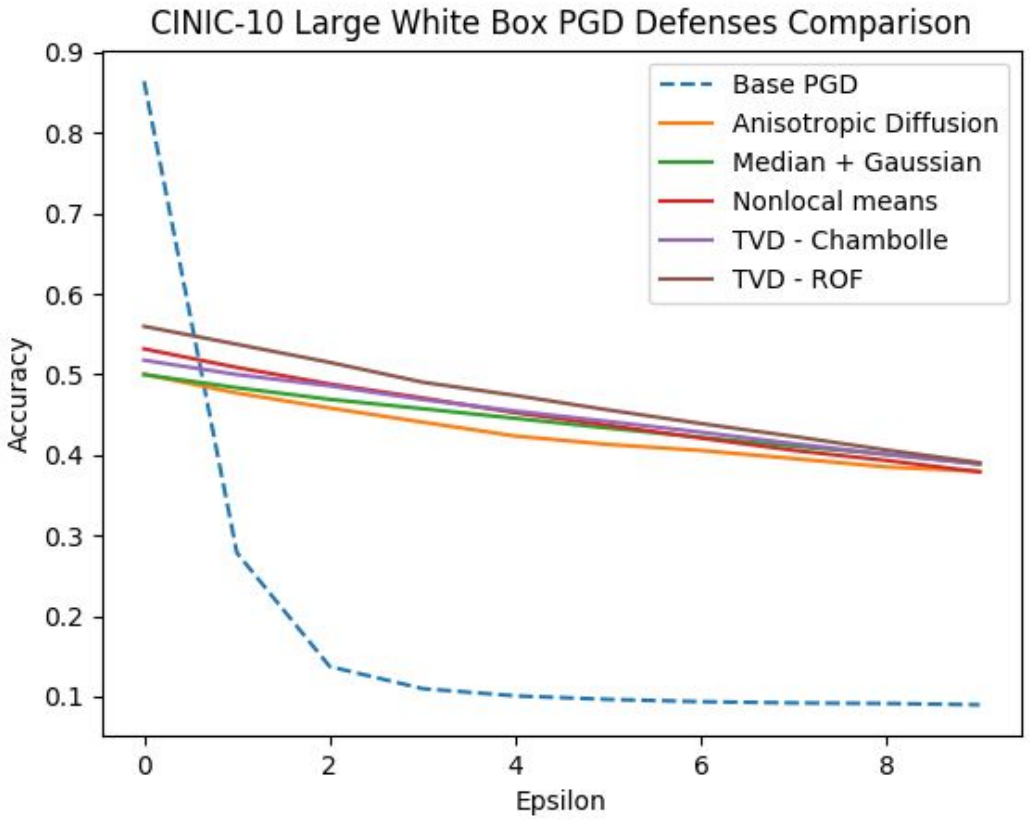


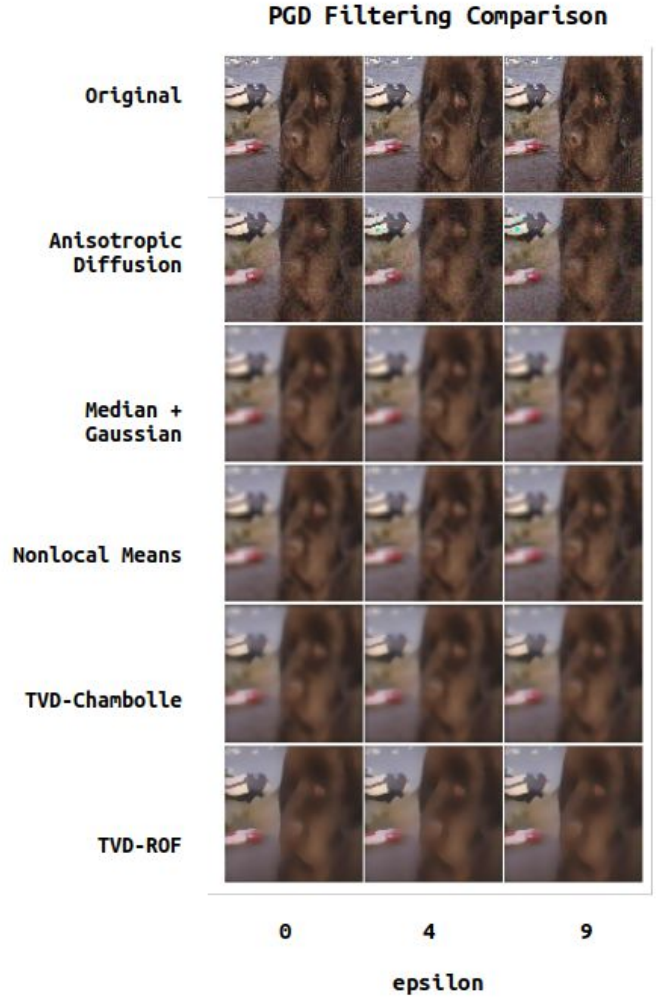
CINIC-10 Large Defenses

- **TVD - ROF**
  - Weight of 65
- **TVD - Chambolle**
  - Weight of 130
- **Non-local Means**
  - Sigma of 10, h of 60
- **Anisotropic Diffusion**
  - K of 0.1, 20 iterations
- **Gaussian + median filter**
  - Median kernel size of 5 by 5, sigma of 20, Gaussian kernel size of 11 by 11

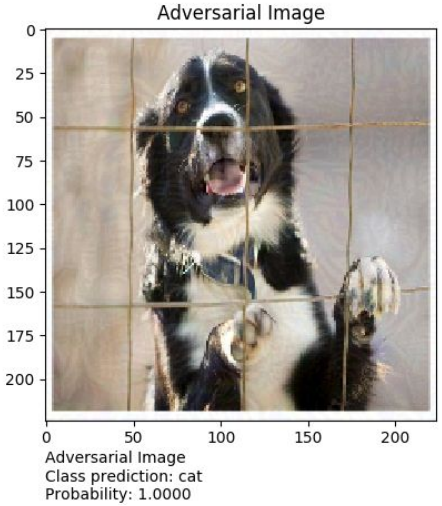
Note: Values selected via parameter search based on original and highest epsilon accuracy scores on white-box PGD.

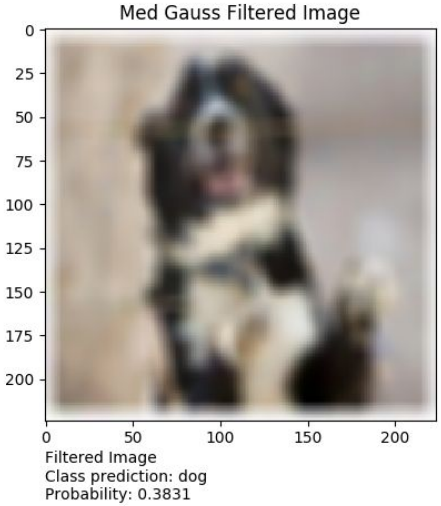
CINIC-10 Large PGD Defenses

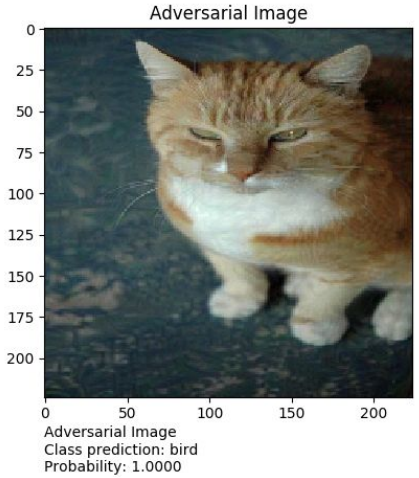


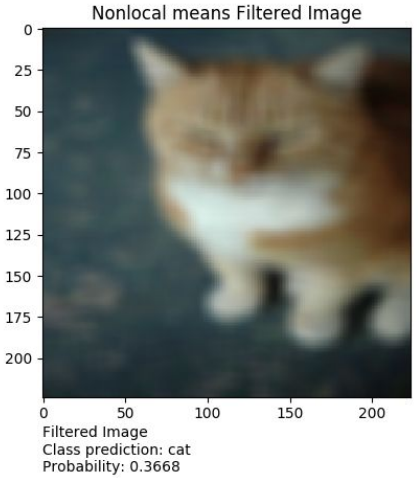


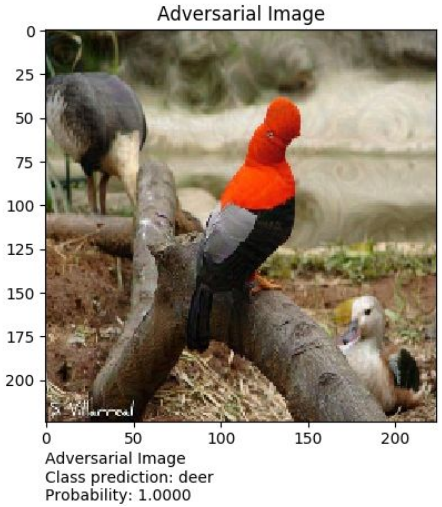
Defenses in Action

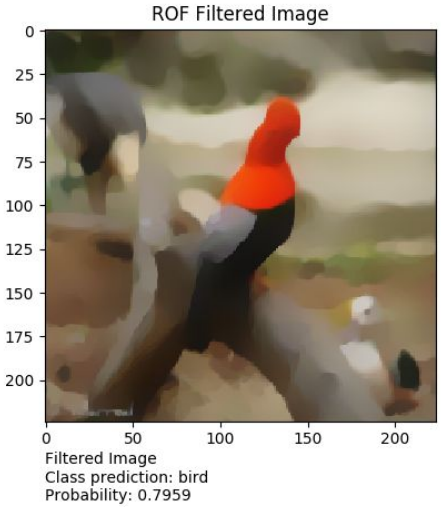


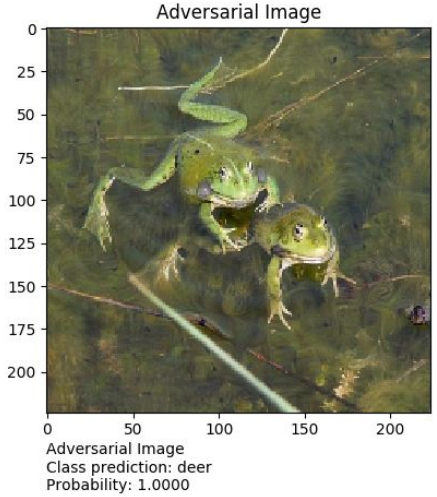


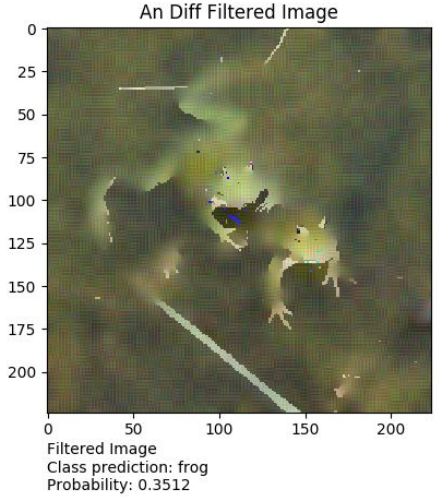








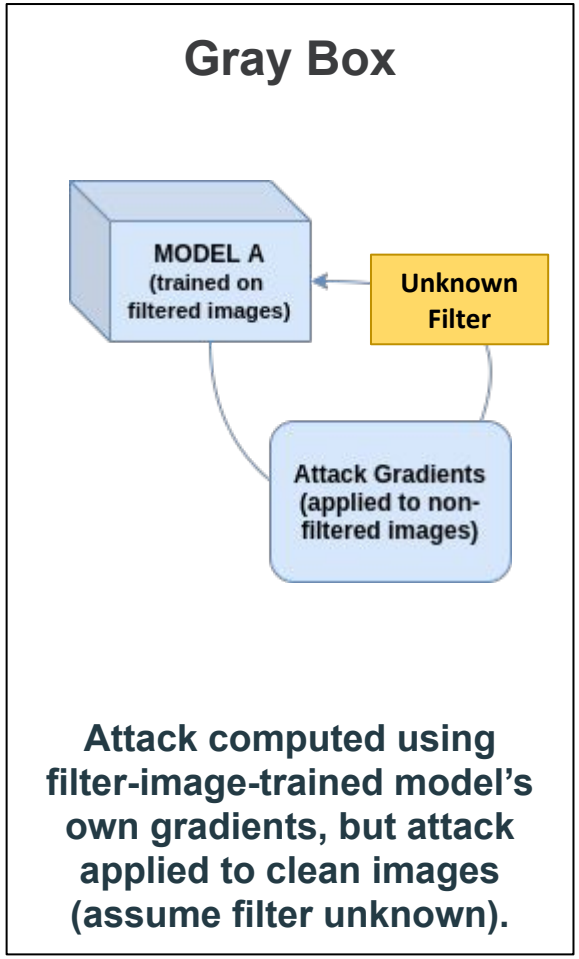




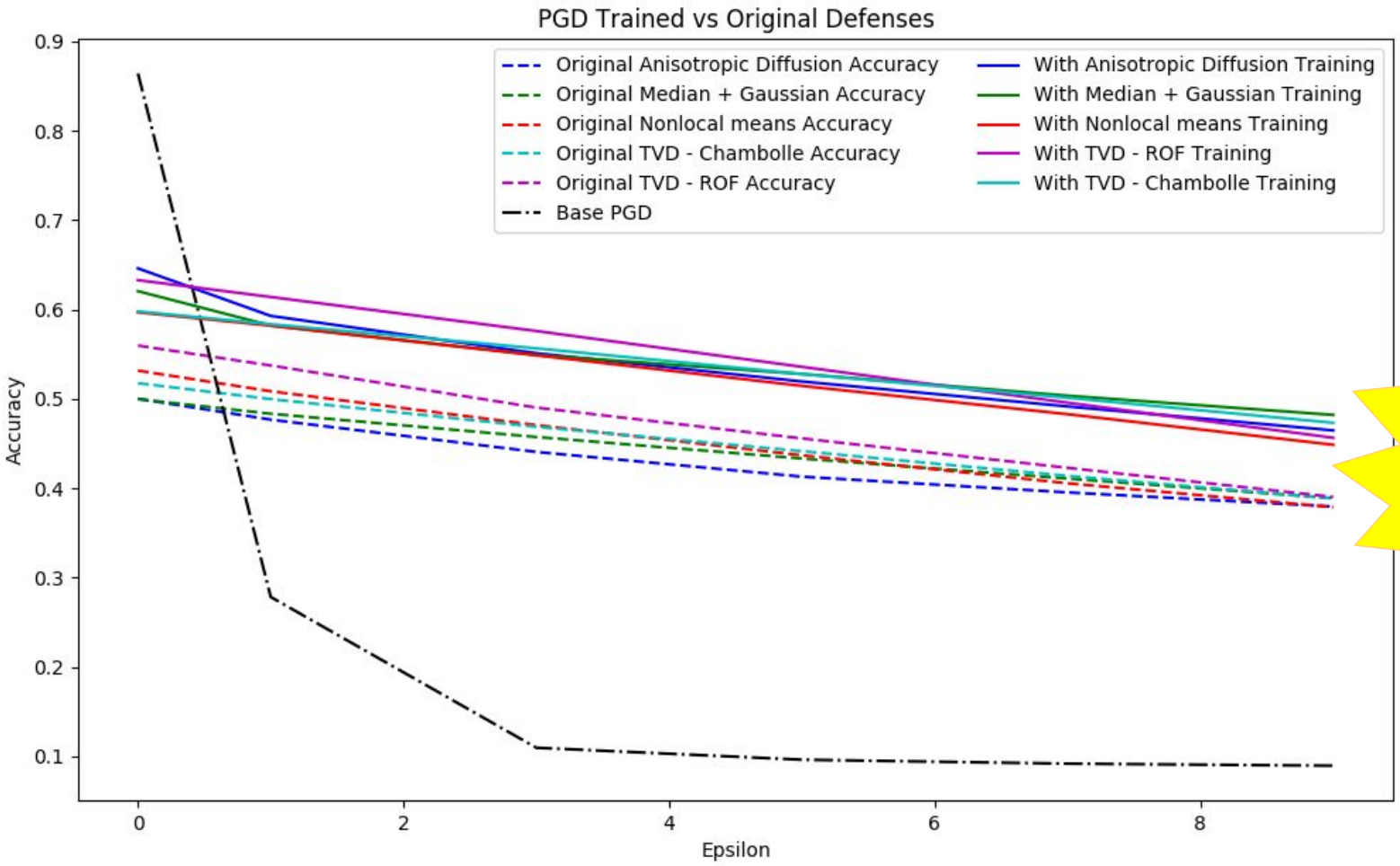
CINIC-10 Large Gray-Box

- In order to ***increase the classifier accuracy on the filtered images***, 5 new models were created and trained on each respective set of filtered images.
- ***For a fair comparison***, each model was then attacked with the white-box PGD attack.
- This is referred to as "**gray-box**" because **the attacker knows the model parameters but not the filter being used.**

Gray-box Reminder



CINIC-10 Large PGD Gray-box



Training on filtered images increased accuracy!

CINIC-10 Large Conclusions

- **The filtering increased accuracy!**
- **Training on the filtered images improved accuracy significantly.**
- Best performing filter:
  - Median + Gaussian, by a hair
- All filters increased accuracy from 7% to ~55% on average on the strongest attack.

Conclusions

Restatement of Hypothesis

- Adversarial images are a result of ***pixel manipulation.***
- Therefore, ***additional pixel manipulation (filtering)*** that masks adversarial noise should ***reduce adversariality.***
  - However, filtering the images will cause reduction in classifier accuracy on clean images.

Conclusion

- The results support the hypothesis...

Filtering the adversarial images reduces their adversariality.

Conclusions

- Training the network on filtered images is important to increase accuracy.
- The best performing filters were the ones that removed unnecessary noise from the image while preserving the image shape.
  - This can be explained by removal of non-robust features, which are vulnerable to adversarial noise (Ilyas, 2019).

Conclusions

- Winners in smaller images are the resizing and Gaussian filters.
  - Gaussian filtering in CIFAR-10 improved accuracy from 8% to 65% on the strongest epsilon.
- Winner in larger images is the median + Gaussian filter (by a hair).
  - Training on the filtered images increased accuracy from 7% to 57% on the strongest epsilon.

Questions?

References

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.,and Fergus, R. (2013).Intriguing properties of neural networks.arXiv preprintarXiv:1312.6199.

[2] Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A.,Kohno, T., and Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1625–1634.

[3] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world.arXiv preprint arXiv:1607.02533.

References

[4] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.arXiv preprint arXiv:1412.6572.

[5] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. (2019). Adversarial examples are not bugs, they are features. In Advances in Neural Information Processing Systems, pages 125–136.

[6] Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In Proceedings of the aaai conference on artificial intelligence, volume 33, pages 4780–4789.

Appendix

Cause of Adversarial Examples

What causes adversarial examples?

- ***The linear components in a CNN*** (Goodfellow, 2014).
- Just a few small “wiggles” in high dimensional input multiplies to create a large change in the output.
- The following equation explains how linear behavior transforms the original image ***x*** into the adversarial image ***x’*** (Goodfellow, 2014).

$$w^\top \tilde{x} = w^\top x + w^\top \eta$$

Exploiting linear behavior : $w^\top \tilde{x} = w^\top x + w^\top \eta$

- The precision of features in an image is limited (8 bit pixel)
- If perturbation **η** is applied to image **x** and is below the precision, the classifier should not respond differently to **x** and **x** + **η**.
- However, with high dimensional input, the perturbation will be multiplied by the weight vector **ω** with dimensionality *n* and average magnitude *m.*
- The perturbation can grow as much as ϵmn if **η** = sign(**ω**)
  - ϵ is the max value below the precision.

Filter Details

Thresholding

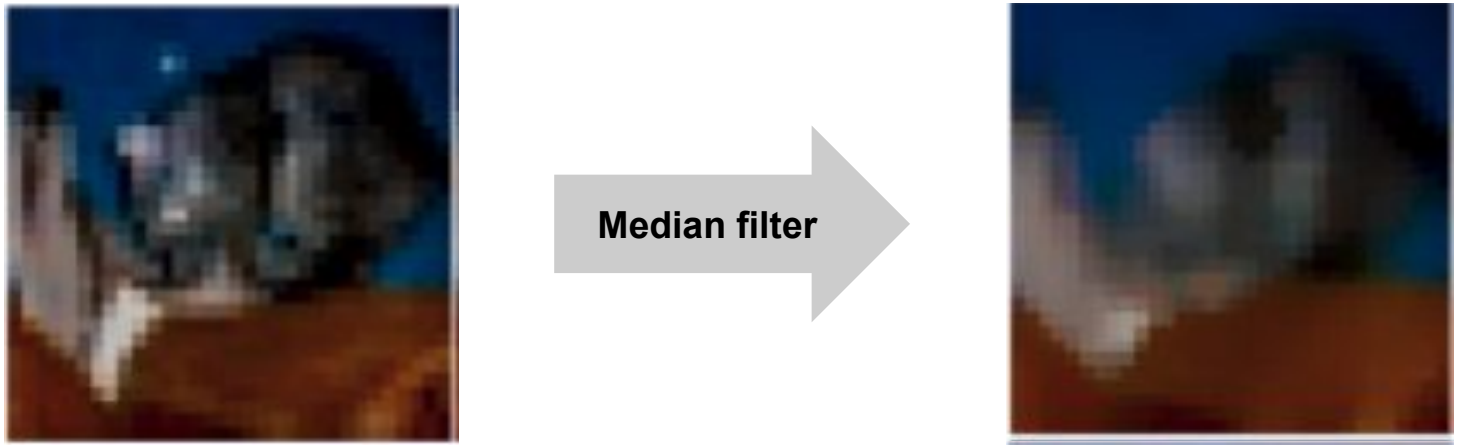
- Makes an image binary by setting pixel intensities to 255 if above certain threshold and to 0 if below certain threshold.
- Controllable parameters
  - Threshold value

Gaussian Filter

- Convolves an image with Gaussian kernel, has blurring effect.
- Controllable parameters
  - Sigma - larger results in more blurring
  - Kernel size - larger results in more pixels in convolution

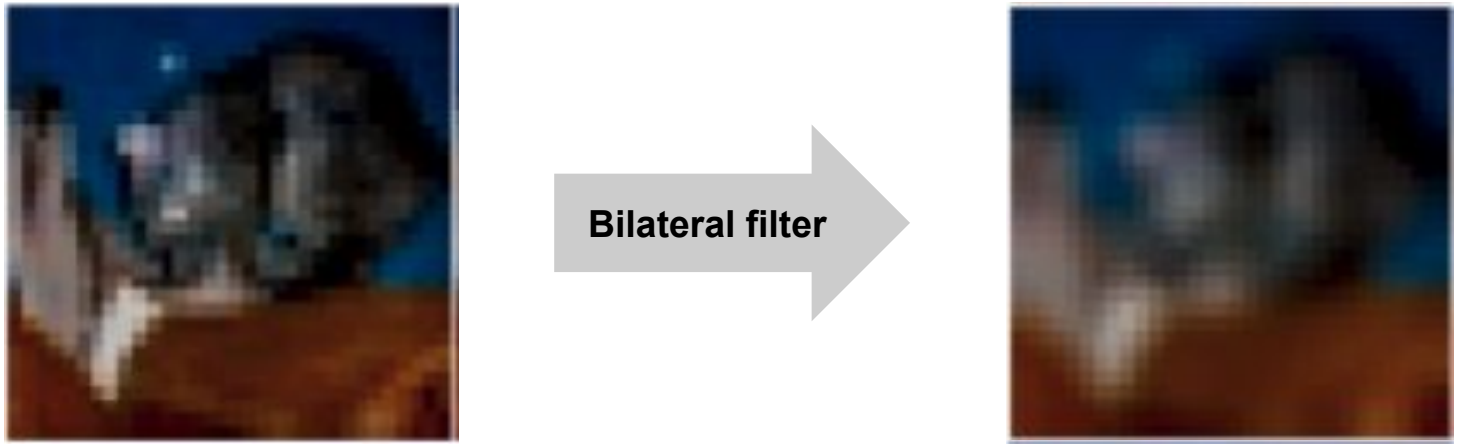
Median Filter

- Locates spikes of intensity in image, replaces with median of pixels surrounding spike
- Controllable parameters
  - Window size - pixels considered in averaging



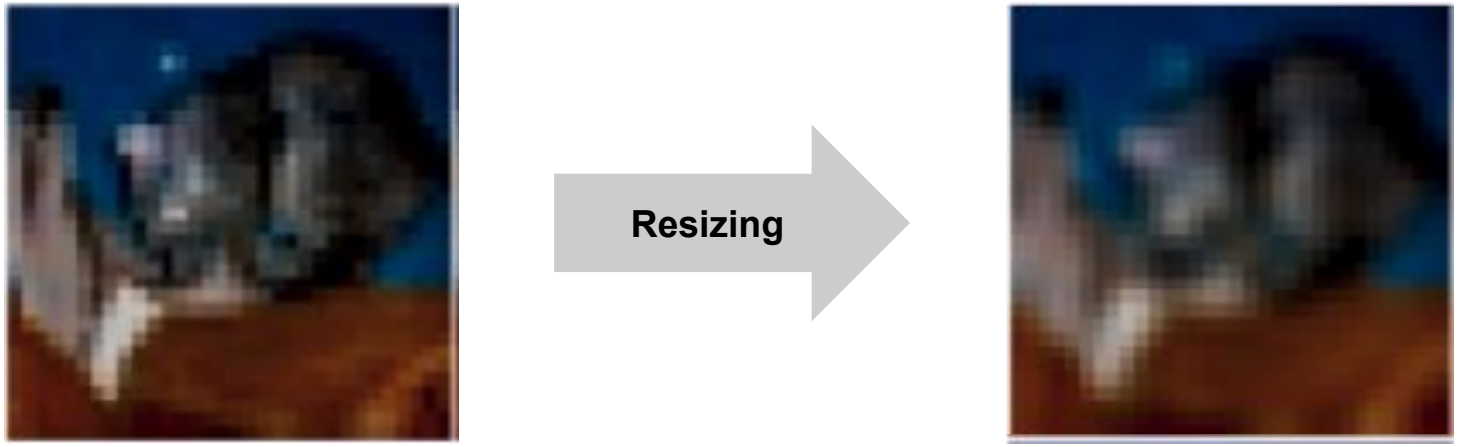
Bilateral Filter

- Replaces intensity of each pixel with weighted average from pixel neighborhood, is edge preserving
- Controllable parameters
  - Window size - pixels considered in averaging
  - Range kernel, domain kernel



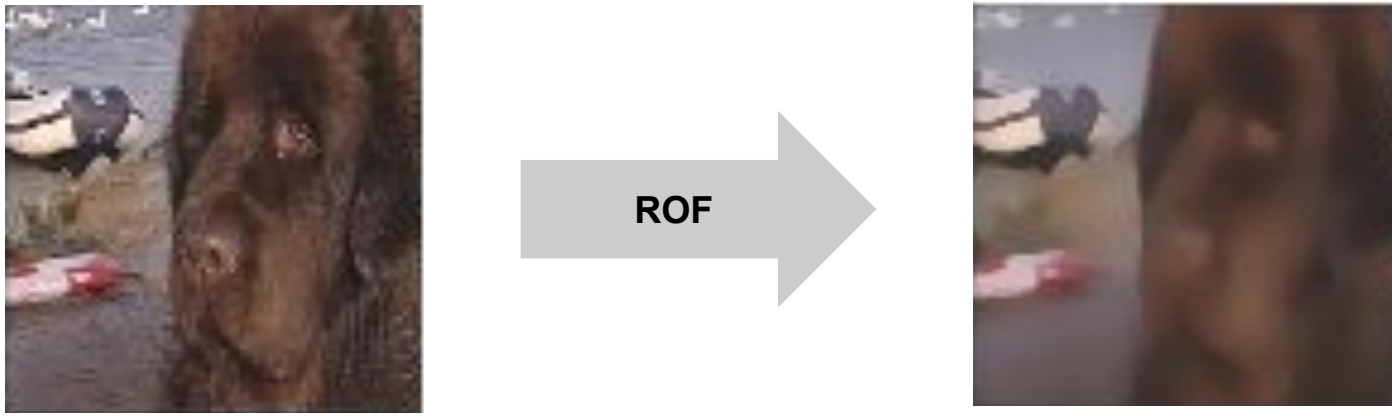
Resizing

- Downsamples image, averages pixels to create smaller image
- Upsamples image, interpolates “missing” values
- Controllable parameters
  - Size



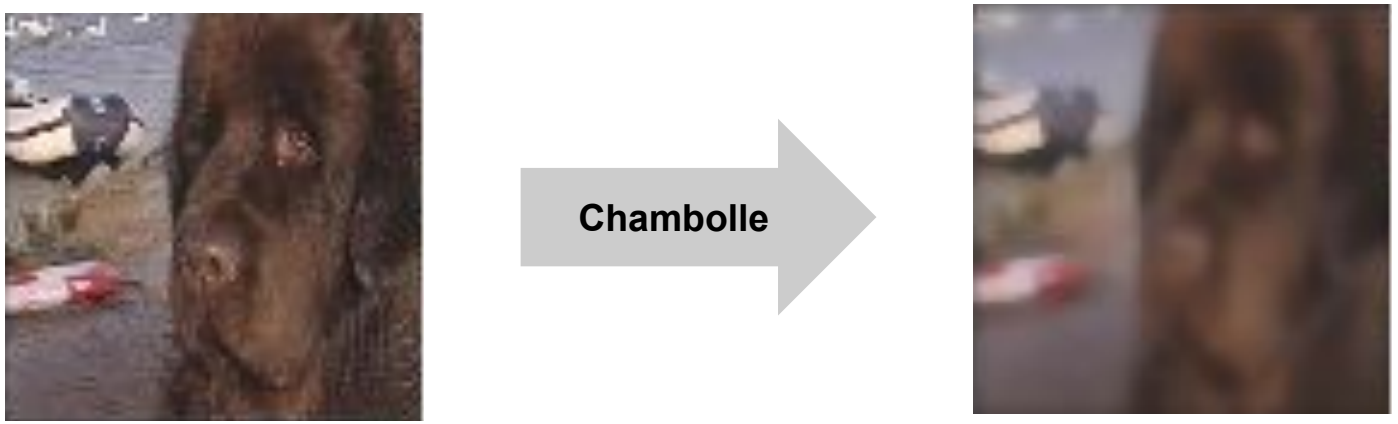
Total Variation Denoising - ROF

- Original TVD, seeks to minimize total variation norm via constrained optimization problem.
- Attempts to recover original image from noisy image while staying within bounded variation.
- Controllable parameters
  - Gamma or weight - larger makes image more “cartoonish”



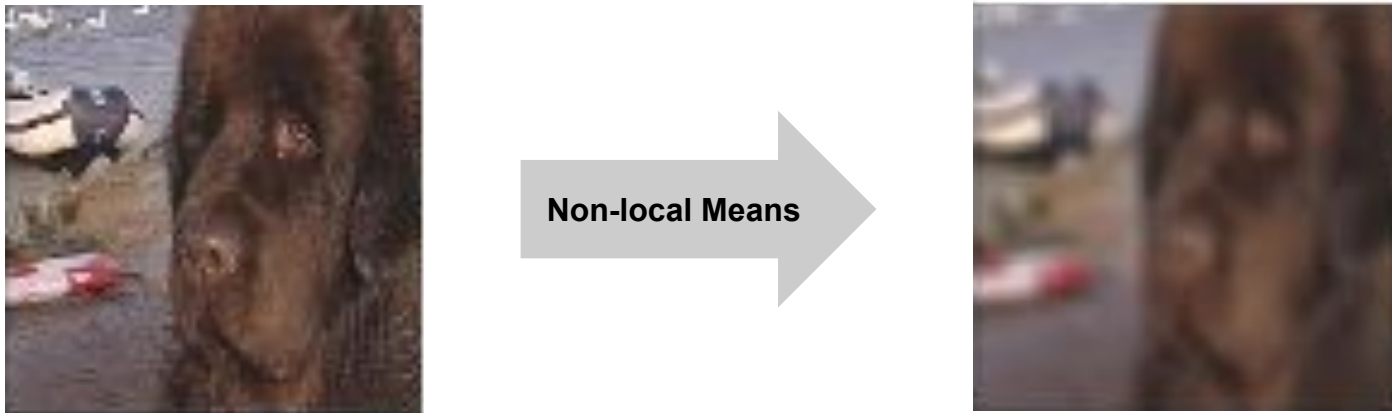
Total Variation Denoising - Chambolle

- Improved TVD, solves minimization using dual formation.
- Results in less “cartoonish” image.
- Controllable parameters
  - Gamma or weight - larger makes image more “cartoonish”



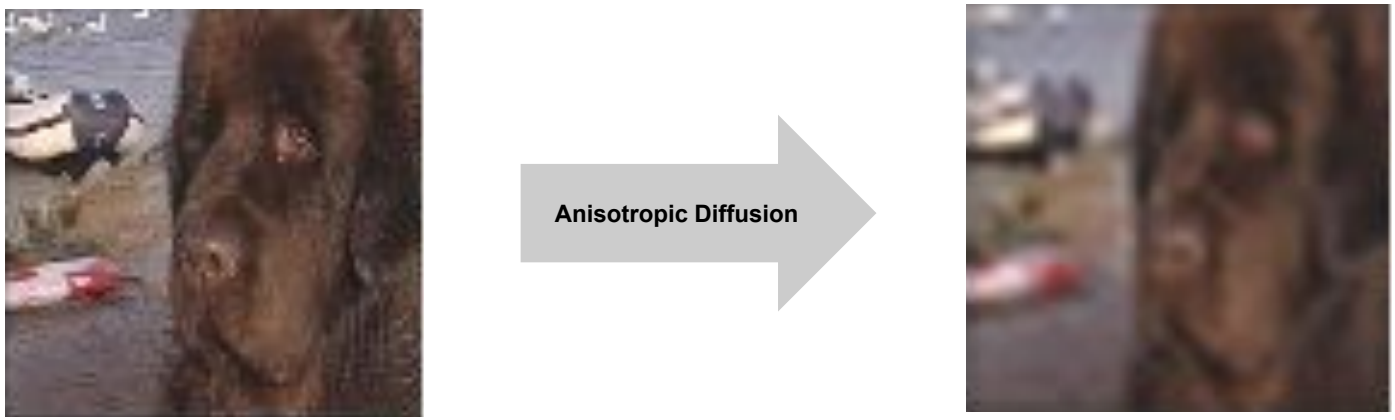
Non-local Means

- Searches entire image for similar patches to replace noisy pixels with, replaces with average of the similar patch.
- Controllable parameters
  - Sigma - expected noise variance
  - H - patch weight decay (larger = blurrier)



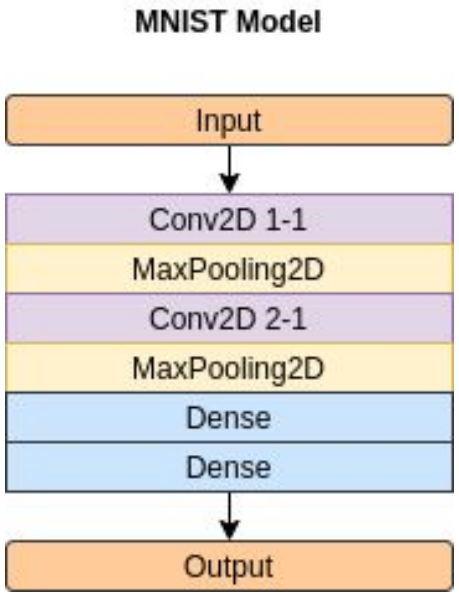
Anisotropic Diffusion

- Convolves image orthogonal to edge only (edge preserving)
- Controllable parameters
  - K - edge sensitivity (smaller = more edges preserved)
  - Num iterations

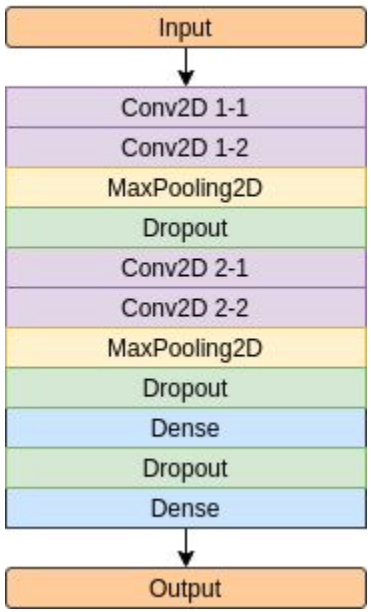


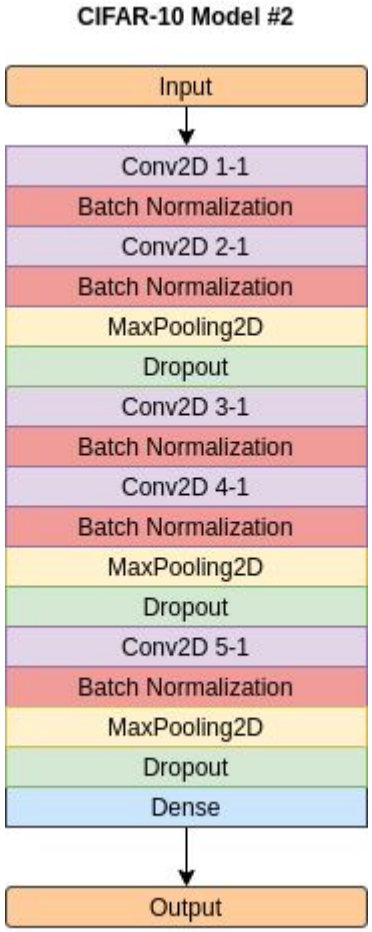
Network Details

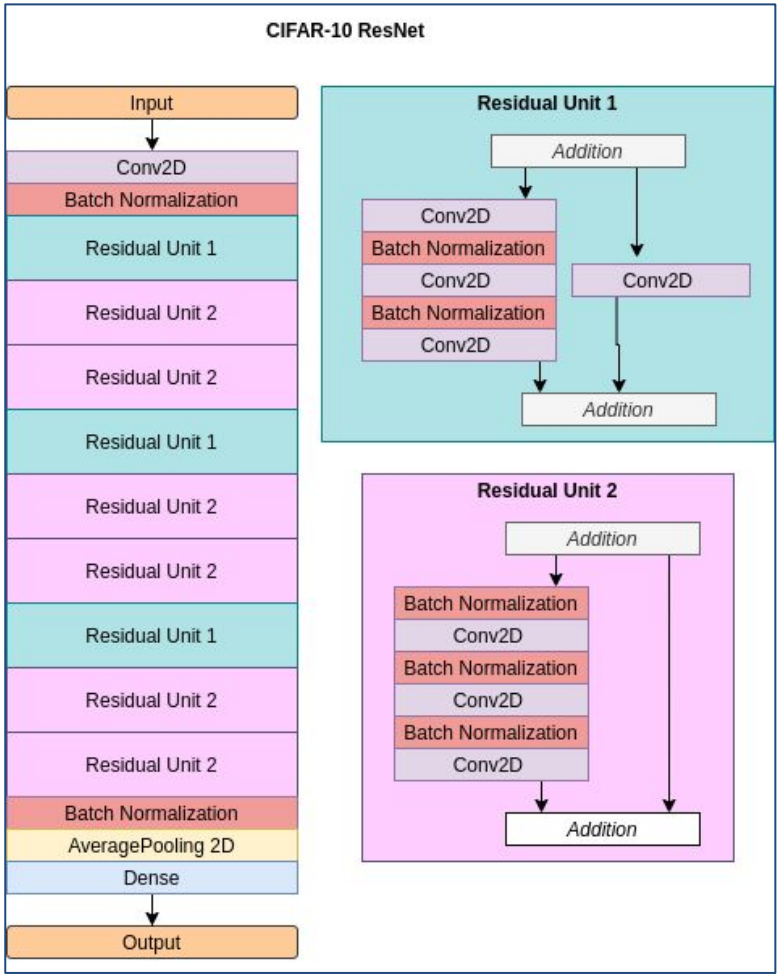
MNIST Model



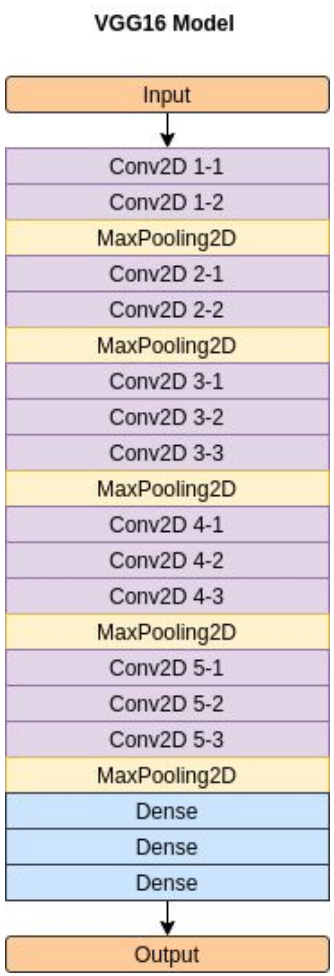
CIFAR-10 Models

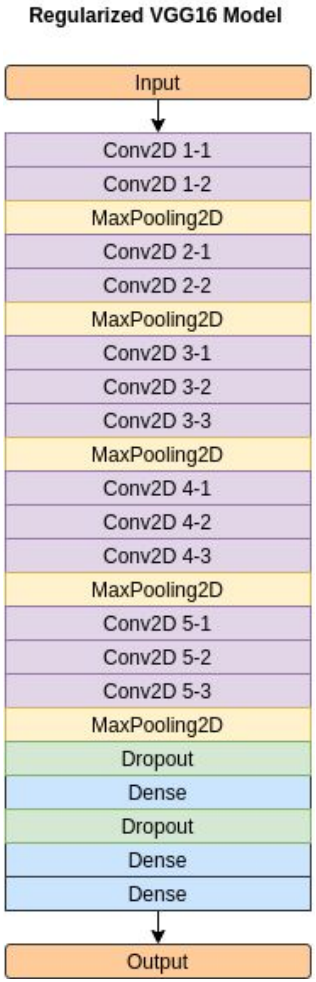






CINIC-10 Large Models

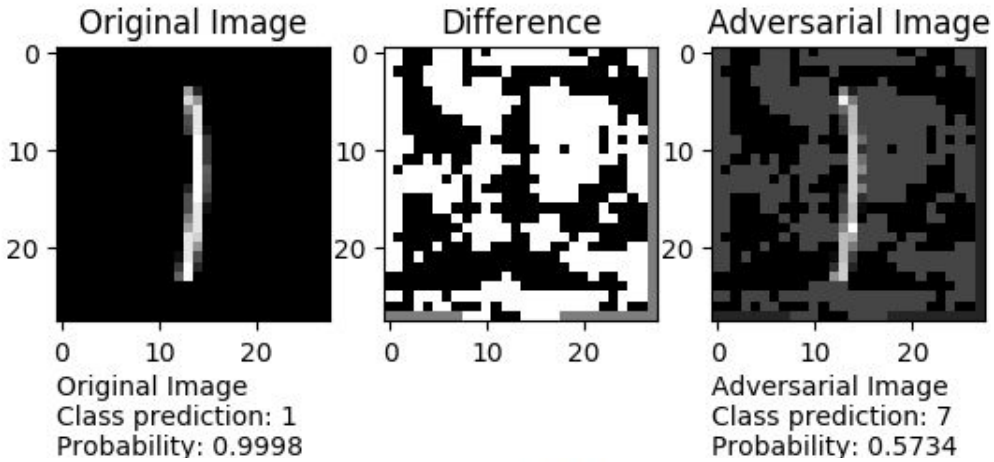


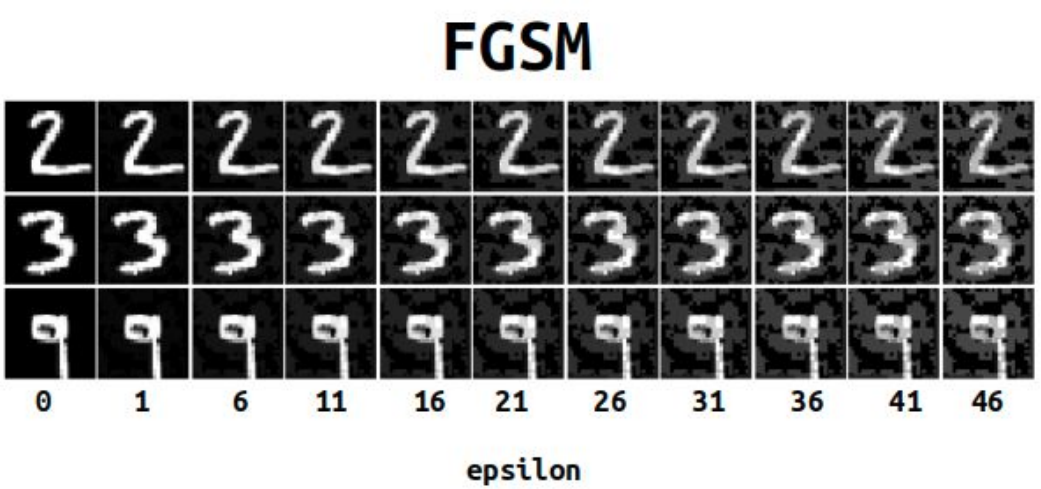


Extra Attack Details

MNIST

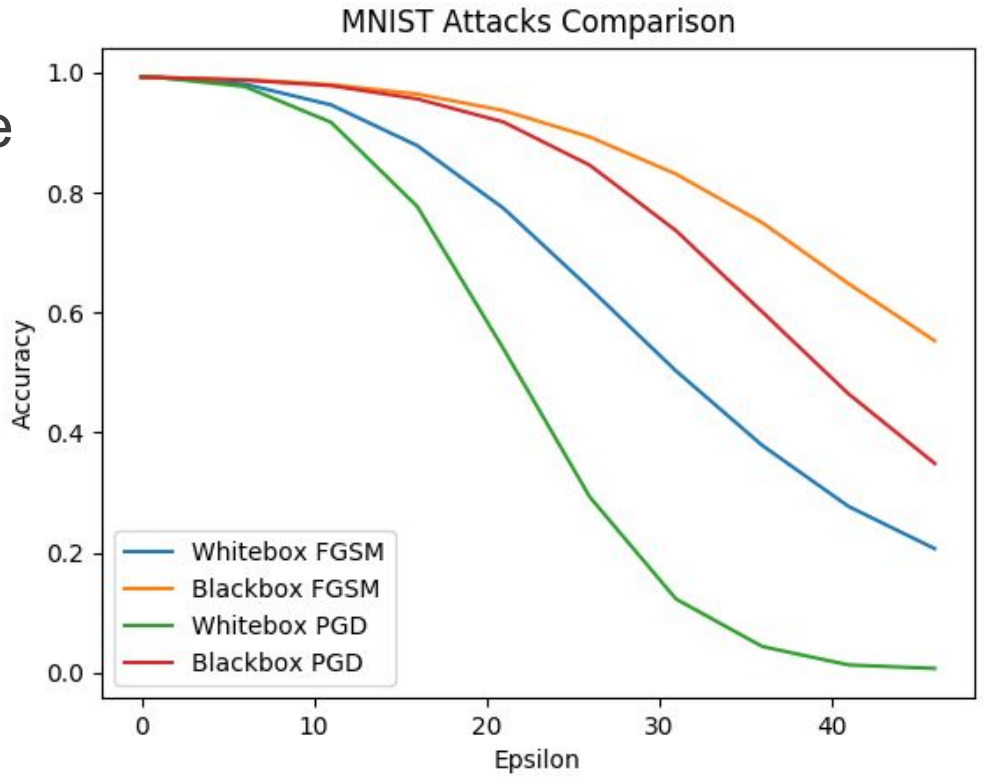
MNIST FGSM



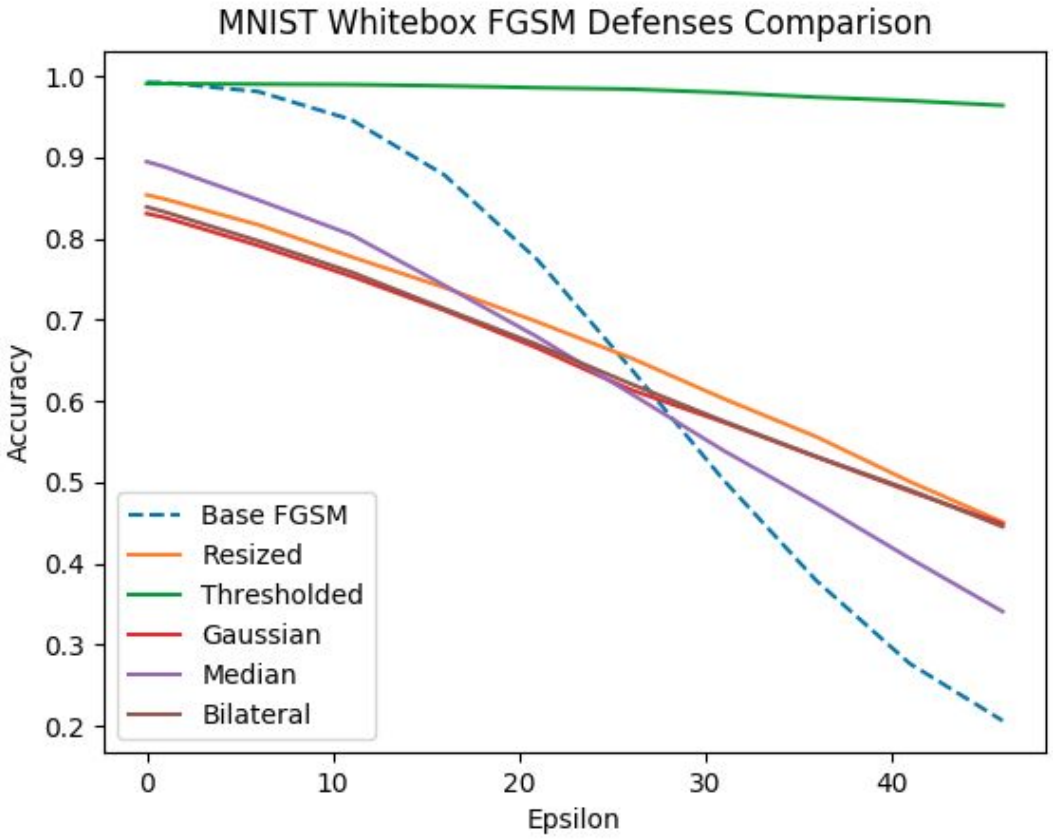


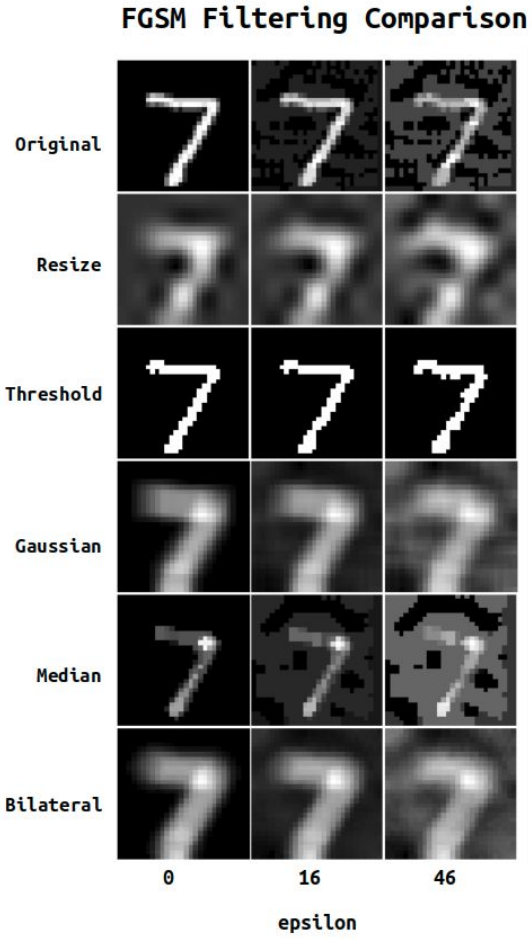
MNIST Attack Summary

- White-box PGD is the most powerful attack.
- Black-box FGSM is the weakest attack.
- These attacks have an initial plateau before accuracy decreases, likely because of MNIST's simplicity.

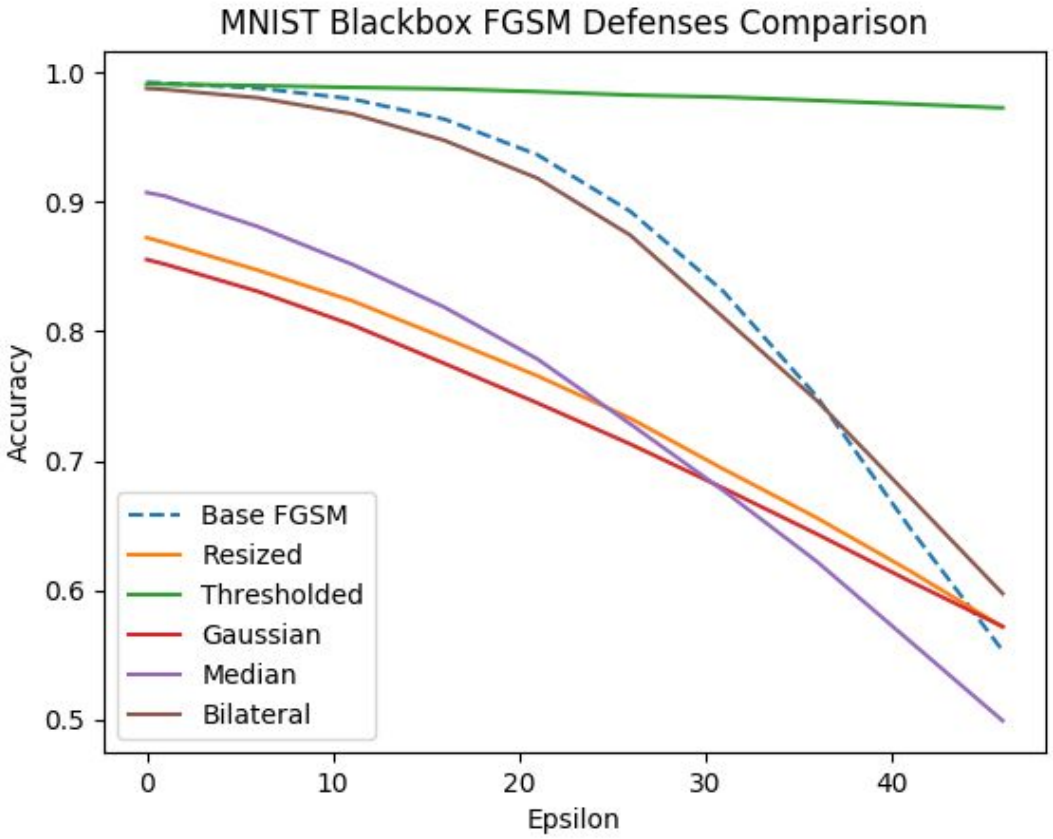


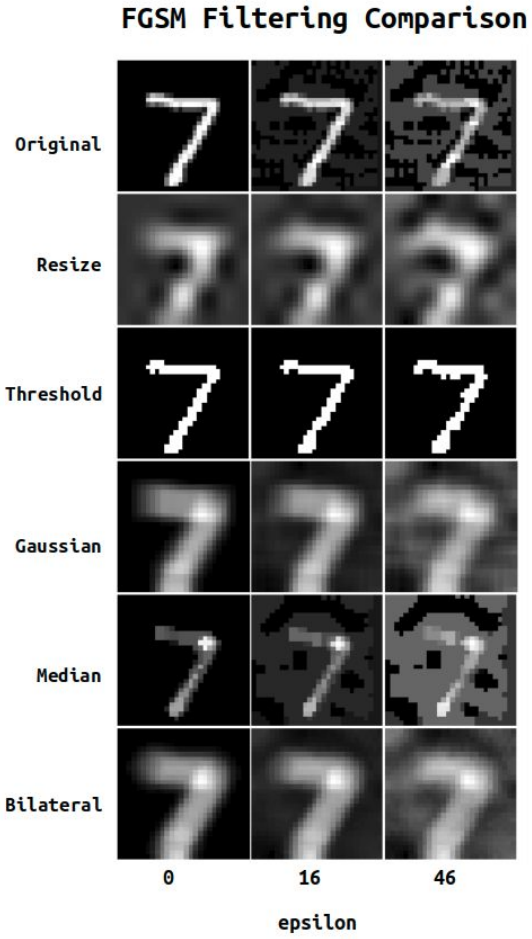
MNIST White-box FGSM Defenses



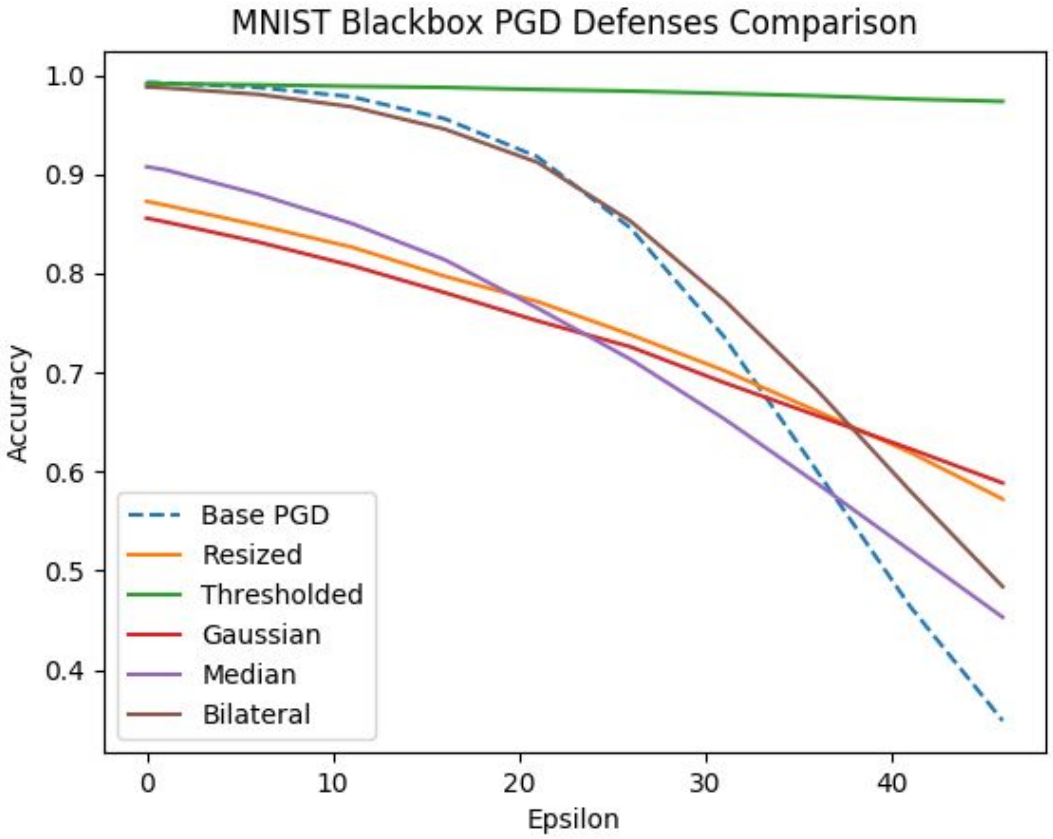


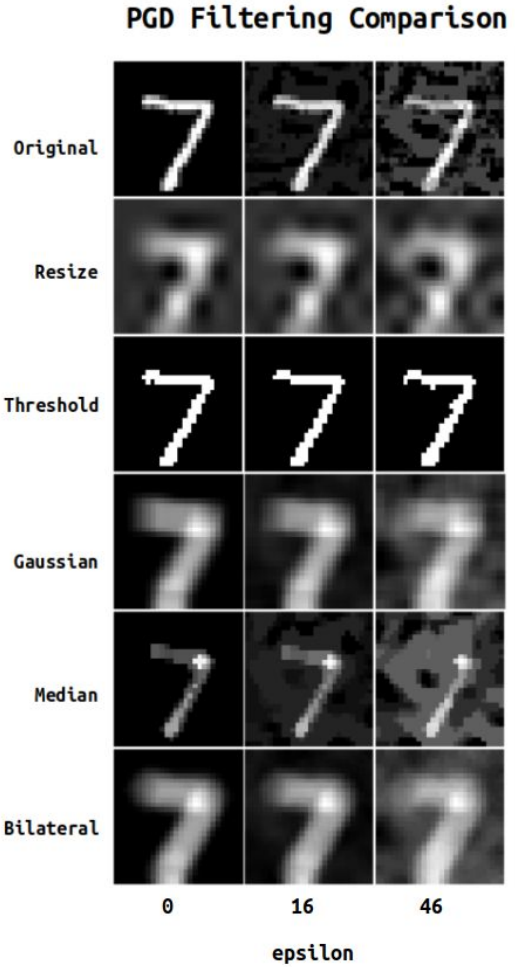
MNIST Black-box FGSM Defenses





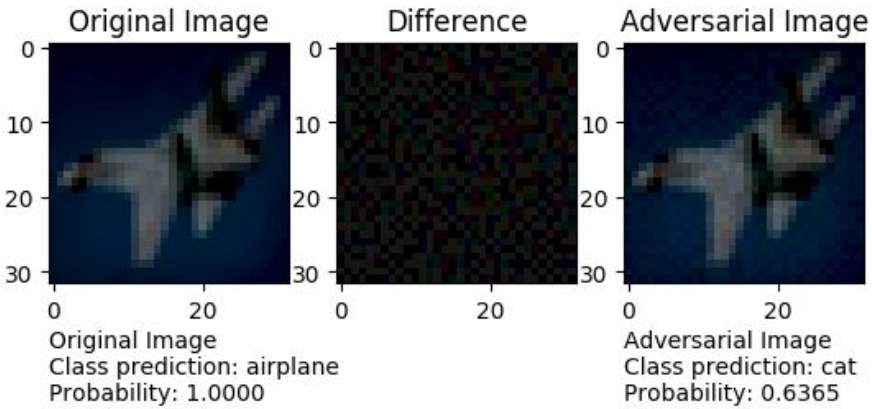
MNIST Black-box PGD Defenses

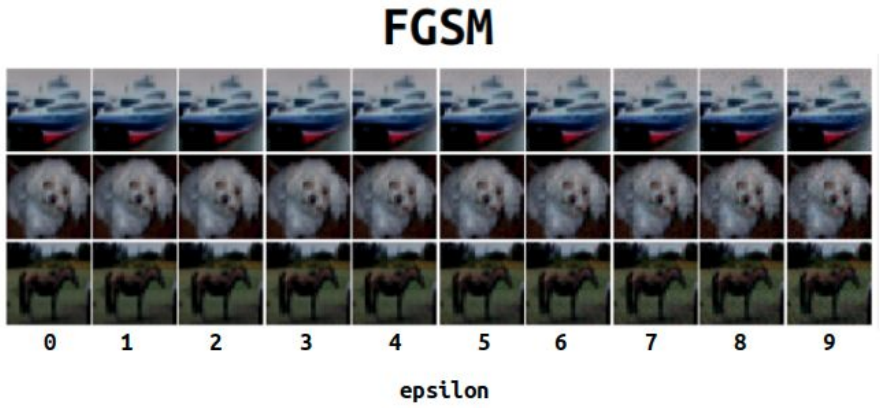




CIFAR-10

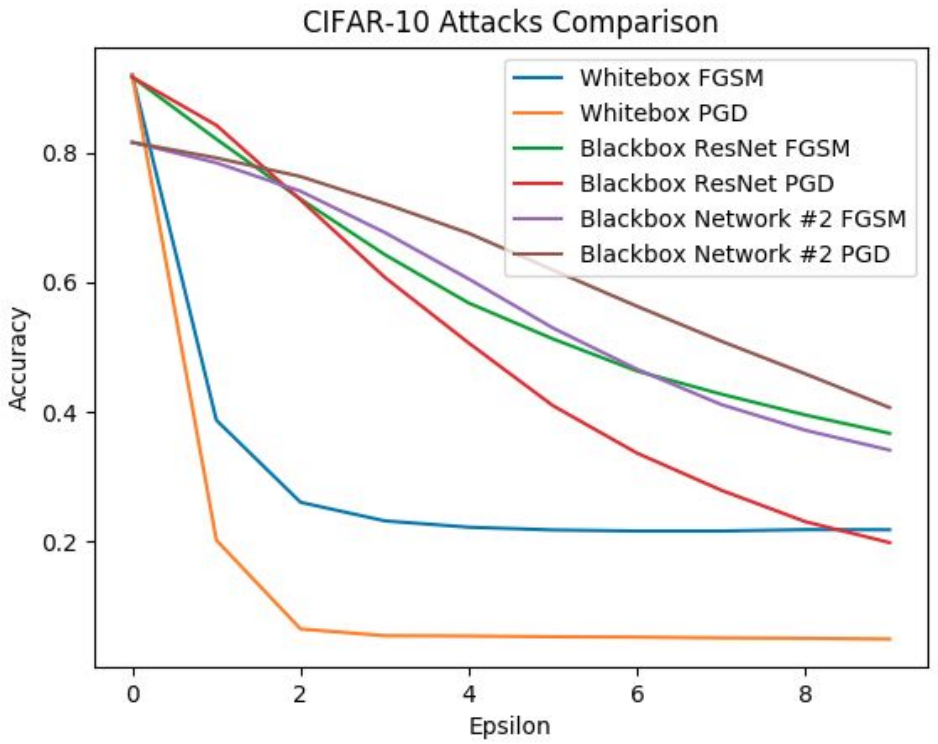
CIFAR-10 FGSM



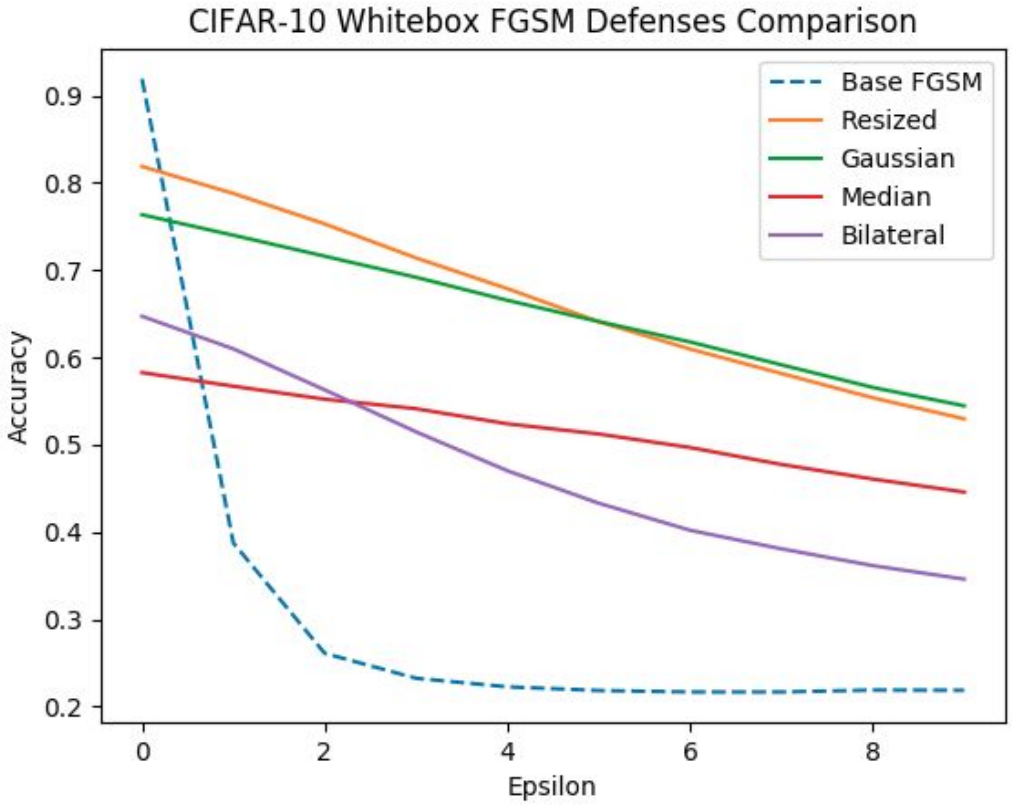


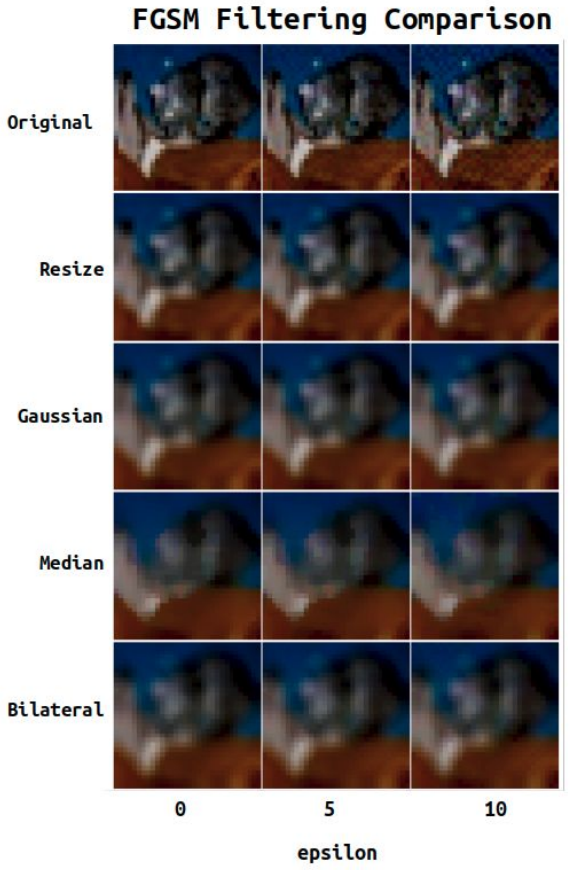
CIFAR-10 Attack Summary

- Two black-box networks are attacked here to test how a model with different architecture fares.
  - *Blackbox Network #2* is a network with slightly lower accuracy.
- The different architecture was not as affected.

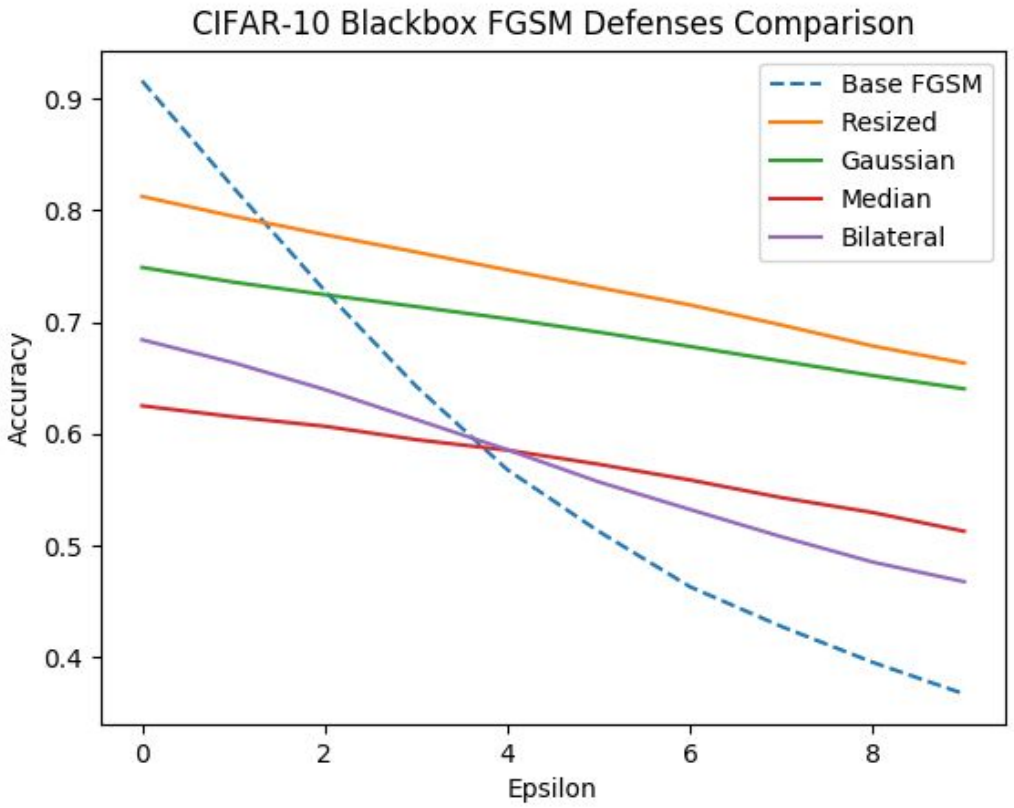


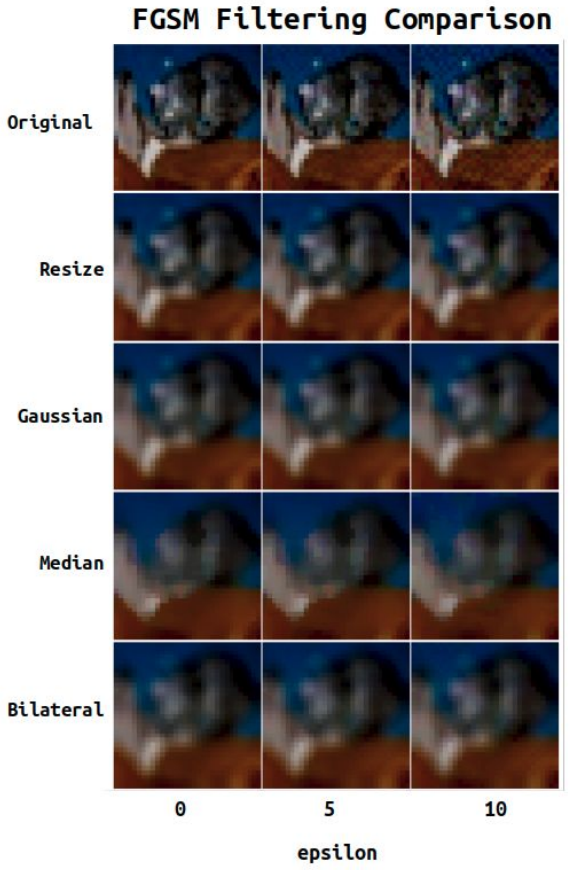
CIFAR-10 FGSM Defenses



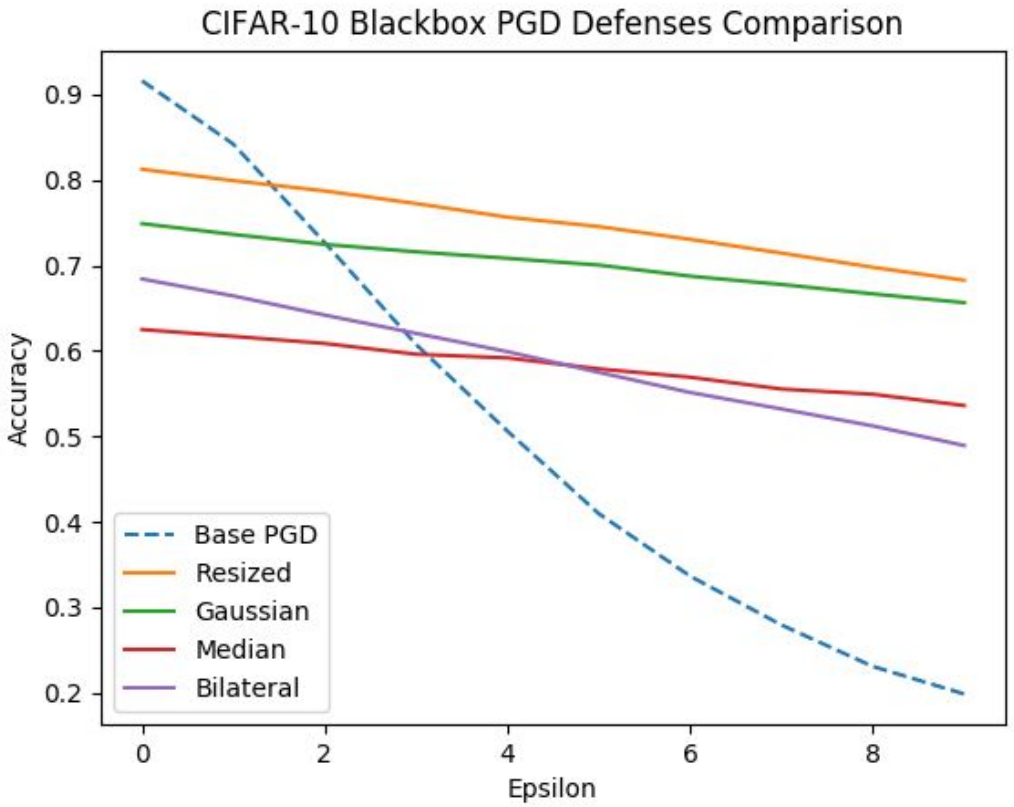


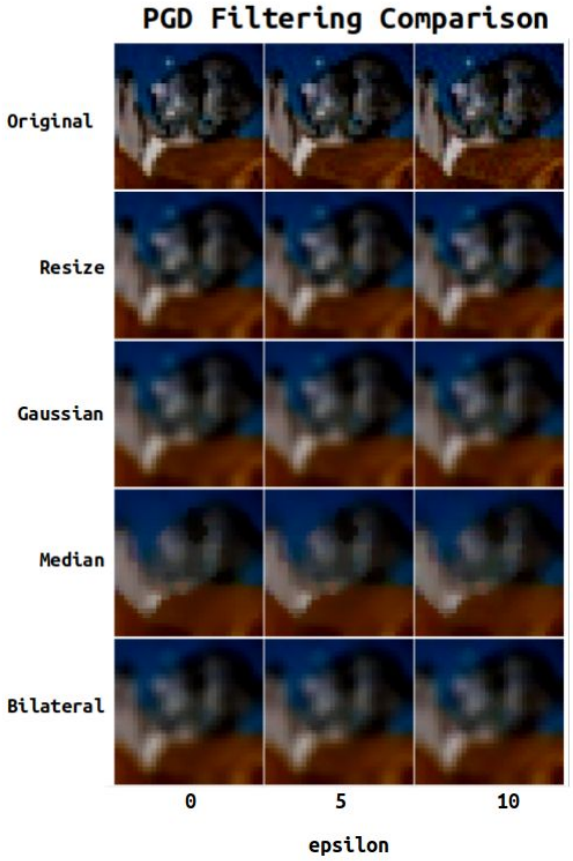
CIFAR-10 FGSM Defenses





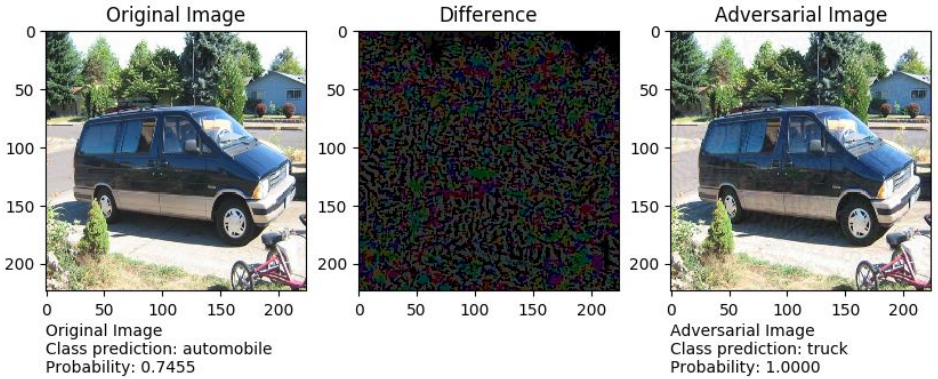
CIFAR-10 PGD Defenses

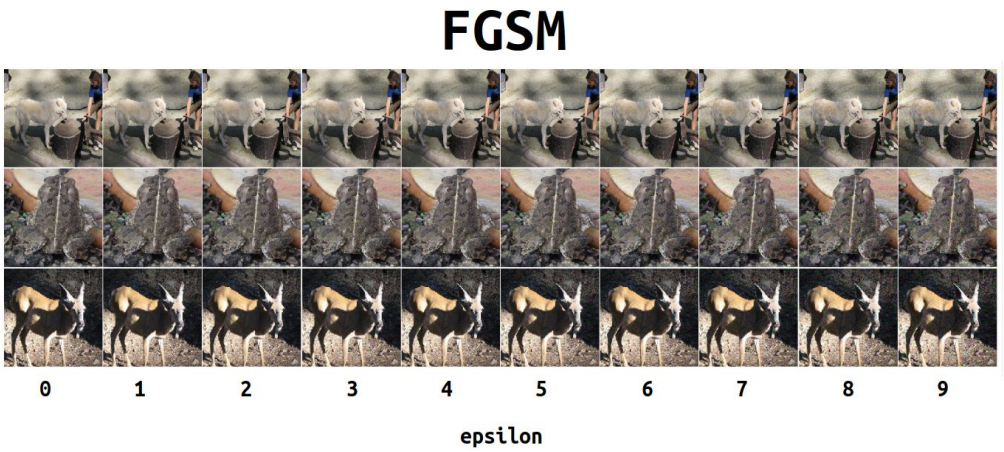




CINIC-10 Large

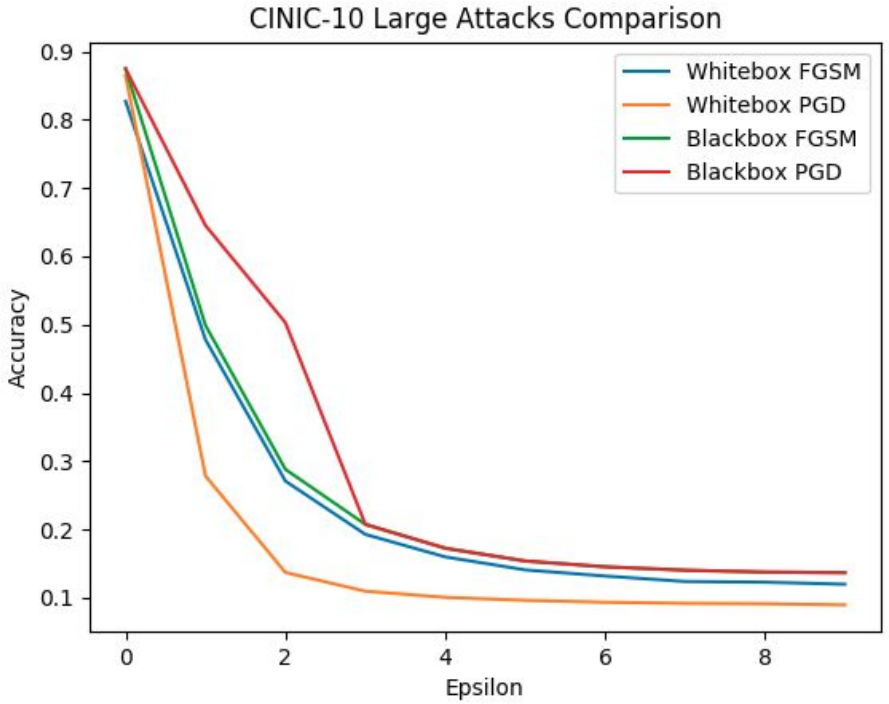
CINIC-10 Large FGSM



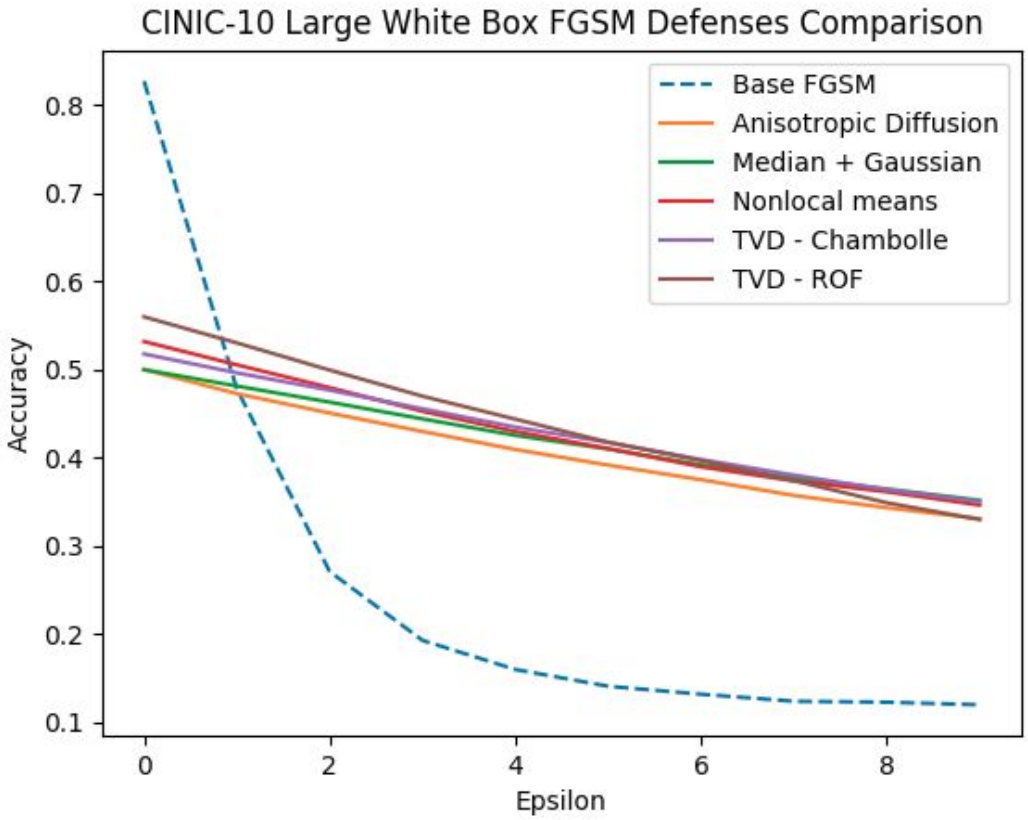


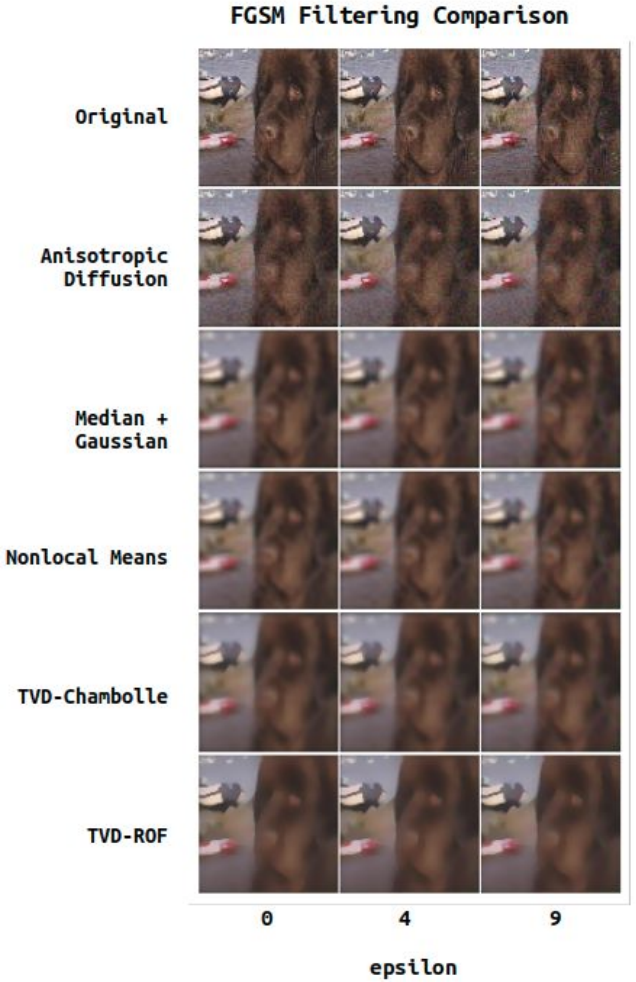
CINIC-10 Large Attack Summary

- White-box PGD is the most powerful attack.
- Classifier performance on white-box PGD was used to perform parameter selection for filtering defenses.

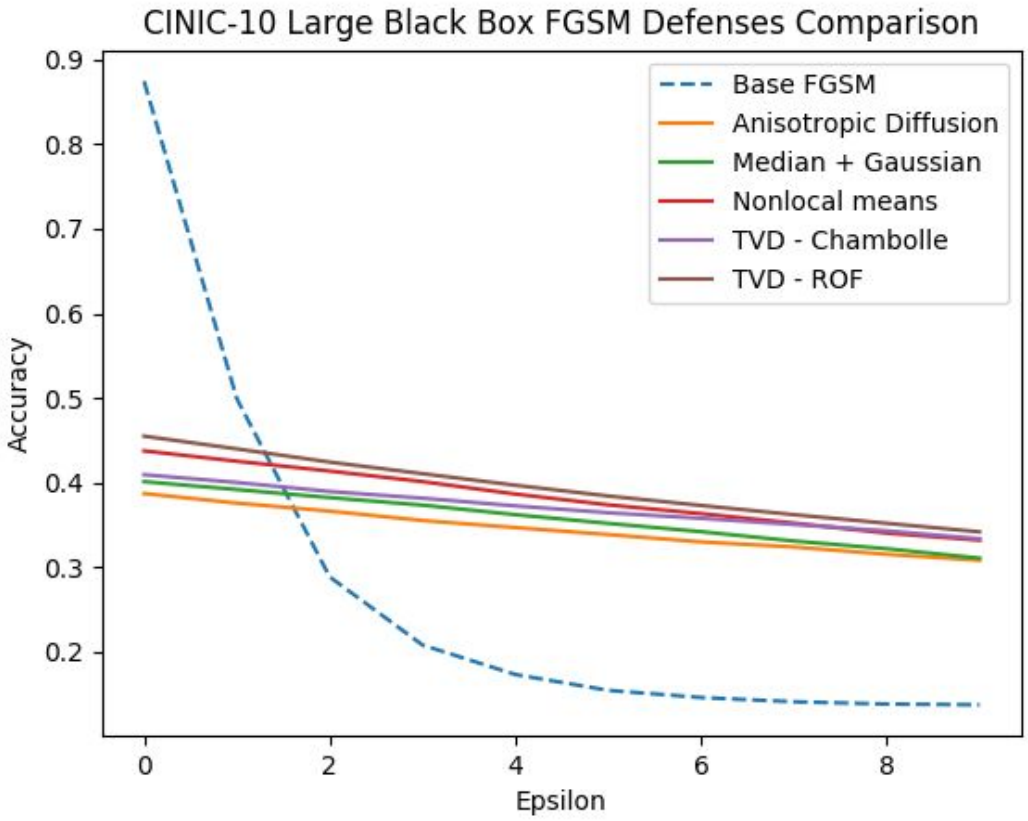


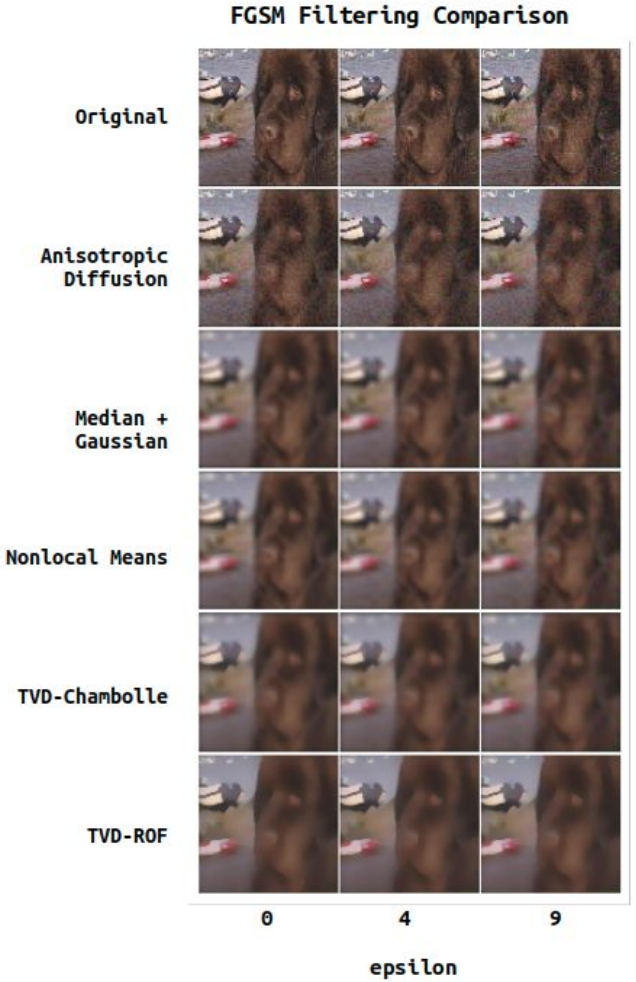
CINIC-10 Large FGSM Defenses



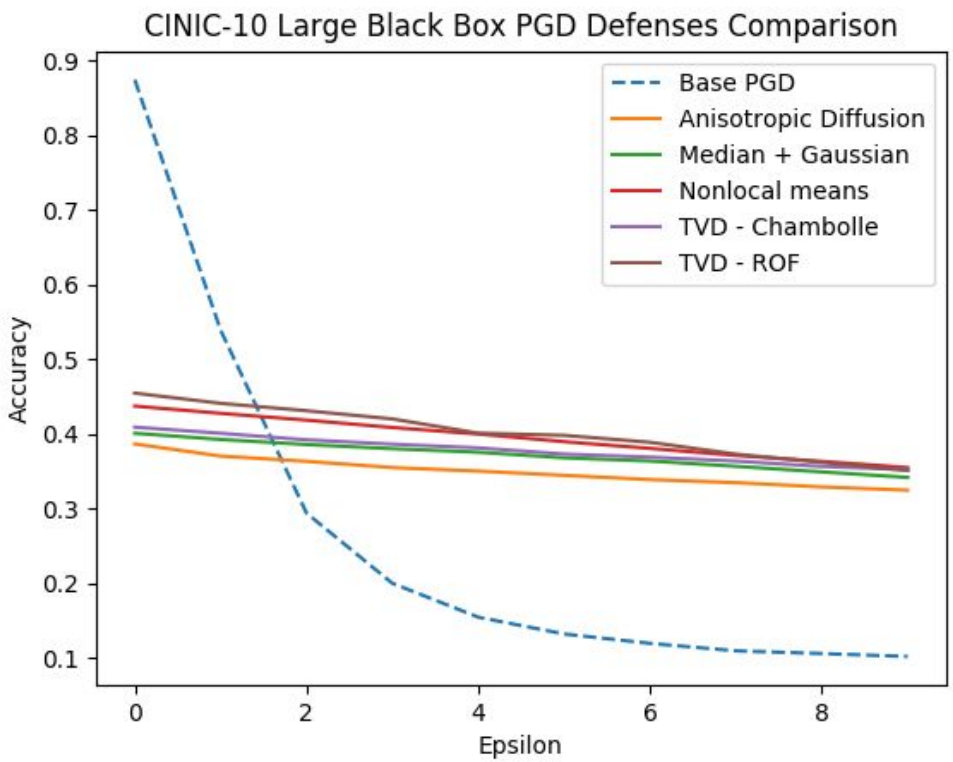


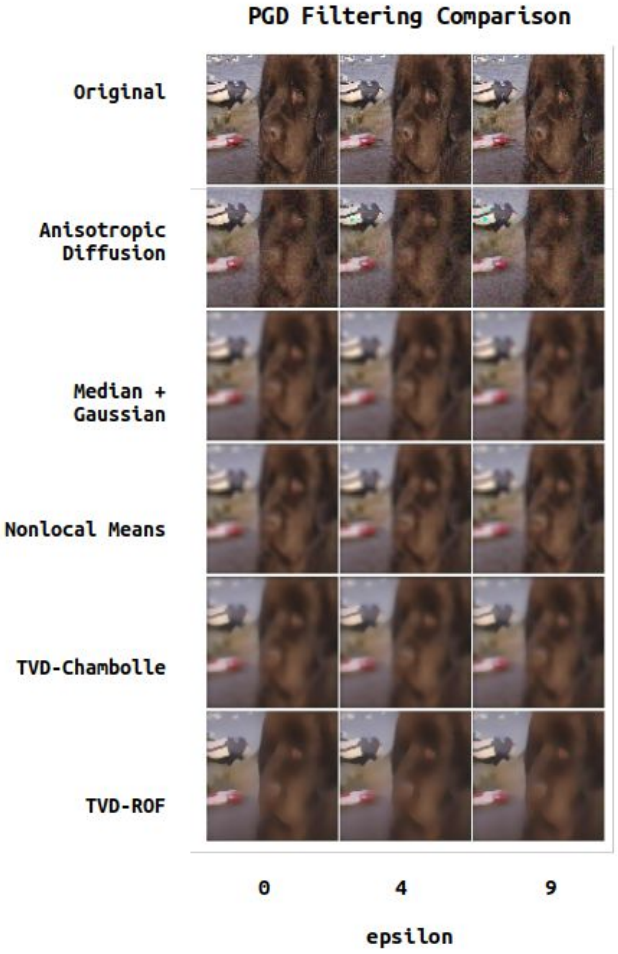
CINIC-10 Large FGSM Defenses



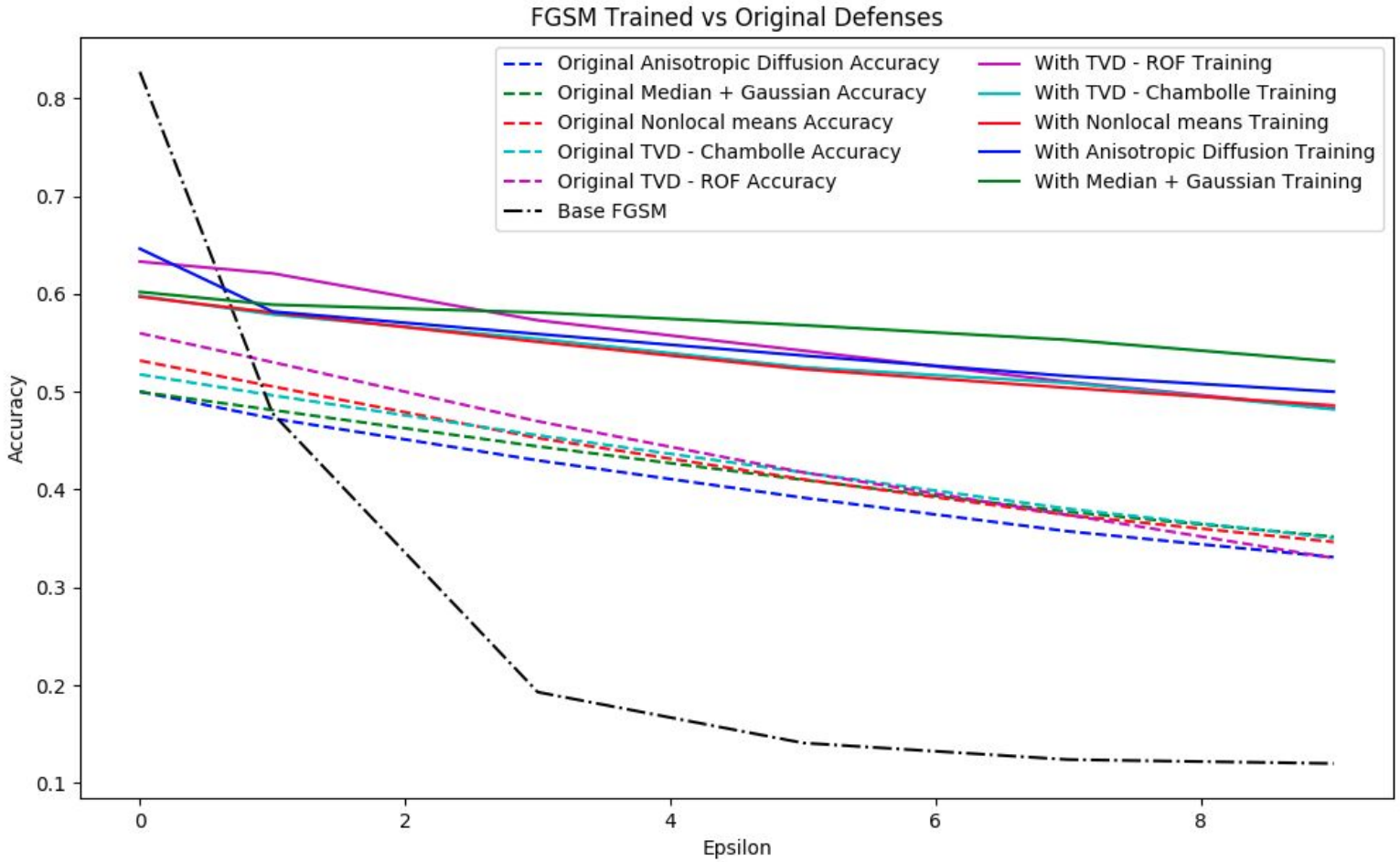


CINIC-10 Large PGD Defenses

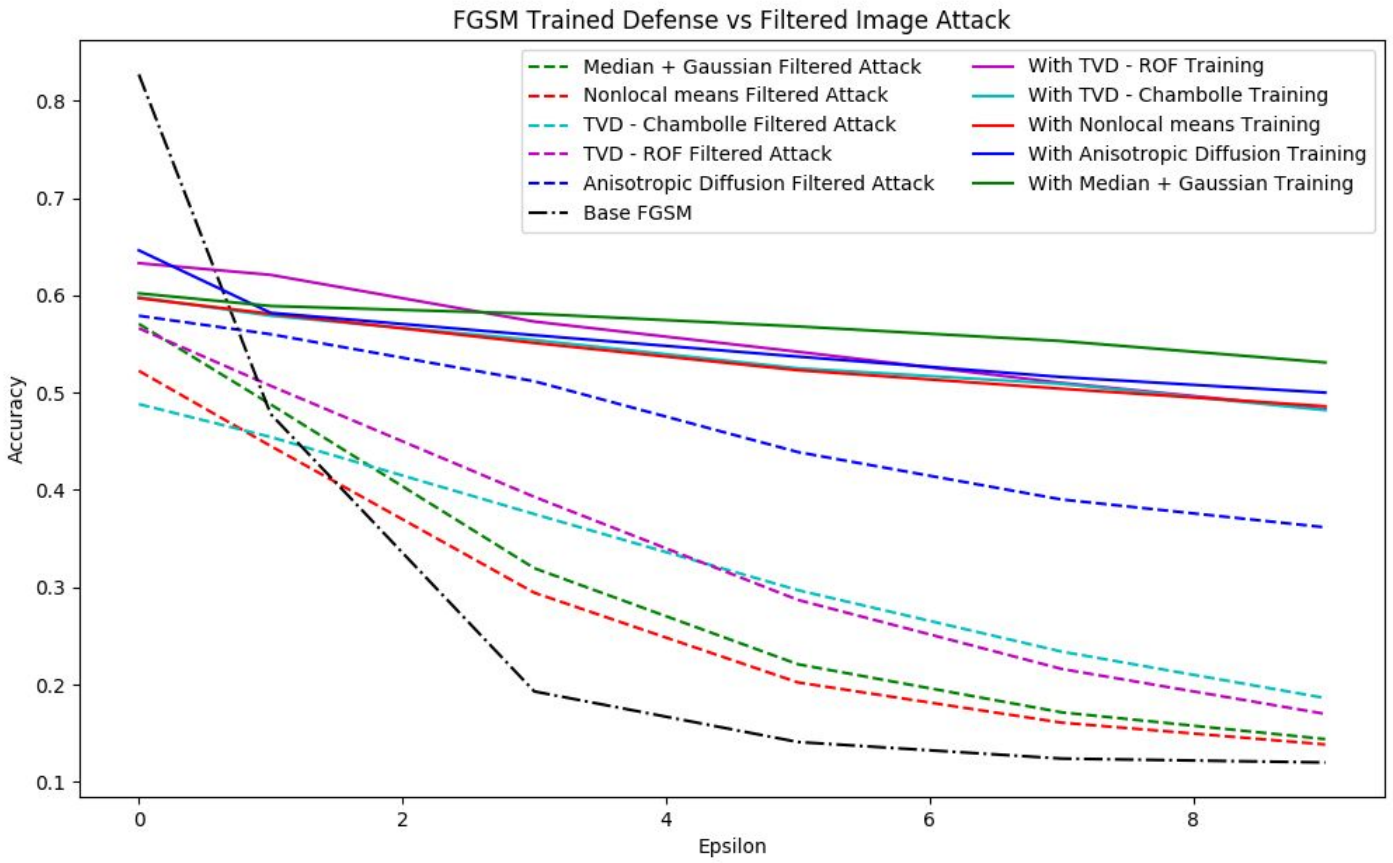




CINIC-10 Large FGSM Gray-box



CINIC-10 Large “True” White-box FGSM



CINIC-10 Large “True” White-box PGD