**DEEP LEARNING PROJECT**: **Ensemble methods as a Defense against Adversarial examples.**

Many Deep Neural Networks are vulnerable to adversarial perturbation.

[1] says that Linear behavior in high-dimensional space is sufficient to cause adversarial examples. It argues that non-linear behavior is not the cause for adversarial examples.

[1] also says that the adversarial examples generalize over different architectures which are trained for same task.

[1] says training on adversarial examples can regularize the model and sometimes better than dropout.

[1] proposes FGSM for generating adversarial examples. It considers the direction of gradient for each pixel. They incorporate the sign in to the training process itself.

[1] also mentions that it is better to perturb input images rather than perturbing hidden layers (no explanation of how it is done.)

[1] also says that ensemble provides only limited resistance to adversarial perturbation.

[1] says that the adversarial perturbations are highly aligned with the weight vectors of a model and the direction matters most.

[2] provides a taxonomy of attacks and defense mechanism proposed in literature.

[2] In our project we will mainly focus on Non-targeted attacks and just one targeted attack. Most of them will be white-box attacks and just one will be black-box attack. All attacks will be individual attacks. Both one-time and iterative attacks will be explored.

1. L-BFGS Attack: target, iterative cleverhans.attacks.**LBFGS**(*model*, *back='tf'*, *sess=None*)

2. FGSM: Non-target, one-time

cleverhans.attacks.**FastGradientMethod**

3.Jacobian Based Saliency Map Attack (JSMA): very slow, targeted, iterative

cleverhans.attacks.**SaliencyMapMethod**

4. Basic Iterative Method (BIM): Non-target, iterative

cleverhans.attacks.**BasicIterativeMethod**(*model*, *back='tf'*, *sess=None*)

Datasets:

1. MNIST
2. ImageNet

Defenses: (limit ensemble to 3 or 4 neural networks, otherwise it might take long time for just training.)

1. Network Distillation.
2. Train multiple Neural networks with random initial weights and take avg of final results. (majority vote)
3. Train different neural networks with different architectures and take avg of final results. (However according to [1], if adversarial examples are transferrable then this approach doesn’t work.)
4. Use bagging on training set to obtain different samples which are used for training ensemble.
5. Add noise to data and generate different sets of training data, now train each network in an ensemble.
6. Network distillation with different architectures and may be increasing number of networks to become an ensemble?
7. Combining 2,3,4 methods. Noise is added to training data, then random sampling is done, then each network (different architecture) is trained with the random noised data.
8. All of the above methods can be done using either majority vote or combining them just as in network distillation (decision trees vs random forests.)
9. Ensemble adversarial training.?

The gradients to be used for attacks can be calculated either using only a single classifier in the ensemble or the average of gradients from all networks in the ensemble.

1. We can consider gradient of easiest classifier and complex classifier and observe how it affects.
2. We can observe by taking average of all gradients.

General Procedure:

1. Build a network.
2. Train a network as mentioned in defenses.
3. Generate adversarial images using attacks as mentioned above.
4. Test these adversarial images on the trained network in 2. (If retraining then repeat 2, 3)
5. Report the results.

Architectures:

1. For MNSIT, use architecture from HW01.
2. For ImageNet, use Imagenet?