Adversarial Defense For Computer Vision

William Lu
Joseph Roth

Introduction

In 2012, the computer vision landscape was forever changed when the AlexNet convolutional neural network (CNN) architecture won a decisive victory in the ImageNet classification competition. Since then, CNNs have overtaken hand-engineered feature extractors as the dominant paradigm for image classification, object detection, and semantic segmentation tasks. Over the next few years, the ResNet[1][2], Faster R-CNN[3][4], and RefineNet[5] architectures rose to prominence as state-of-the-art solutions for these respective tasks. Today, CNNs are used in a wide range of applications, from self-driving cars to automated analysis of medical imagery.

Due to their increasing prevalence in mission-critical applications, CNNs must remain robust and reliable under challenging input conditions. Unfortunately, CNNs are notoriously susceptible to adversarial examples, which are images crafted to look normal to humans while tricking CNNs into producing incorrect predictions with high confidence. As a result, adversarial defense has become a popular research topic, aiming to discover CNN architectures that are more resilient to adversarial examples.

The vast majority of adversarial defense research so far focuses on improving classification models. There is a paucity of published literature dealing with detection and segmentation models. In our project, we aim to close this gap. We examine the Max-Mahalanobis Center (MMC) loss[6], a popular classification defense, and apply it to the Faster-RCNN and RefineNet models. Our code is hosted at https://github.com/googleinterns/out-of-distribution.

Notation

In the Approach and Designing the Adversarial Objective Function For Gaussian Models sections, we use the following notation:

- $\vec{x}$ is the image to perturb
- y is the true class of $\vec{x}$
- $\hat{y}$ is the predicted class of $\vec{x}$
- k is the number of classes
- c is a generic class index; we have $1 \leq c \leq k$
- $\tilde{x}$ is the perturbed image

The notation below refers to various parts of an architecture that uses the MMC loss:

- $\vec{z}$ is the hidden vector output by the architecture's penultimate layer
- $\overrightarrow{\mu_c}$ is the MMC center for class c
- p_c is the prior probability for class c
- $\ell_c(\vec{z})$ is the Gaussian PDF (likelihood) for class c
- $f(\vec{z})$ is the objective function to maximize when generating an adversarial example

[1] https://arxiv.org/abs/1512.03385
[2] https://arxiv.org/abs/1603.05027
[3] https://arxiv.org/abs/1506.01497
[4] https://arxiv.org/abs/1612.03144
[5] https://ieeexplore.ieee.org/document/8618363
[6] https://arxiv.org/abs/1905.10626

Approach

During this internship, we reproduced the results of the MMC paper on several ResNet architectures for image classification. Due to time constraints, we were unable to experiment with applying the MMC loss to Faster R-CNN and RefineNet as we originally planned. Nevertheless, our work so far confirms the MMC loss as a promising adversarial defense mechanism.

Unless otherwise noted, all of our experiments use the CIFAR-10 dataset[7], which consists of 60K images in a 50K/10K train/test split. We further perform a random split on the former 50K images, obtaining a 45K training set and a 5K validation set. Throughout this paper, all reported results are obtained on this 45K/5K split. None of our work so far has used the 10K test images because we are saving them for future use after we complete our exploratory analyses.

All of our experiments use ResNet architectures. In this paper, we refer to each architecture using a naming convention that communicates:

- The number of layers in the architecture
- Whether pre-activated blocks are used (v1 = no, v2 = yes)
- Which loss function (softmax cross-entropy or MMC Gaussian squared distance) is used

For example, *ResNet20-v1-Softmax* refers to the 20-layer architecture without pre-activation and with softmax loss; *ResNet32-v2-Gaussian* refers to the 32-layer architecture with pre-activation and with MMC loss.

As described in the ResNet papers, our architectures consist of a stem, three stages, and a fully-connected layer. Each stage consists of n blocks. The ResNet-v1 architectures use 6-layer residual blocks without pre-activation. The ResNet-v2 architectures use 9-layer bottleneck blocks with pre-activation. In our experiments, we consider $n = 2,4,6,8$, thus giving rise to the architectures in the table below:

Pre-activated blocks?	Loss function	Architectures
No	Softmax cross-entropy	ResNet20-v1-Softmax ResNet32-v1-Softmax ResNet44-v1-Softmax ResNet56-v1-Softmax
Yes	Softmax cross-entropy	ResNet29-v2-Softmax ResNet47-v2-Softmax ResNet65-v2-Softmax ResNet83-v2-Softmax
Yes	MMC Gaussian squared distance	ResNet29-v2-Gaussian ResNet47-v2-Gaussian ResNet65-v2-Gaussian ResNet83-v2-Gaussian

We find that ResNet-v1-Gaussian architectures fail to converge during training, so we do not investigate them further in our paper. The authors of the MMC paper observed this phenomenon as well. We hypothesize that ResNet-v1 fails to converge because its ReLU shortcuts are a fundamental design flaw that defeats the purpose of shortcuts in the first place (by contrast, ResNet-v2 uses identity shortcuts) and the MMC loss exacerbates this flaw.

[7] https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

We train our ResNets on CIFAR-10 for 200 epochs, using a batch size of 128. We use a learning rate of 0.1 for the softmax architectures and 0.01 for the Gaussian architectures. We decrease the learning rate by 10x after 100 and 150 epochs. We use non-Nesterov momentum with a factor of 0.9. We use weight decay with a factor of 10^{-4} on weight parameters, but not on biases or batch-normalization parameters. After training completes, we benchmark the training and validation accuracy achieved on the original (unperturbed) CIFAR-10 images.

Next, we conduct experiments to investigate whether the MMC loss is an effective out-of-distribution (OOD) detector. We experiment with SVHN[8] and MNIST[9] as the OOD datasets. SVHN consists of ≈630K images in a 73K/26K/531K train/test/extra split; we randomly choose 20% of the train/extra images as our validation set. MNIST consists of 60K images in a 50K/10K train/test split; we randomly choose 10K of the train images as our validation set. Using the ResNet83-v2-Gaussian model trained on CIFAR-10, we perform inference on these validation sets. The MMC layer outputs class-wise log-likelihoods for each validation image. We construct a *minimum histogram* containing the minimum class-wise log-likelihood for each image. We also construct an analogous *maximum histogram*. If the MMC loss is an effective OOD detector, the minimum histograms for the in-distribution and OOD datasets should have minimal overlap. This allows simple thresholding to separate most OOD inputs from in-distribution inputs. Analogous claims can be made for the maximum histograms.

We investigate the accuracy achieved by the ResNet56-v1-Softmax, ResNet83-v2-Softmax, and ResNet83-v2-Gaussian models against adversarial attacks on the CIFAR-10 validation set. We do not perform adversarial training. Since untargeted attacks are more difficult to defend against than targeted attacks, we only consider untargeted attacks in this paper. The table below lists the attack algorithms we experimented with. The white-box algorithms are typical first-order adversaries that would likely be used by an attacker with complete access to our models' architectural details. The black-box algorithms serve as a baseline to ensure that any robustness we observe is not caused by gradient masking.

Type	Algorithm	Hyperparameters
White-box	Fast gradient sign method[10]	(none)
	Iterative FGSM[11]	• Step size: 1/255
	Momentum FGSM[12]	• # of iterations: 10 • Decay factor: 1.0
	Projected gradient descent[13]	• # of iterations: 50 • Step size: 2/255
Black-box	Random sampling[14]	• # of samples: 200
	Momentum FGSM transfer, where adversarial examples generated on ResNet83-v2-Softmax are used to attack ResNet83-v2-Gaussian	• # of iterations: 10 • Decay factor: 1.0
	Simultaneous perturbation stochastic approximation[15]	• Perturbation size: 0.01 • Step size: 0.01

[8] http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
[9] http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf
[10] https://arxiv.org/abs/1412.6572
[11] https://arxiv.org/abs/1607.02533
[12] https://arxiv.org/abs/1710.06081
[13] https://arxiv.org/abs/1706.06083
[14] https://blog.acolyer.org/2018/08/15/obfuscated-gradients-give-a-false-sense-of-security-circumventing-defenses-to-adversarial-examples/
[15] https://arxiv.org/abs/1802.05666

		• Batch size: 128 • # of iterations: 10
	Carlini and Wagner[16]	• # of binary search steps: 9 • # of Adam iterations per binary search step: 1000 • Initial c: 0.01 • Learning rate: 0.005

We also run projected gradient descent with 10, 100, and 200 iterations to estimate the number of iterations necessary for PGD to converge.

Let ϵ be the upper bound on the perturbation applied to the Red, Green, or Blue value of any pixel; formally, $\|\tilde{x} - \vec{x}\|_\infty < \epsilon$. We run each attack algorithm with $\epsilon = \frac{1}{255}, \frac{2}{255}, \frac{4}{255}, \frac{8}{255}$ (except Carlini and Wagner, which does not take ϵ as a hyperparameter.)

All of the attack algorithms in the table above generate adversarial examples by perturbing the given images to maximize an objective function. We carefully design this objective function so the gradients do not vanish, ensuring we are experimenting on strong attackers. Details for the objective used with the Gaussian models are presented in the next section, Designing the Adversarial Objective Function For Gaussian Models.

We examine whether the MMC loss scales up to large, multi-class datasets such as ImageNet. The MMC paper does not address this question, but leaves it open to future work. We train ResNet101-v2-Softmax, ResNet101-v2-Gaussian, and ResNet152-v2-Gaussian on ImageNet, visualizing the training loss, training and test accuracy, and first-order filter weights as the training progresses. Note that these are 4-stage ResNet architectures. We use a batch size of 64 due to GPU memory constraints. We use a learning rate of $2.5 * 10^{-2}$ for the softmax architecture and $2.5 * 10^{-3}$ for the Gaussian architectures. We decrease the learning rate by 10x after 30 and 60 epochs. We use non-Nesterov momentum with a factor of 0.9. We use weight decay with a factor of 10^{-4} for the softmax architecture and 10^{-6} for the Gaussian architectures. Weight decay is only applied to weight parameters, and not on biases or batch-normalization parameters.

Finally, we plot the angle and distance between pairs of MMC centers as a function of k and the dimensionality of $\vec{z}$. If the angle and distance do not vary significantly when k is increased from 10 to 1000, we intuitively expect similar qualitative observations on CIFAR-10 and ImageNet.

All code is written in Python and PyTorch[17]. All figures are generated using Seaborn[18].

Designing the Adversarial Objective Function For Gaussian Models

Attempt 1:

At test time, ResNet-Gaussian classifies $\vec{x}$ into the class with the highest posterior probability:

$$\hat{y} = \arg\max_c \frac{\ell_c(\vec{z}) \cdot p_c}{\sum_{c'=1}^{k} \ell_{c'}(\vec{z}) \cdot p_{c'}}$$

[16] https://arxiv.org/abs/1608.04644
[17] https://arxiv.org/abs/1912.01703
[18] https://zenodo.org/record/4019146

In accordance with the MMC paper, we assume a uniform prior $p_1 = \cdots = p_k = \frac{1}{k}$. Then the above simplifies to:

$$\hat{y} = \arg\max_c \frac{\ell_c(\vec{z})}{\sum_{c'=1}^{k} \ell_{c'}(\vec{z})}$$

To generate adversarial examples, a natural idea might be to minimize the posterior probability of the true class. We will now show why this is a bad idea. This idea is equivalent to maximizing the negative log-posterior of the true class:

$$f(\vec{z}) = -\ln \frac{\ell_y(\vec{z})}{\sum_{c=1}^{k} \ell_c(\vec{z})}$$

Let z_j and μ_{cj} represent the jth components of $\vec{z}$ and $\overrightarrow{\mu_c}$, respectively. The partial derivative of the objective function with respect to z_j is:

$$\frac{\partial f}{\partial z_j} = \left(z_j - \mu_{yj}\right) - \frac{\sum_{c=1}^{k} \ell_c(\vec{z}) \cdot \left(z_j - \mu_{cj}\right)}{\sum_{c=1}^{k} \ell_c(\vec{z})}$$

Assume the ResNet is accurate on unperturbed inputs (otherwise we wouldn't need to attack it adversarially.) Formally, assume $\vec{z} \approx \overrightarrow{u_y}$. Then since the maximum of a Gaussian PDF is achieved at the mean (the MMC center,) it follows that:

$$\ell_y(\vec{z}) \approx \ell_y\left(\overrightarrow{\mu_y}\right) = \max \ell_y$$

By construction of the MMC centers, the true class center $\overrightarrow{\mu_y}$ is "as far as possible" from all other centers. Since a Gaussian PDF vanishes quickly as we move away from the mean, it follows that:

$$\forall c \neq y, \ell_c\left(\overrightarrow{\mu_y}\right) \approx 0$$
$$\forall c \neq y, \ell_c(\vec{z}) \approx \ell_c\left(\overrightarrow{\mu_y}\right) \approx 0$$

Thus, the partial derivative simplifies to:

$$\frac{\partial f}{\partial z_j} \approx \left(z_j - \mu_{yj}\right) - \frac{\left(\max \ell_y\right)\left(z_j - \mu_{yj}\right)}{\max \ell_y}$$
$$= \left(z_j - \mu_{yj}\right) - \left(z_j - \mu_{yj}\right) = 0$$

Thus, the gradient $\vec{\nabla} f(\vec{z})$ almost vanishes. Any attack algorithm using this objective is weak and will make little progress.

<u>Attempt 2:</u>

The vanishing gradient in our first attempt was caused by the non-linearity of the objective. This time, let's try maximizing the negative log-likelihood instead of the negative log-posterior, which has the same maximizer:

$$f(\vec{z}) = \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2$$

The gradient is:

$$\vec{\nabla} f(\vec{z}) = \vec{z} - \overrightarrow{\mu_y}$$

Gradient ascent on this objective will always make progress, assuming $\vec{z}$ does not exactly equal $\overrightarrow{\mu_y}$ initially. The first gradient ascent step increases the distance between $\vec{z}$ and $\overrightarrow{\mu_y}$, causing subsequent gradients to become even larger. However, this objective is still flawed: when $\vec{z}$ moves away from the true class center $\overrightarrow{\mu_y}$, it might also be moving away from all other centers. Ultimately, $\overrightarrow{\mu_y}$ might still be the center closest to $\tilde{z}$. If this is the case, the perturbed image will still be correctly classified.

Note that this is the objective used in the MMC paper. The authors specifically chose this objective to circumvent the vanishing gradient problem discussed above. As we will show in the attempts below, it is possible to fix this objective's flaw as well, causing even more perturbed images to be incorrectly classified.

Attempt 3:

Our previous attempt failed because the objective didn't encourage moving $\vec{z}$ closer to any of the incorrect class centers. We design an objective that explicitly encourages this:

$$f(\vec{z}) = \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2 - \min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2$$

To maximize this objective, $\vec{z}$ will be moved away from the true class center $\overrightarrow{\mu_y}$ and towards some other center $\overrightarrow{\mu_c}$. Eventually, $\tilde{z}$ will be closer to an incorrect class center, causing the perturbed image to be incorrectly classified. The gradient is:

$$\begin{aligned} \vec{\nabla} f(\vec{z}) &= \left(\vec{z} - \overrightarrow{\mu_y}\right) - (\vec{z} - \overrightarrow{\mu_{c^*}}) \\ &= \overrightarrow{\mu_{c^*}} - \overrightarrow{\mu_y} \end{aligned}$$

where $c^* = \arg\min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2$.

There is one remaining complication: as long as c^* doesn't change, the gradient also doesn't change. Ideally, the attack algorithm should stop as soon as the image becomes misclassified.

Attempt 4:

We clamp our objective so its gradient is $\vec{0}$ as soon as the image is misclassified. This objective was inspired by Carlini and Wagner:

$$f(\vec{z}) = \min\left\{10^{-8}, \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2 - \min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2\right\}$$

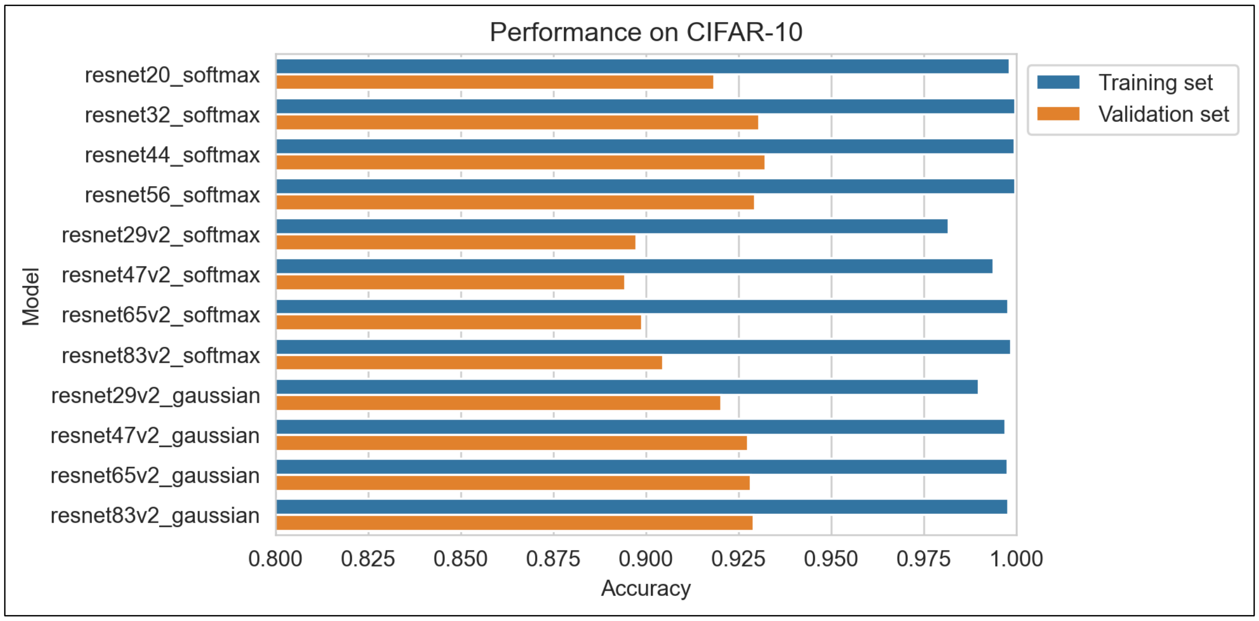
This objective is maximized when:

$$\begin{aligned} \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2 - \min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2 &\geq 10^{-8} \\ \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2 &\geq 10^{-8} + \min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2 \\ \frac{1}{2}\left\|\vec{z} - \overrightarrow{\mu_y}\right\|^2 &> \min_{c \neq y} \frac{1}{2}\|\vec{z} - \overrightarrow{\mu_c}\|^2 \end{aligned}$$

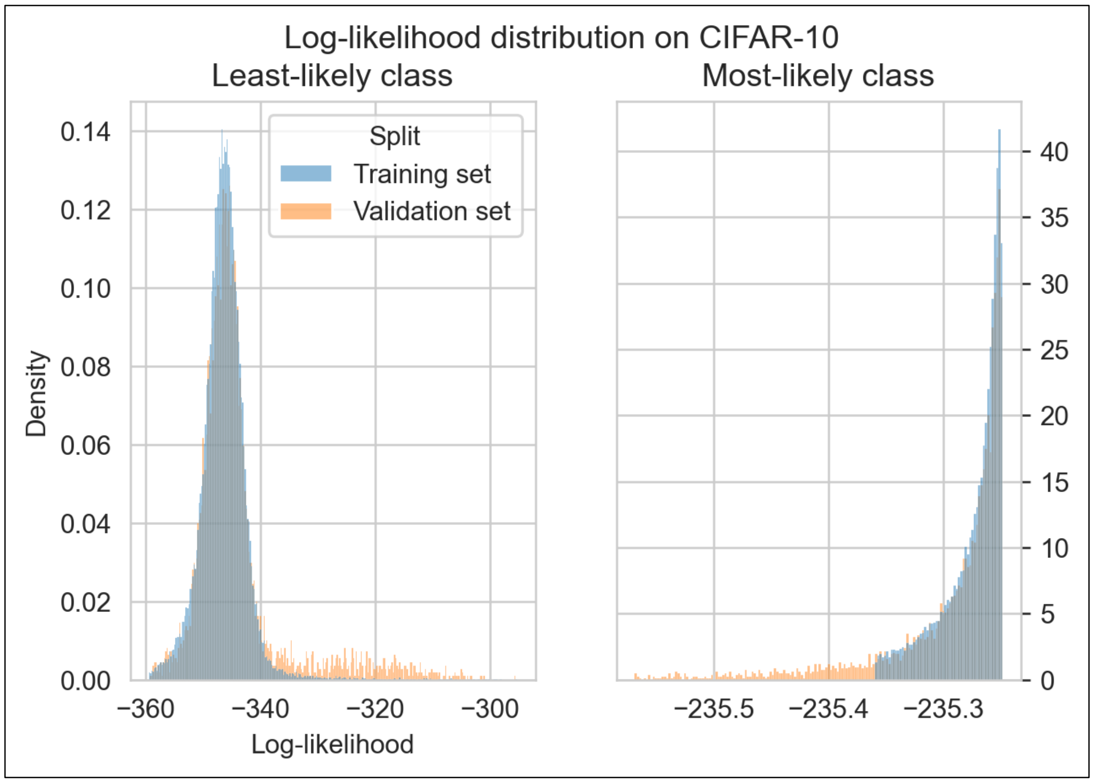
It follows that the objective is maximized iff the image is misclassified. Once the objective is maximized, its value is 10^{-8} (due to the outer min) its gradient is $\vec{0}$, thus stopping the optimization. This is the objective we maximize in our experiments with ResNet-Gaussian models.

Results

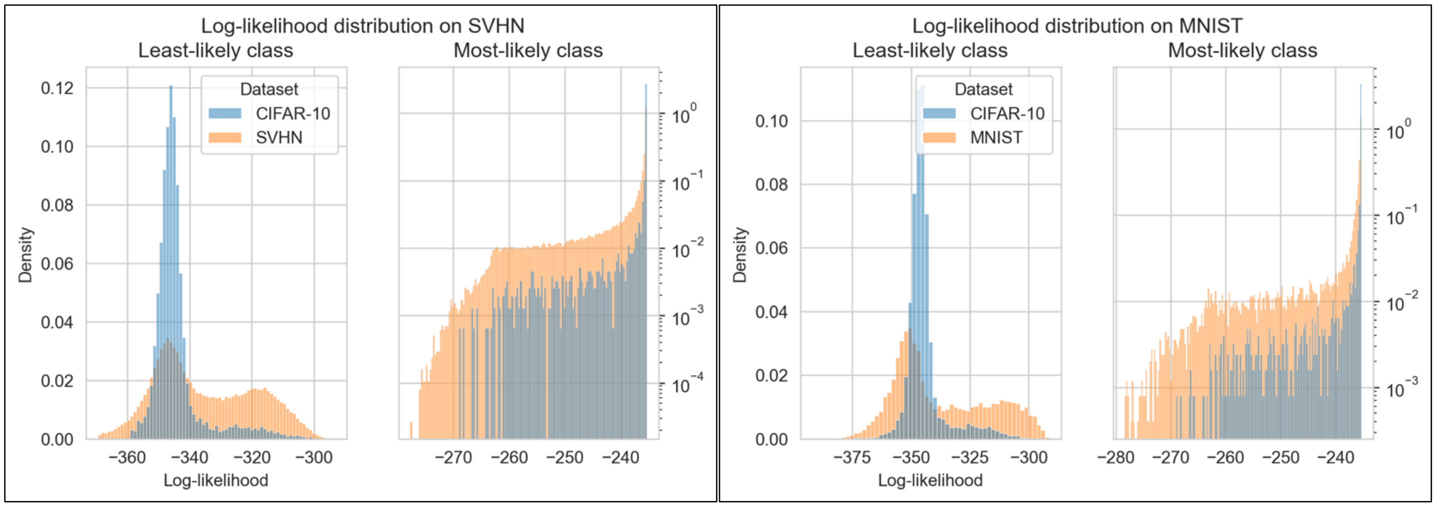
The bar graph below displays the training and validation accuracy achieved by each architecture on CIFAR-10. Note that the x-axis starts at 0.8. All training accuracies are nearly perfect. Within each architecture family, increasing the number of layers monotonically increases the training accuracy, showing that no optimization difficulty is encountered. Switching from ResNet-v1 to ResNet-v2 reduces the validation accuracy in general, while switching from softmax loss to Gaussian loss increases the validation accuracy in general. We argue that these minor differences in accuracy are sufficiently small to not be a significant concern, so we don't investigate the cause. This experiment shows that in general, the Gaussian loss does not degrade performance compared to the softmax loss.



Below, we visualize the minimum and maximum log-likelihood histograms predicted by ResNet83-v2-Gaussian on the (unperturbed) CIFAR-10 training and validation sets. To increase the readability of the histograms, we omit the bottom 1% observations from the minimum histogram and the bottom 15% observations from the maximum histogram. In either the minimum or maximum case, the training and validation histograms overlap substantially. This is expected since the training and validation sets come from the same underlying distribution. The overlap shows that the ResNet has indeed learned useful representations and is not just overfitting.



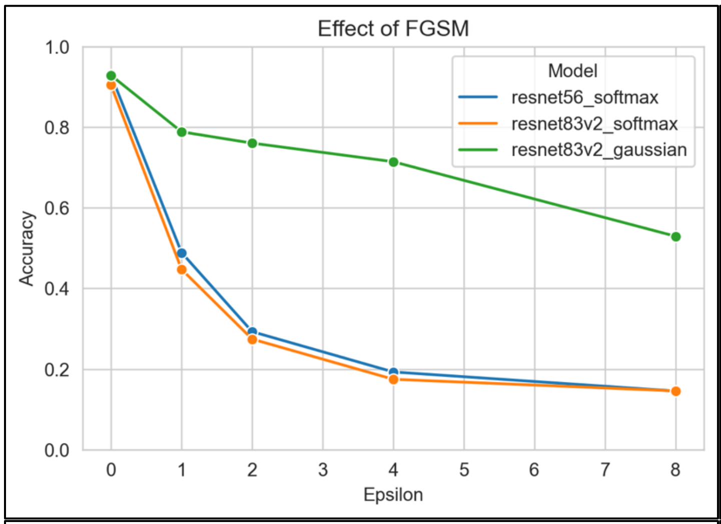
Below, we compare the histograms predicted on the in-distribution dataset (CIFAR-10) and the OOD datasets (SVHN and MNIST.) For readability, we omit the bottom 1% observations from the minimum SVHN histogram and we plot the maximum SVHN and MNIST histograms on a logarithmic y-axis. Although the in-distribution and OOD histograms have different shapes, they overlap enough that no log-likelihood threshold can separate in-distribution and OOD images. Interestingly, the Gaussian Likelihood Out of Distribution Detector[19] modifies the MMC loss by making the means and covariances of the Gaussians trainable; this modification significantly improves OOD detection accuracy.

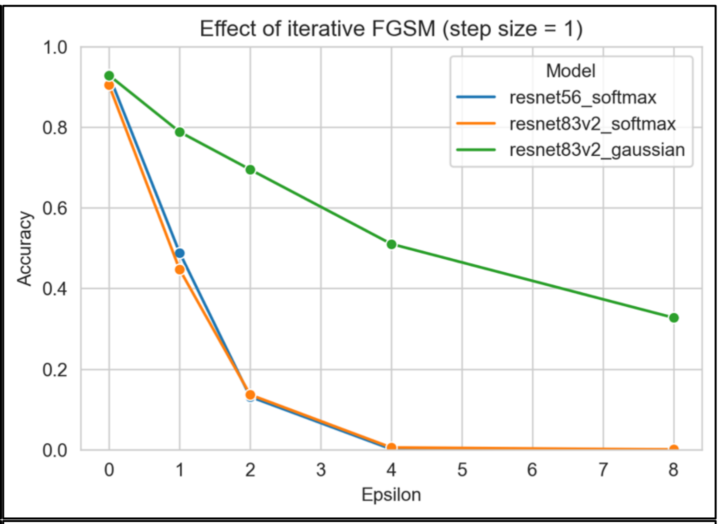


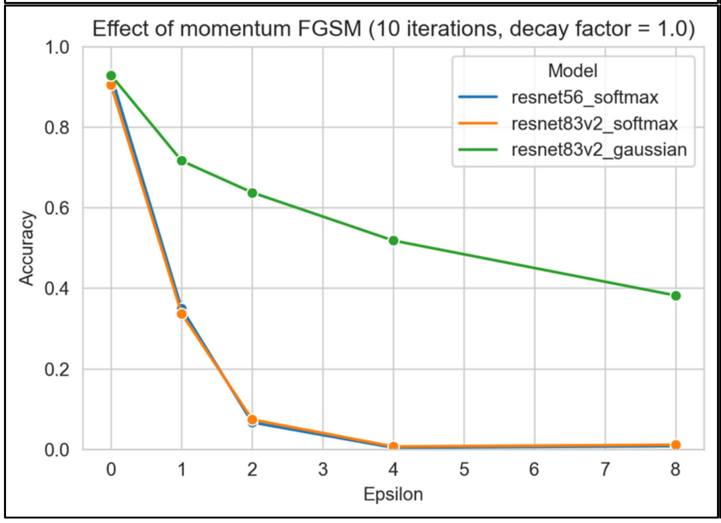
Below, we compare the effectiveness of the white-box attack algorithms on ResNet56-v1-Softmax, ResNet83-v2-Softmax, and ResNet83-v2-Gaussian. As the allowed perturbation amount ϵ increases, the model accuracy monotonically decreases as expected. PGD is the "ultimate" first-order adversary, achieving the lowest model

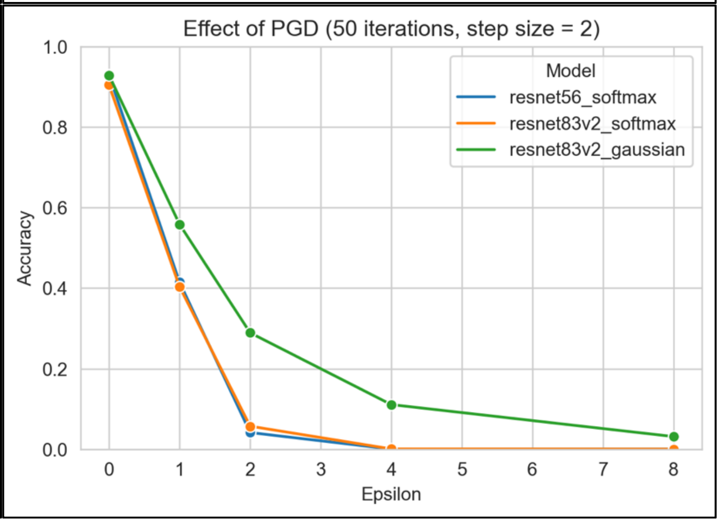
[19] https://arxiv.org/abs/2008.06856

accuracy scores out of the white-box attackers. The Gaussian model is noticeably more robust than the softmax models, both of which are equally robust.

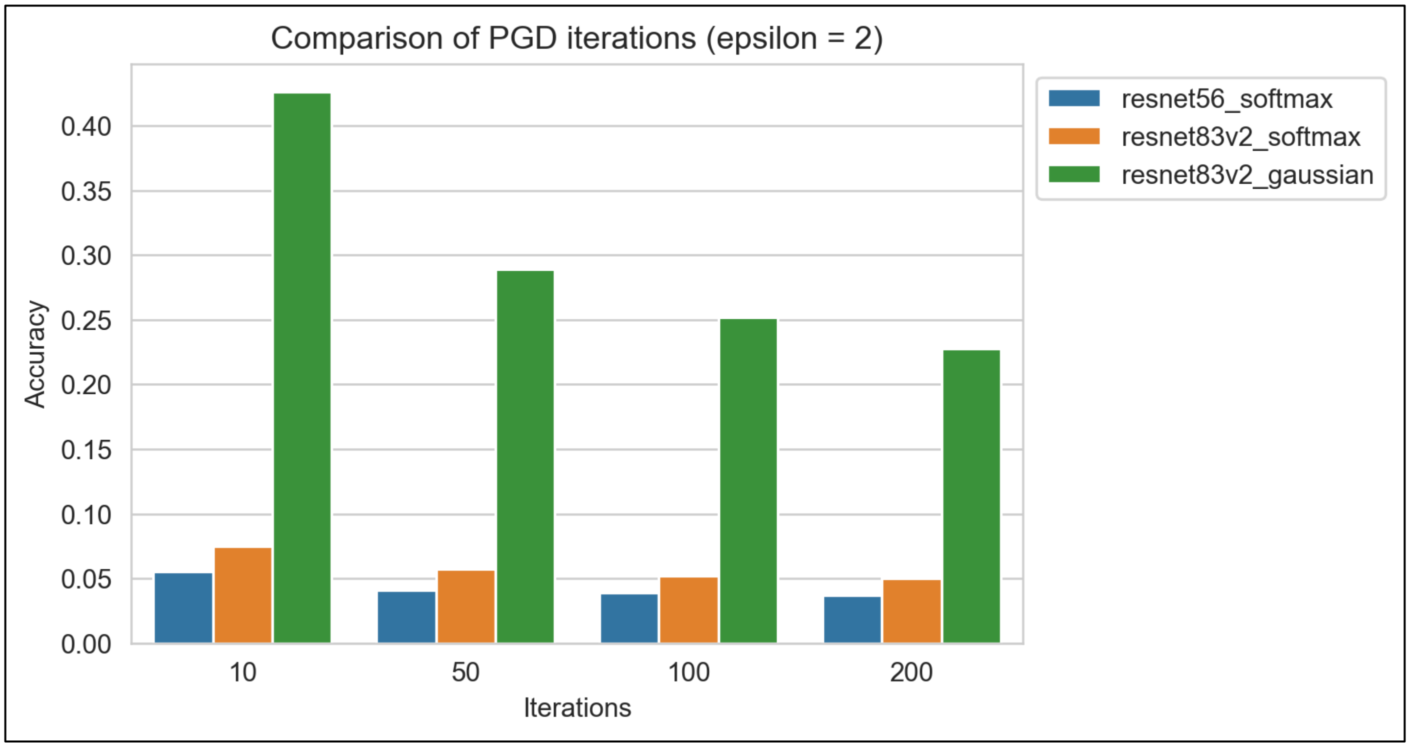








Below, we visualize the performance of PGD after different numbers of iterations. Contrary to the results reported in the MMC paper, PGD does not converge after 50 iterations. We hypothesize that the objective function we optimize is more effective than the one used in the MMC paper, allowing further PGD iterations to continue making progress.

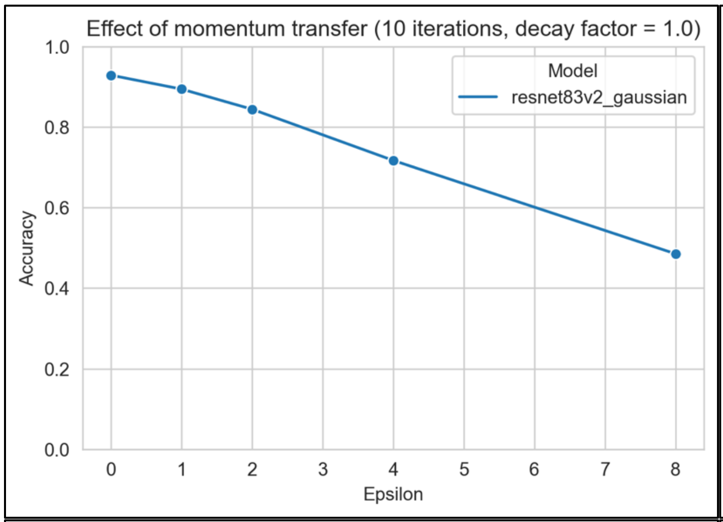


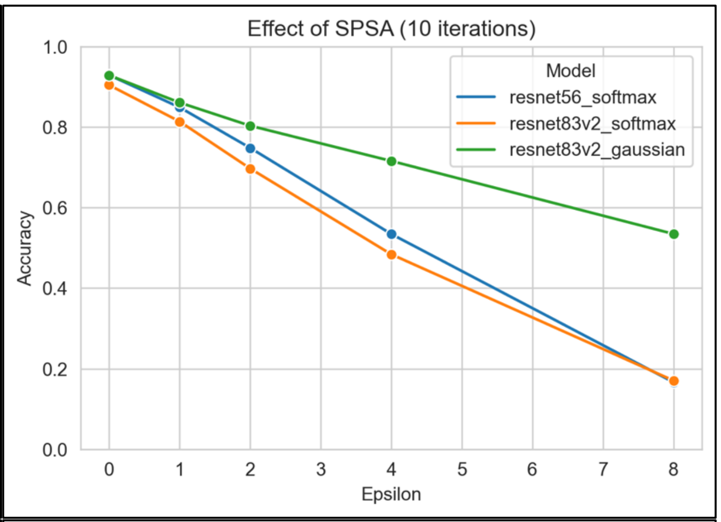
Below, we visualize the effectiveness of the black-box attack algorithms. The adversarial examples generated by momentum FGSM on ResNet83-v2-Softmax transfer surprisingly well to ResNet83-v2-Gaussian. At $\epsilon = 8$, both transfer (black-box) and non-transfer (white-box) momentum FGSM cause the accuracy to drop below 50%. Unfortunately, this shows that merely using a different loss function in place of softmax cross-entropy is insufficient for adversarial robustness. Future research in adversarial defense must revolve around methods of sanitizing the input so adversarial examples do not transfer from non-robust architectures. For example, feature denoising blocks[20] remove the adversarial noise.

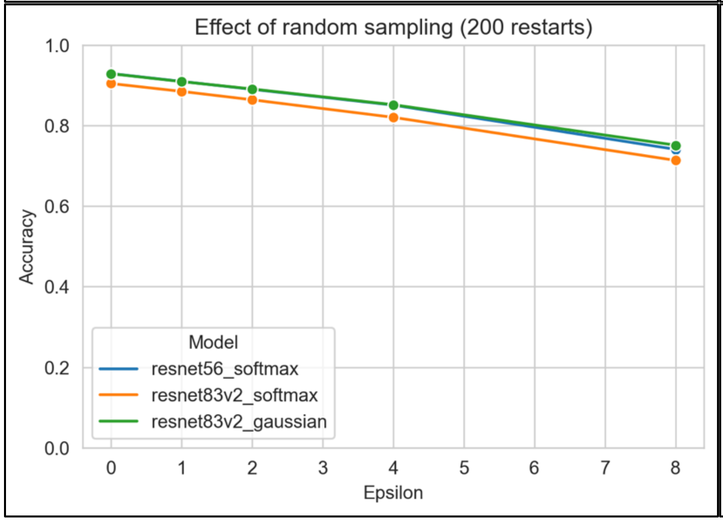
SPSA is less effective than PGD, showing that gradient masking is not in play. This is because PGD uses exact (analytic) gradients while SPSA uses approximate (numerical) gradients, so as long as analytic gradients are usable, PGD is the more effective attacker. Random sampling with only 200 restarts finds adversarial examples, further underscoring the need to pursue input sanitization as an approach to adversarial defense.

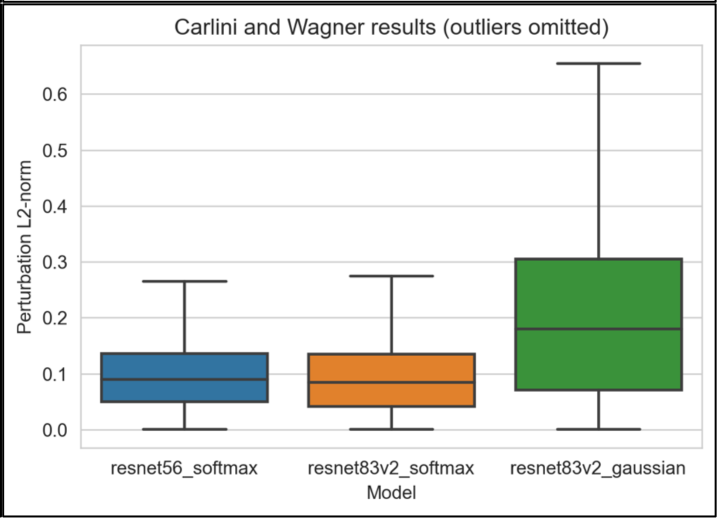
The Carlini and Wagner attacker calculates the minimum perturbation L2-norm necessary to cause an image to be misclassified. The boxplots below displays the distribution of L2-norms over the images in the CIFAR-10 validation set. Generally, perturbations must be twice as large to cause the MMC loss to misclassify an image, relative to the softmax loss.

[20] https://arxiv.org/abs/1812.03411v2

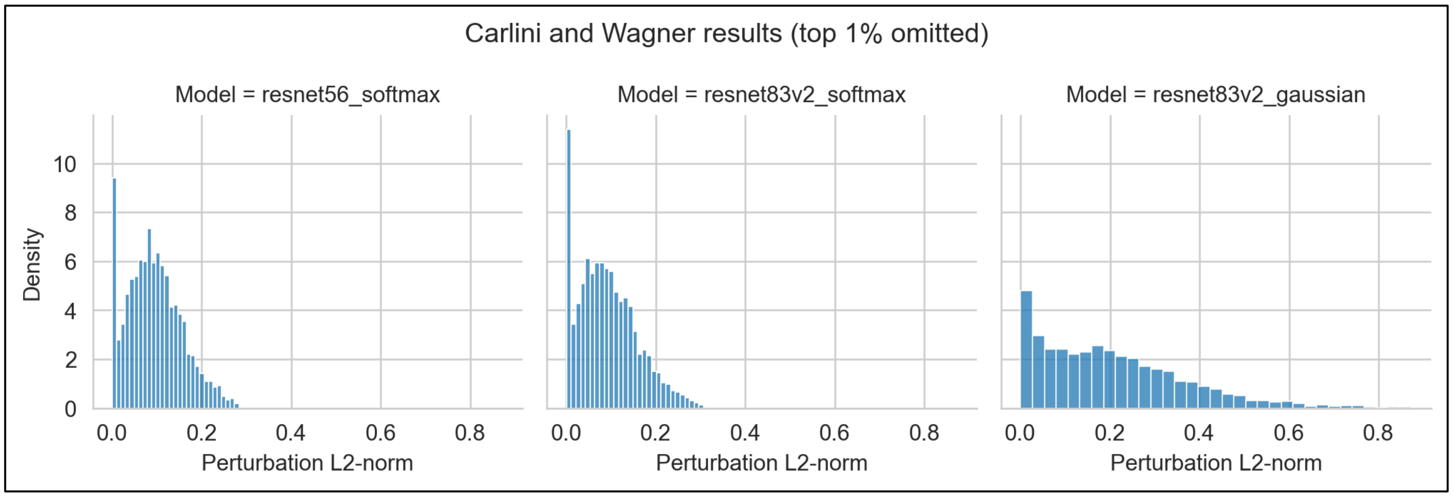






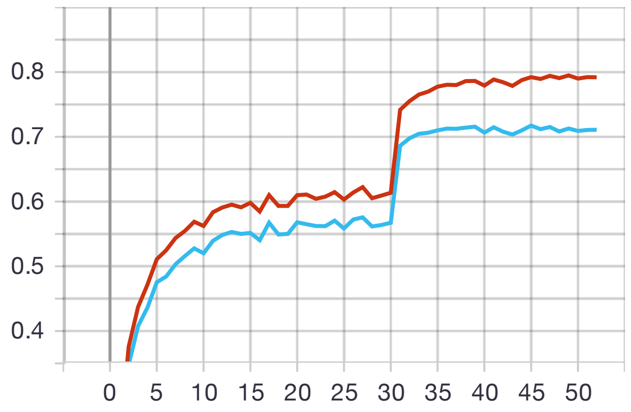


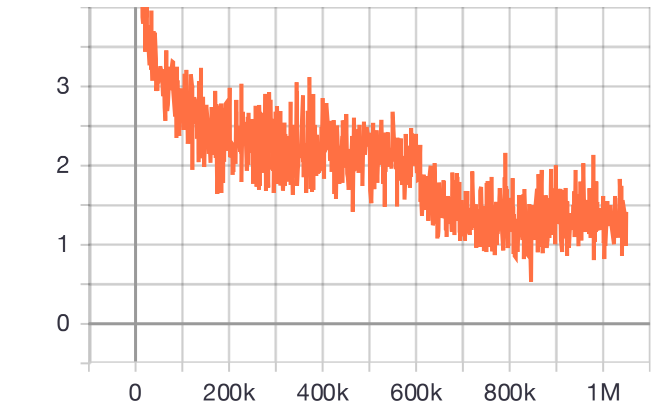
The histograms below depict the L2-norm distributions at a higher level of granularity than the boxplots above.



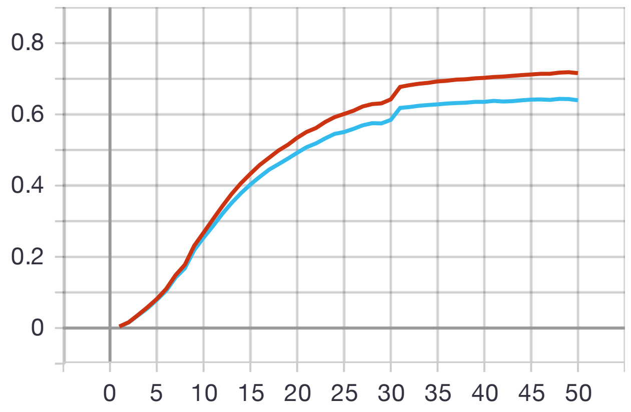
Below, we visualize the training process of ResNet101-v2-Softmax, ResNet101-v2-Gaussian, and ResNet152-v2-Gaussian on ImageNet. These results were obtained after training each model for 3 weeks on an Nvidia Tesla T4 GPU. During this time, ResNet101-v2-Softmax trained for 52 epochs, ResNet101-v2-Gaussian trained for 50 epochs, and ResNet152-v2-Gaussian trained for 37 epochs. Time constraints prevented us from training these models further. In the plots below, the red line represents epoch-wise training accuracy, the blue line represents epoch-wise validation accuracy, and the orange line represents batch-wise loss. The step-size increases in accuracy correspond to decreases in the learning rate.

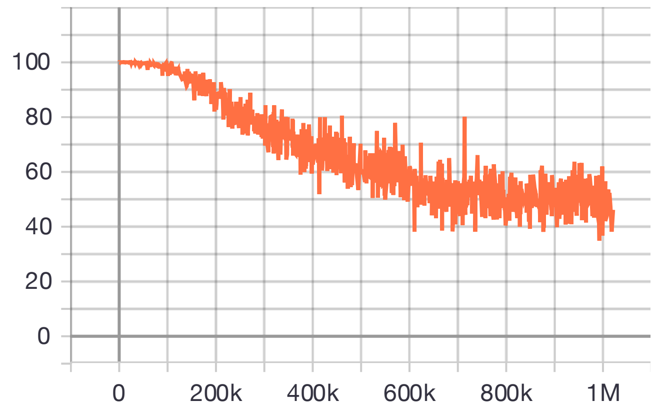
The plots below are for ResNet101-v2-Softmax. This architecture exhibits no problems in converging.



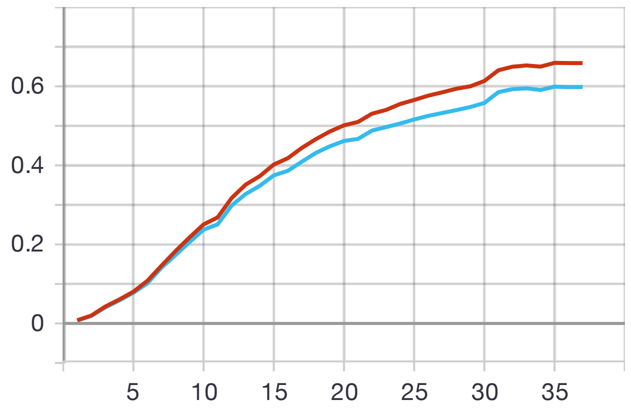


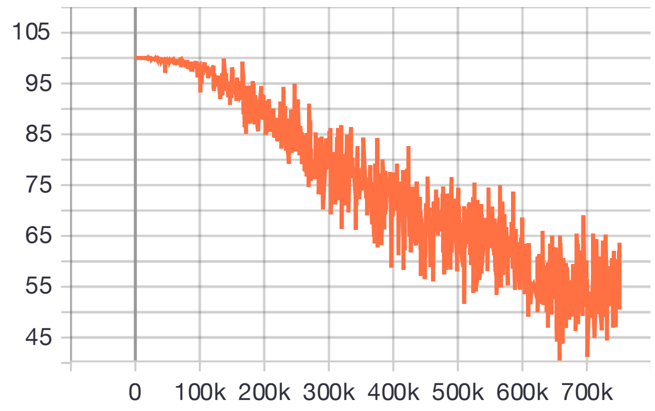
The plots below are for ResNet101-v2-Gaussian. The training begins slowly, but gradually speeds up after 5 epochs. This highlights the difficulty of optimizing non-convex loss functions, such as those for neural networks.



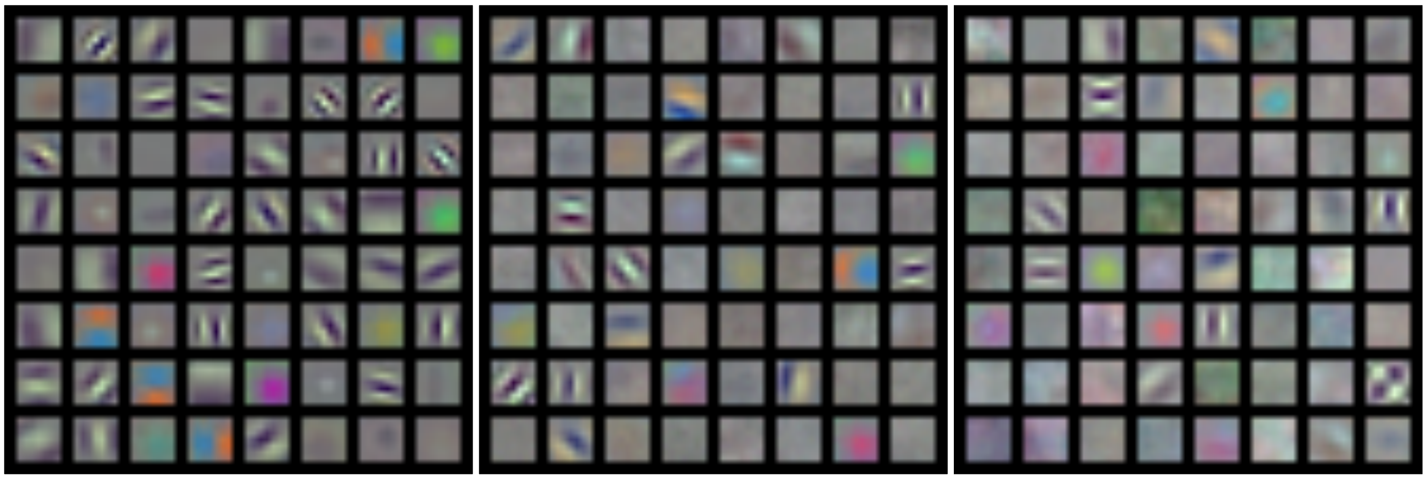


The plots below are for ResNet152-v2-Gaussian. Similarly, the training begins slowly but the model eventually achieves accuracies competitive with ResNet101-v2-Softmax.



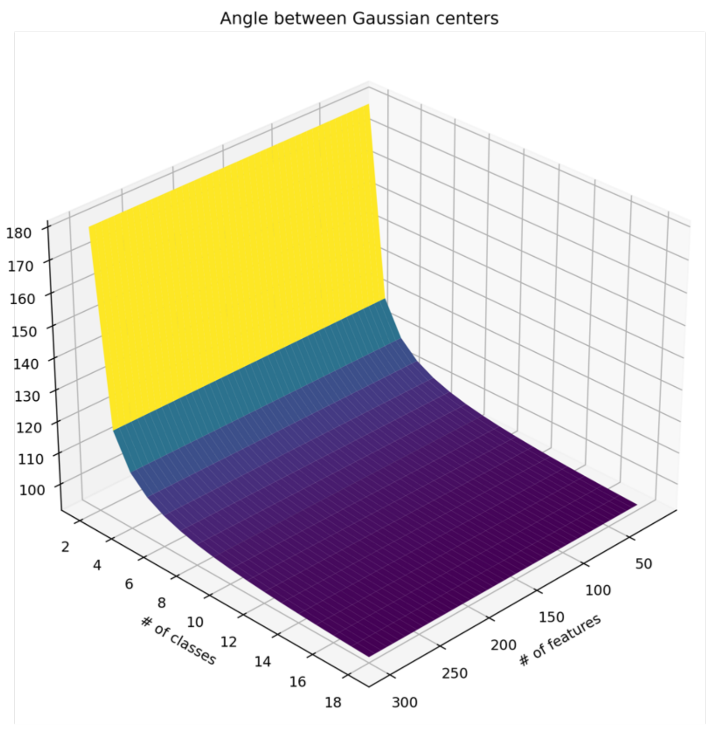


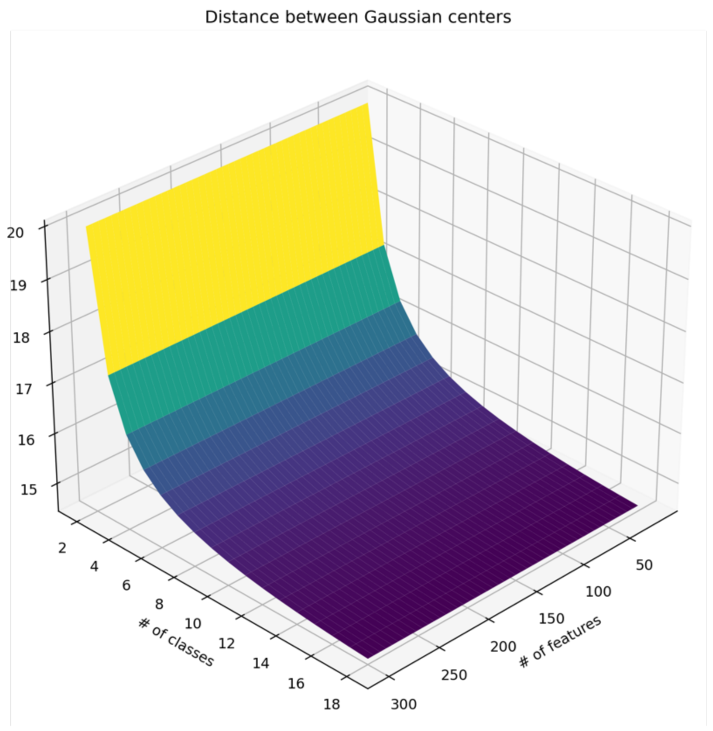
Below, we plot the trained weights of the first convolutional layer of ResNet101-v2-Softmax, ResNet101-v2-Gaussian, and ResNet152-v2-Gaussian, respectively. We observe clearly defined patterns such as stripes and corners in various orientations and colors. This is a good sign the training is proceeding smoothly.



Below, we plot the angle (in degrees) and Euclidean distance between two MMC centers as functions of the number of classes and the dimensionality of the hidden embeddings. We set the MMC radius as 10. Note that by definition of Max-Mahalanobis, the angle and distance between any two centers is identical regardless of the centers chosen.

We observe that angle and distance are independent of the dimensionality of the hidden embeddings. However, note that the dimensionality must be greater than the number of classes for Max-Mahalanobis to be well-defined. Refer to the center generation algorithm in the MMC paper for further details. Furthermore, increasing the number of classes has little effect. In practical applications with more than 10 classes, the angle hovers around $100°$ and the distance hovers around 14. Since the geometry of the MMC centers is not heavily impacted by increasing the number of classes, we expect that the MMC loss will effectively scale up to ImageNet-sized problems.





Future Work

Our highest-priority future goal is to implement Faster R-CNN and RefineNet. We will use the Microsoft COCO dataset[21] for detection and segmentation, and the PASCAL VOC dataset[22] for segmentation. We will implement the Dense Adversary Generation[23] algorithm to test the MMC loss's adversarial robustness when used in these larger models.

Preventing adversarial examples is crucial to ensure the safety of biometric authentication systems that rely on one-shot learning. In the future, we may also implement Siamese Neural Networks[24] and investigate the MMC loss's adversarial robustness on the Omniglot dataset[25].

[21] https://arxiv.org/abs/1405.0312
[22] http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf
[23] https://arxiv.org/abs/1703.08603
[24] https://www.cs.utoronto.ca/~gkoch/files/msc-thesis.pdf
[25] https://science.sciencemag.org/content/350/6266/1332