Research Progress

An Ensemble Neural Network Architecture with Victim Neural Network Integration for Automated Generation of Unknown and Unseen Adversarial Examples

Principal Investigator: Shayan Taheri
Responsibilities: Idea Development, System Design and Implementation, and System Performance Evaluation

Spring 2020

UCF

Outline

1. Introductory Points
2. System Architecture
3. Results
4. Future Directions

Key Points

1. Adversarial Examples → Intentional manipulation of data for cheating machine learning systems

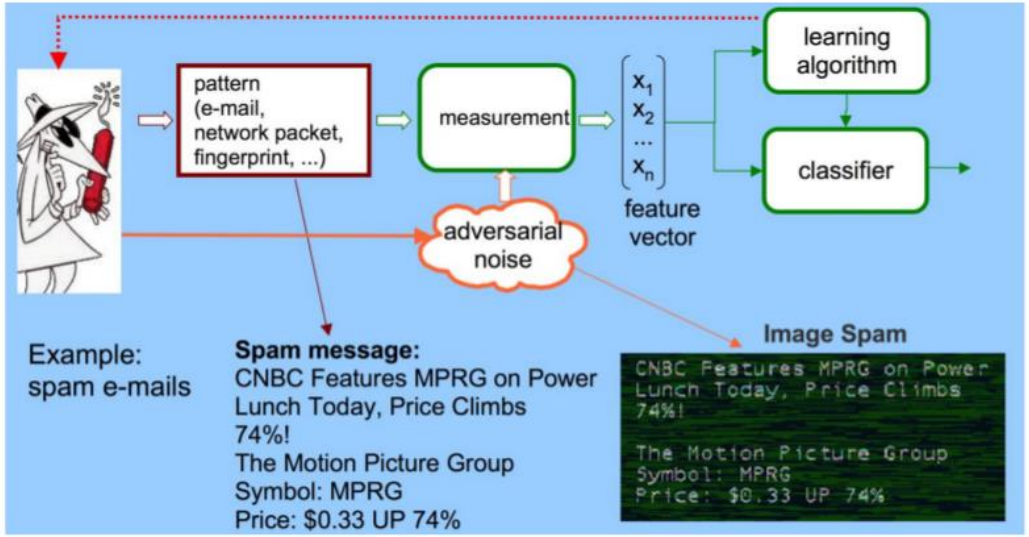
2. Adversarial Learning →

A. An independent research area to develop Adversarial-Aware machine learning systems in Adversarial Environment.

B. A Game between two players: the Classifier and the Attacker

3. Example Application: Spam-Emails Filtering

4. Conditional GAN?



Key Points for GAN

- Generative Adversarial Network
- Excellent test of our ability to use high-dimensional, complicated probability distributions
- GANs are generative models that use supervised learning to approximate an intractable cost function
- GANs can simulate many cost functions, including the one used for maximum likelihood
- Generative Adversarial Nets were recently introduced as a novel way to traingenerative models. In this work we introduce the conditional version of generativeadversarial nets, which can be constructed by simply feeding the data,y, we wishto condition on to both the generator and discriminator. We show that this modelcan generate MNIST digits conditioned on class labels. We also illustrate howthis model could be used to learn a multi-modal model, and provide preliminaryexamples of an application to image tagging in which we demonstrate how thisapproach can generate descriptive tags which are not part of training labels.
- Generative adversarial nets can be extended to a conditional model if both the generator and discrim-inator are conditioned on some extra informationy.ycould be any kind of auxiliary information,such as class labels or data from other modalities. We can perform the conditioning by feedingyinto the both the discriminator and generator as additional input layer

GAN Key Points

- In the generator the prior input noise pz(z), andyare combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed.1In the discriminator x and y are presented as inputs and to a discriminative function (embodied again by a MLP in this case).

- A generative adversarial network, or GAN for short, is an architecture for training deep learning-based generative models.

- The architecture is comprised of a generator and a discriminator model. The generator model is responsible for generating new plausible examples that ideally are indistinguishable from real examples in the dataset. The discriminator model is responsible for classifying a given image as either real (drawn from the dataset) or fake (generated).

- The models are trained together in a zero-sum or adversarial manner, such that improvements in the discriminator come at the cost of a reduced capability of the generator, and vice versa.

- GANs are effective at image synthesis, that is, generating new examples of images for a target dataset. Some datasets have additional information, such as a class label, and it is desirable to make use of this information.

GAN Key Points

- Conditional GANs (CGANs) are an extension of the GANs model. You can read about a variant of GANs called DCGANs in my previous post here. CGANs are allowed to generate images that have certain conditions or attributes.

- Conditional GANs (CGANs): The Generator and Discriminator both receive some additional conditioning input information. This could be the class of the current image or some other property.

- Conditional adversarial networks as a general-purpose solution to image-to-image translation problems.

- These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. We demonstrate that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks.

Adversarial Key Points

- Adversarial Sample / Adversarial Example

  A. Examples that are similar to examples in the true distribution, but that fool a classifier.
  B. Modified versions of a clean image that are intentionally perturbed (by adding noise) to confuse a machine learning system.

- Black-box Attacks: The adversary has no knowledge of the target model.

- Semi-black-box Attacks: The adversary has a limited knowledge of the model (e.g. architecture) but definitely does not know the parameters.

- White-box Attacks: The adversary has a complete knowledge of the target model.

System Key Points

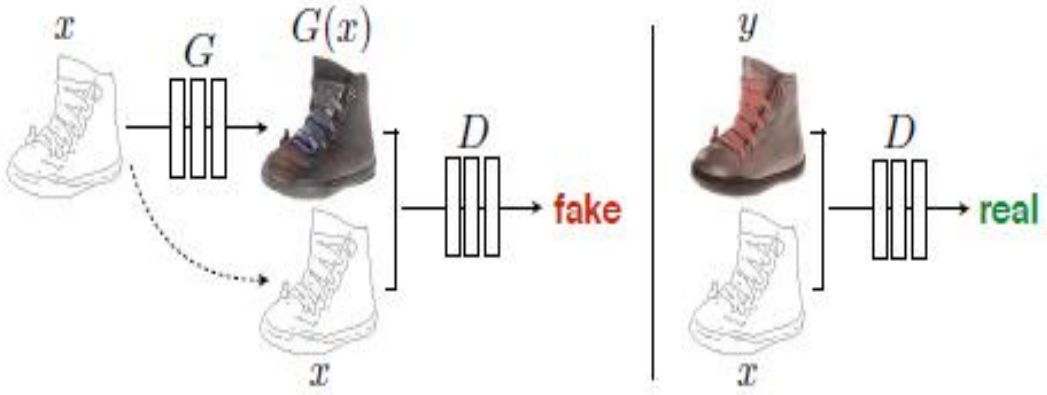
- Threat Model: Guided generation of Unknown and Unseen adversarial examples
- Create/Train/Validate of Neural Networks in Both Domains of Inspection
- Executing Well-Known Attacks of FGSM and DeepFool on Pre-Trained Neural Network on Clean Data
- Neural Network For Initial Generation of Attacks → Let's call it Victim Neural Network
- Dataset to Use: Initial Plan = Software Normal and Malware Data
- Dataset to Use: Final Plan = Network Normal and Botnet Traffic Data
- Modified Pix2Pix Architecture → Pix2Pix System Architecture + Victim Neural Network Architecture + Weights for Victim Neural Network (Updating Constantly)
- Modified Pix2Pix Architecture → Ensemble of Neural Networks → Artificially Intelligence Attack Generator

System Key Points

- Adversarial Data Generator → Understanding Data Distributions and Deep Understanding of the Attacks → Producing New, Unseen, and Highly Complex Malicious Data
- Modified Pix2Pix Architecture!
- Are the generated adversarial examples artificially strong enough in fooling the initial neural network?

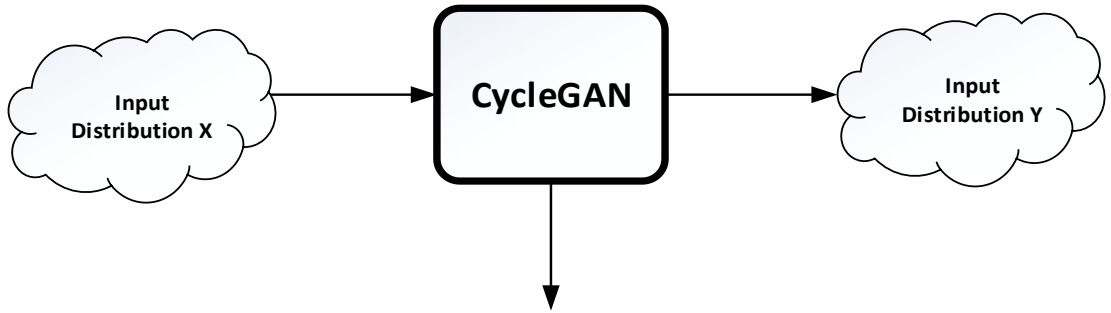
Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

System Key Points



Training a conditional GAN to map edges!photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real fedge, photog tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

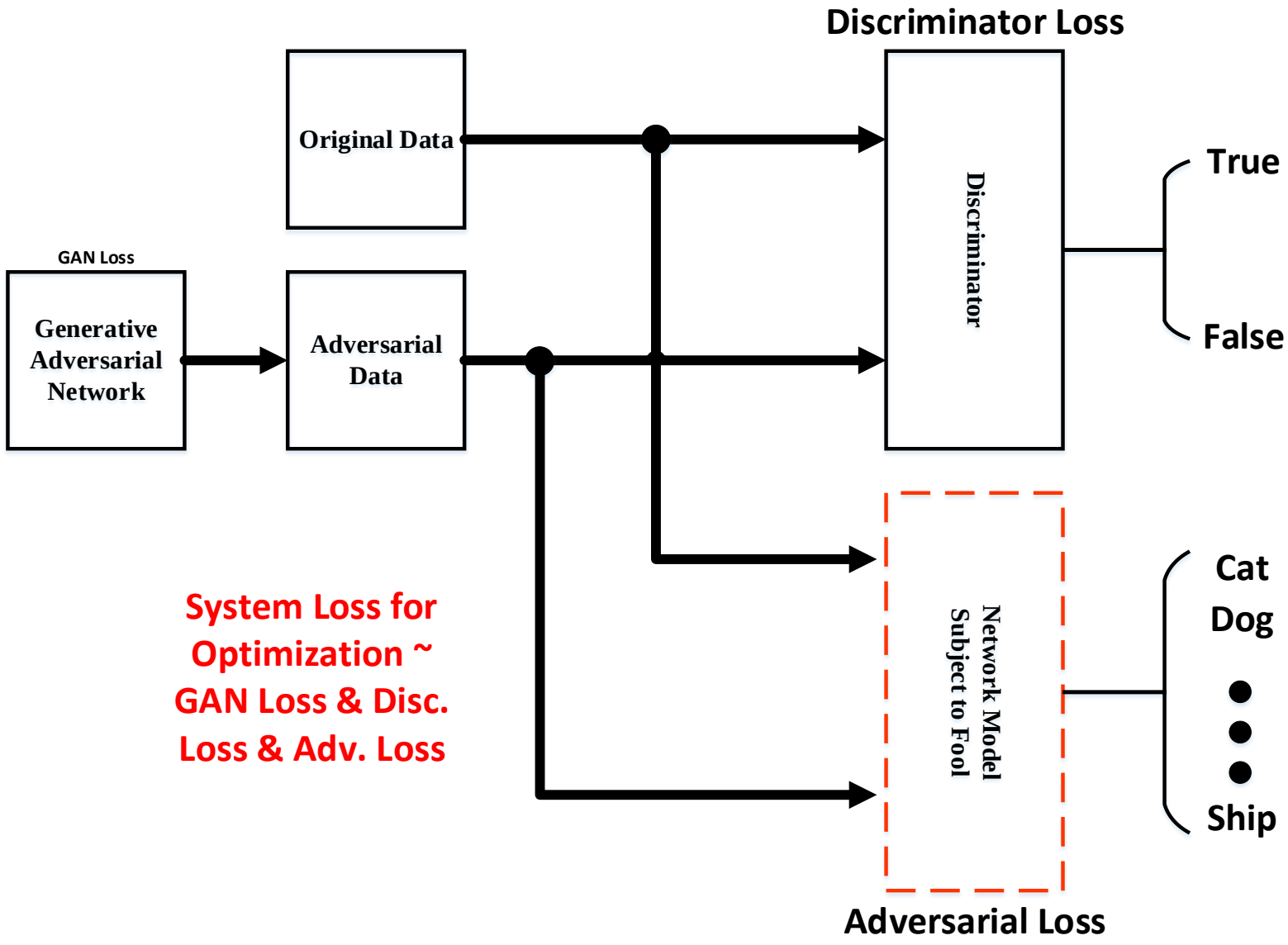
Different losses induce different quality of results. Each column shows results trained under a different loss.



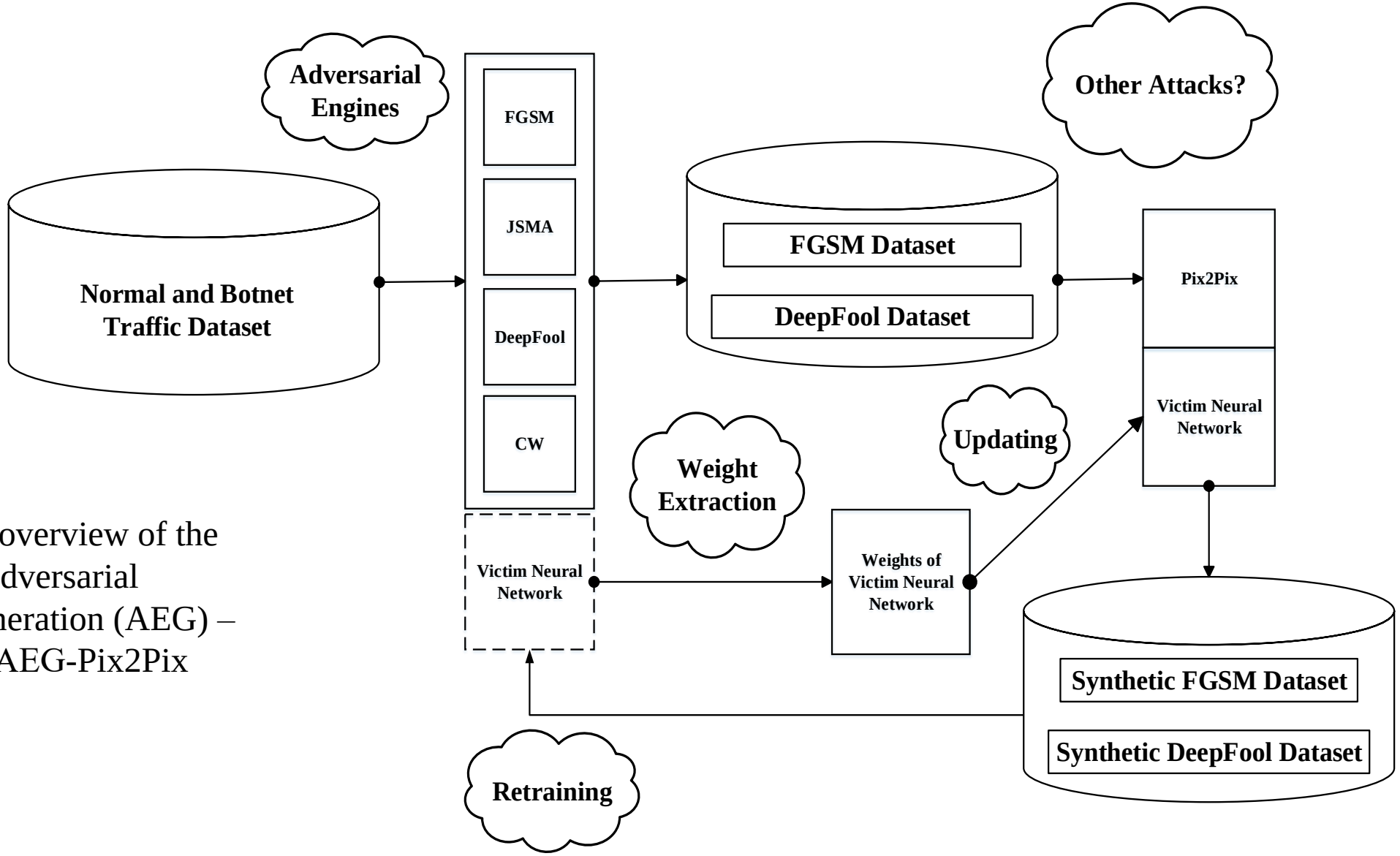
Getting a deep understanding of the relationship between these data and training the CycleGAN in order to find the transformation between malicious data and normal data.

System Architecture

The modified Pix2Pix system: It includes the Pix2Pix connected to the victim neural network pre-trained constantly by its updated model.



System Architecture



The overall overview of the system for adversarial example generation (AEG) – Let's call it AEG-Pix2Pix System!

Initial Results

Table 1: The results for showing the attacking strength of the data generated by Pix2Pix..

Point: We have Stronger Attacks!

Evaluation Case	Training Dataset	Testing Dataset	Epoch Number	Number of Retraining Iteration	Fooling Number
1 (Reference)	FGSM – Original Adversarial Training Data	FGSM – Original Adversarial Testing Data	N/A	1	0/9969
2.1	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	01	1	2742/9969
2.2	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	10	1	2767/9969
2.3	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	20	1	580/9969
2.4	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	30	1	584/9969
2.5	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	40	1	730/9969
2.6	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	50	1	1633/9969
2.7	FGSM – Original Adversarial Training Data	FGSM – Pix2Pix Adversarial Testing Data	60	1	3968/9969

UCF

Initial Results

Point: Epoch 20 can be selected!

Table 2: The results for showing the attacking strength of the data generated by Pix2Pix after retraining and defense capability.

Point: We can defend!

Evaluation Case	Training Dataset	Testing Dataset	Epoch Number	Number of Retraining Iteration	Fooling Number
1 (Reference)	FGSM – Original Adversarial Training Data	FGSM – Original Adversarial Testing Data	N/A	1	0/9969
3.1	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	01	1	765/9969
3.2	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	10	1	673/9969
3.3	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	20	1	606/9969
3.4	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	30	1	831/9969
3.5	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	40	1	550/9969
3.6	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	50	1	841/9969
3.7	FGSM – Pix2Pix Adversarial Training Data	FGSM – Original Adversarial Testing Data	60	1	578/9969

UCF

To Do Tasks

- Running the system for more number of iterations! The simulation is run for fixed number of epochs.
- Designing new attacks using other types of noises for fooling neural networks.