FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Pattern Recognition Lab

Deep Learning Algorithms in Remote Sensing Image Scene Classification: Adversarial Attacks and Defenses

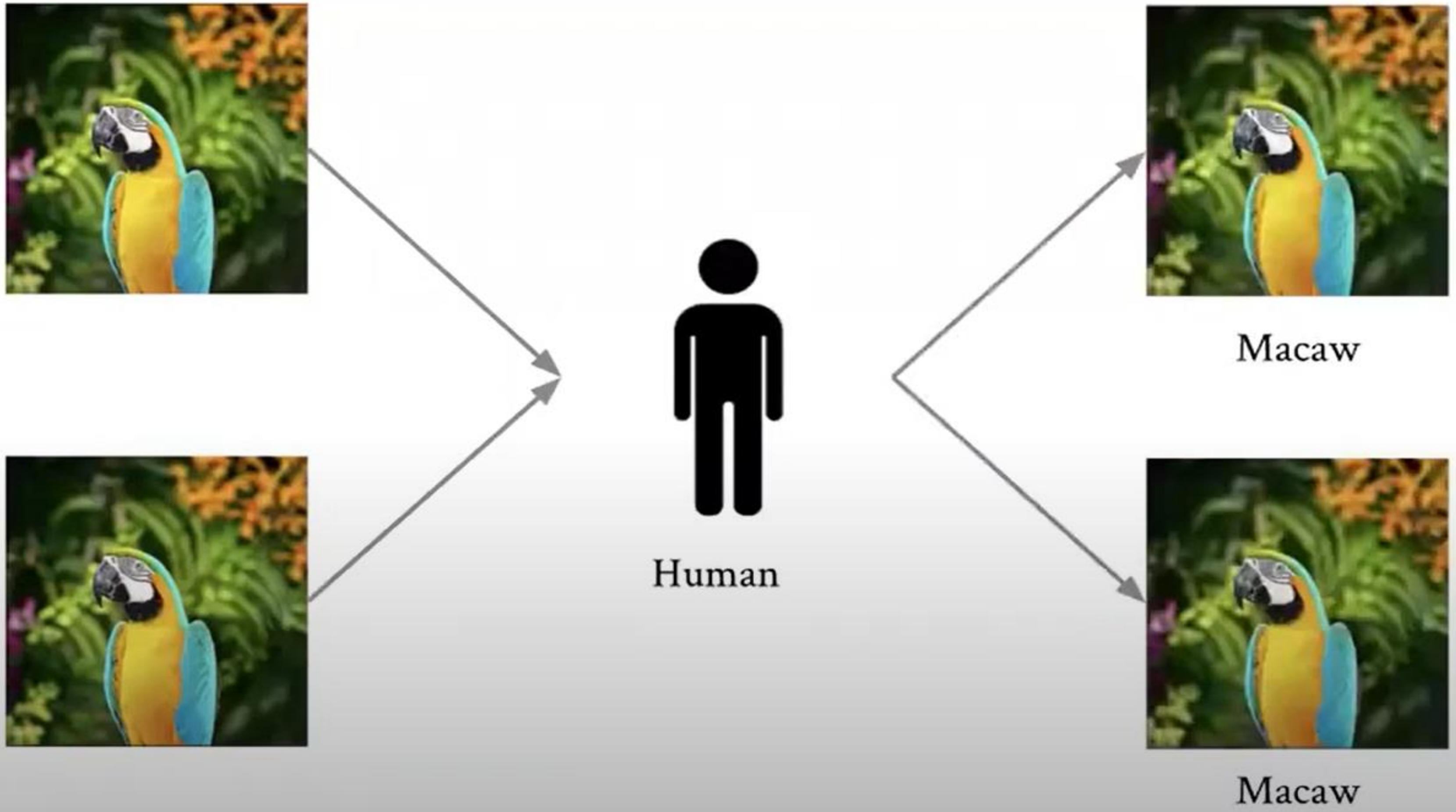
Supervisor: **Prof. Dr. Andreas Maier**
Pattern Recognition Lab
Dept. of Computer Science
University of Erlangen-Nuremberg

SHORYA SHARMA
19EE01017
School of Electrical Sciences
Indian Institute of Technology Bhubaneswar

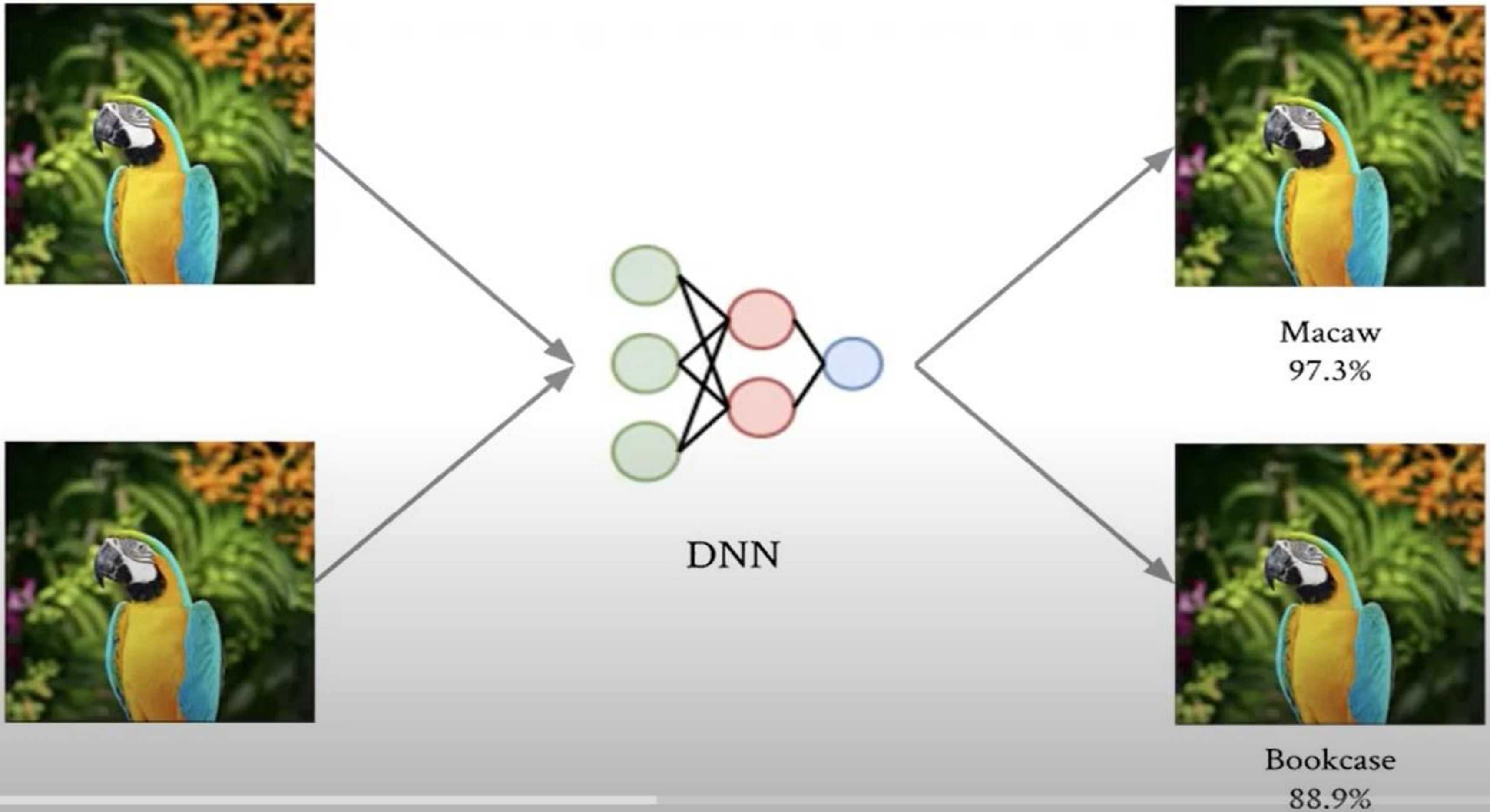
Introduction: Aerial Scene Classification

- Aerial scene classification is the foundation and important technology of ground object detection, land use management and geographic analysis
- Most traditional RSI interpretation algorithms design the hand-craft features for different applications. However, these well-designed features cannot solve the RSI interpretation problem well since traditional algorithms need to be carefully designed with solid domain knowledge in order to be effective in different applications.
- During recent years, convolutional neural networks (CNNs) have achieved significant success and are widely applied in RSI scene classification. These RSI scene classification models perform much better in terms of accuracy than traditional RSIs.

Are DNNs as good as the human eye?



Are DNNs as good as the human eye?

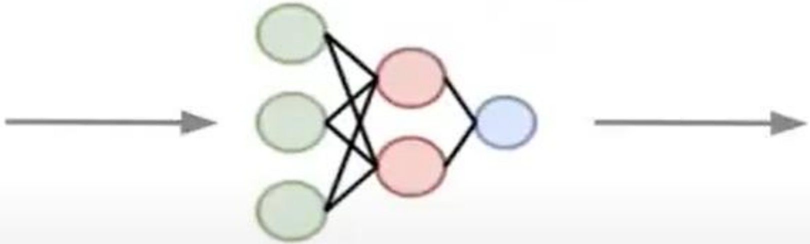


Problem: DNNs are vulnerable to Adversarial Samples

Forcing a DNN to misclassify an input using an Adversarial Sample is called an **Adversarial Attack**

Example -

Stop sign

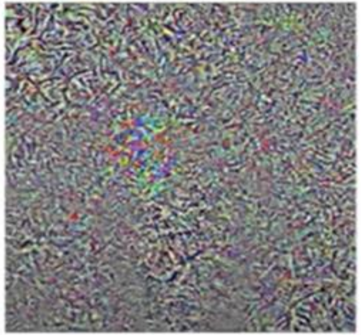


Misclassified by the DNN as a speed sign

This can be hazardous in scenarios where an autonomous vehicle relies on deep learning based computer vision to detect road signs

Adversarial Samples

X



Noise

Perturbed Image
aka **Adversarial sample**

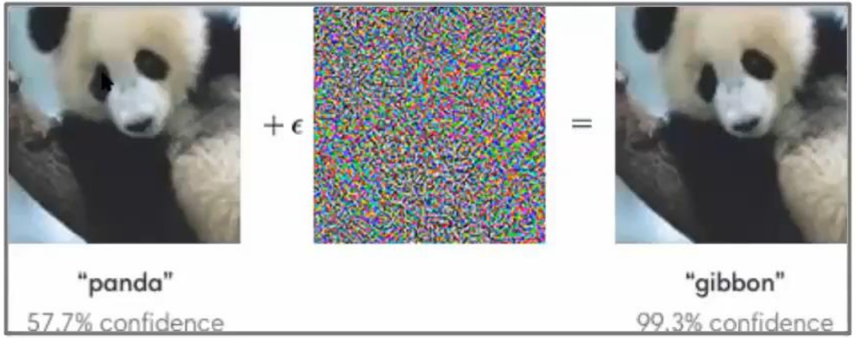
97.3%
Macaw

88.9%
Bookcase

DNNs can be easily fooled to misclassify images that are imperceptible to human eye!

Background - What are Adversarial Attacks?

Adversarial attacks are *imperceptible perturbations* which cause neural networks to fail.



- Inputs are changed in the direction of the gradient w.r.t the target model's outputs
- Can be whitebox attacks (attacker knows everything about the model) or blackbox (attacker doesn't know anything about the model)
- These attacks can even make a well trained model perform badly!!
- Adversarial attacks are a big risk for Deep Learning models deployed in production

Hence, we need Defenses against this dark art of adversarial attacks!

Objectives and Tasks

- To explore the properties of adversarial examples of RSI scene classification, we create different scenarios by testing 29 major attack algorithms trained on different RSI benchmark datasets to fool CNNs.
- In the experiment, we train several CNN models with the widest application in RSI scene classification systems. In these high-accuracy CNNs, we use a variety of attack algorithms to generate different adversarial examples.
- To implement state-of-the-art defense mechanisms to defend against these adversarial attacks on these RSI datasets. The different defense mechanisms implemented in this project are: adversarial training, multi-SAP networks and GANs defense mechanisms.

Adversarial Attacks in RSI

- The adversarial example problem of RSIs can be represented as follows:

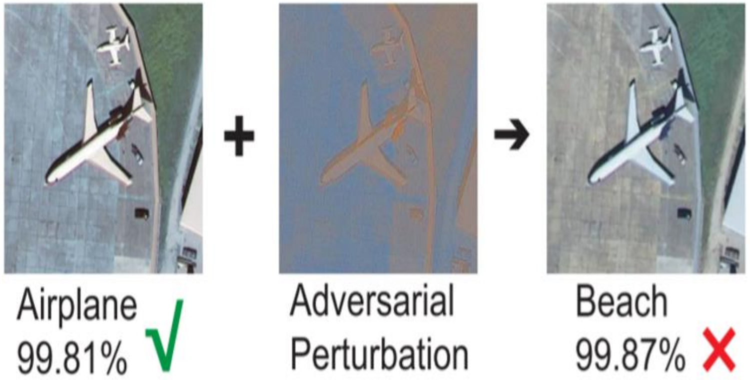
$$f(x+\rho) \neq y \qquad s.t.\ \min_{\rho} ||\rho||$$

where $\min_{\rho} ||\rho||$ denotes the regularization constraint on the adversarial perturbation, and it ensures that the change of the perturbation is small

- The amount of distortion is an important measure of the quality of adversarial example. The smaller the distortion of adversarial example, the closer it is to the original example, the more difficult it is to detect and recognize.

Crafting adversarial examples requires following elements:

- a model that takes an input (e.g. an image) and makes a prediction (e.g. class-probabilities).
- a criterion that defines what an adversarial is (e.g. misclassification).
- a distance measure that measures the size of a perturbation (e.g. L1-norm).
- Finally, an attack algorithm that takes an input and its label as well as the model, the adversarial criterion and the distance measure to generate an adversarial perturbation



On the left is the original airplane. The RSI scene classification system can correctly classify it as an airplane with confidence of 99.81%. However, after adding an adversarial perturbation to the image, the RSI scene classification system can classify it as a beach with 99.87% confidence. The latter result is wrong, and we refer to the modified image as an adversarial example. Obviously, the adversarial example cannot affect human classification, but it leads to system errors and has serious consequences.

Datasets used for Adversarial Attacks

❖ NWPU-RESEIC45

RESISC45 dataset is a publicly available benchmark for Remote Sensing Image Scene Classification (RESISC), created by Northwestern Polytechnical University (NWPU). This dataset contains **31,500** images, covering **45** scene classes with **700** images in each class.

❖ UC Merced Land Use Dataset

UC Merced is a **21** class land use remote sensing image dataset, with **100** images per class. The dataset contains **2100** images which were manually extracted from large images from the USGS National Map Urban Area Imagery collection for various urban areas around the country. The pixel resolution of this public domain imagery is **0.3** m.

Experiments Details

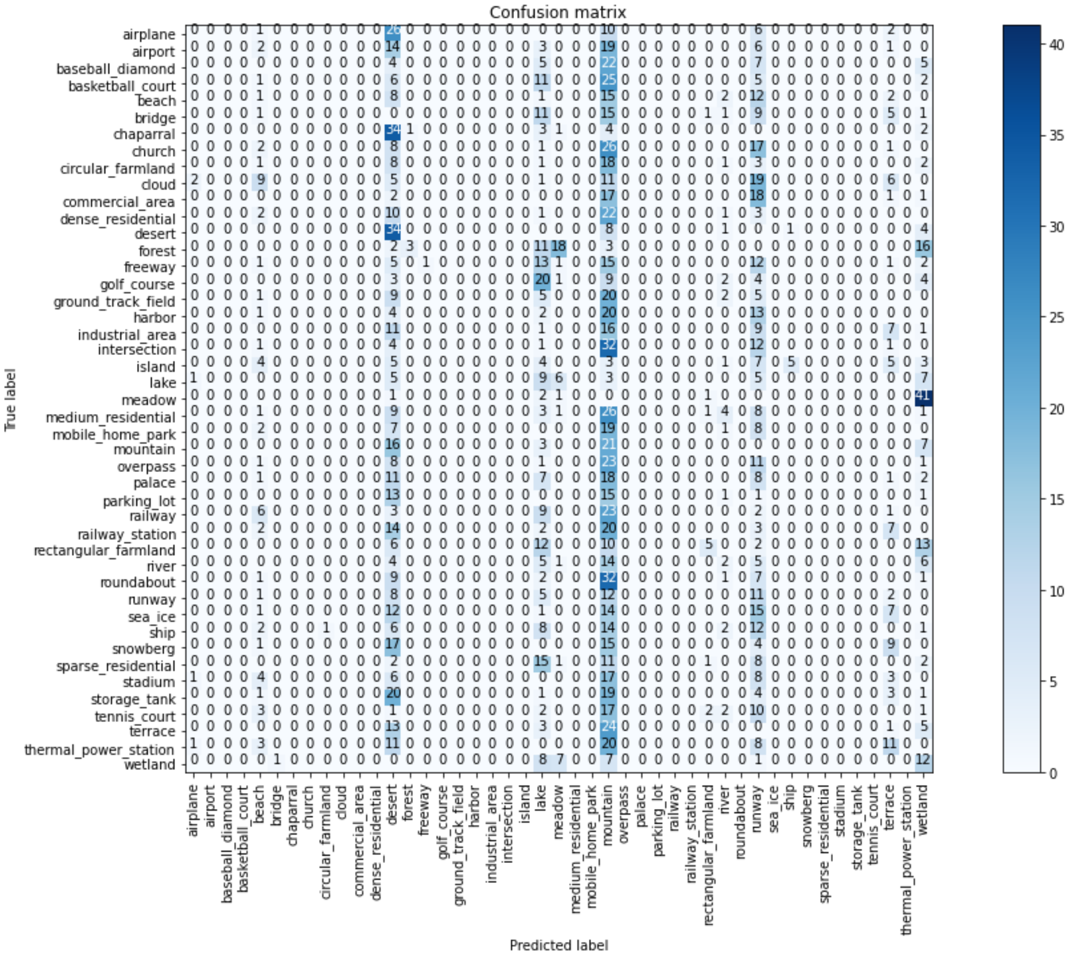
- Deep Learning Framework: Pytorch
- GPU: Tesla M60(batch size=16)
- Neural Network Architectures:
  - Alex Net
  - ResNet50
  - ResNet101
  - MobileNetV2
  - DenseNet121
- Optimizer: Adam Optimiser(learning rate=0.01, weight decay rate= 0.001)
- Loss Function: Cross Entropy Function
- Epochs: 100
- Train Set : Test Set :: 80:20
- Epsilon: 0.05; 1.0
- Evaluation Metrics: Accuracy, Confusion Matrix

Results

- **NWPU-RESIC45**
  - Epsilon=0.05

	Original Accuracy	0.747058824	0.885373609	0.892686804	0.894117647	0.914149444
	Best Attack Accuracy	0.185	0.479	0.381	0.573	0.449
	Percentage of Performance Drop	**75.23622047**	45.89854552	57.31985752	35.91447368	50.88330435
Attack No.	Attack Name	AlexNet	Resnet50	Resnet101	MobileNet_v2	DensetNet
1	L2 Contrast Reduction Attack	0.206999958	0.582499981	0.55249998	0.604999989	0.653999984
2	Virtual Adversarial Attack	**0.185499966**	0.566999972	0.567499965	0.605999976	0.655499995
3	DDNA Attack	0.204999983	0.565999985	0.566499978	0.597499967	0.660999984
4	L2 Projected Gradient Descent Attack	0.20449996	0.549499989	0.56249997	0.602999985	0.648499995
5	Linf Projected Gradient Descent Attack	0.20449996	0.552999973	0.561499983	0.592499971	0.661999971
6	L2 Basic Iterative Attack	0.194999933	0.57249999	0.563999981	0.583499968	0.67049998
7	Linf Basic Iterative Attack	0.20449996	0.555999964	0.554499984	**0.573499978**	0.631499976
8	L2 Fast Gradient Attack	0.21299994	0.555999964	0.557999969	0.627999991	0.66049999
9	Linf Fast Gradient Attack	0.201499939	0.563999981	0.551999986	0.602499992	0.660999984
10	L2 Repeated Additive Gaussian Noise	0.21299994	0.552999973	0.552999973	0.593499988	0.664499998
11	L2 Repeated Additive Uniform Noise	0.206499934	0.584999979	0.539499968	0.617499977	0.662999988
12	L2 Clipping Aware Repeated Additive Gaussian Noise Attack	0.20599997	0.564499974	0.559999973	0.604999989	0.661499977
13	L2 Clipping Aware Repeated Additive Uniform Noise Attack	0.193499982	0.588999987	0.573999971	0.603499979	0.666999996
14	Attack	0.218999982	0.564499974	0.576499969	0.59799999	0.673999995
15	Newton Fool Attack	0.201499939	**0.479499996**	**0.380499959**	0.615999997	**0.449499965**
16	Linf Deep Fool Attack	0.213499963	0.539999992	0.552999973	0.594999969	0.661499977
17	Salt And Pepper Noise Attack	0.19599998	0.546999991	0.564999968	0.593499988	0.659999996
18	L2 Deep Fool Attack	0.20599997	0.548499972	0.57249999	0.586499989	0.66049999
19	L2 Additive Gaussian Noise Attack	0.237999976	0.536999971	0.548999965	0.613999993	0.676999986
20	L2 Additive Gaussian Noise Attack	0.23149997	0.532999992	0.563499987	0.602499992	0.662999988
21	L2 Clipping Aware Additive Gaussian	0.219499946	0.532999992	0.569499969	0.57949999	0.675499976
22	Attack	0.222499967	0.539499968	0.561499983	0.572999984	0.672499985
23	Linf Additive Uniform Noise Attack	0.224499941	0.523499966	0.570499986	0.59799999	0.659499973
24	L2 Carlini Wagner Attack	0.229499936	0.532499969	0.57249999	0.597499967	0.656499982
25	FGM	0.229999959	0.544999987	0.566999972	0.599499971	0.672999978
26	FGSM	0.220999956	0.54549998	0.533499986	0.584499985	0.663999975
27	L2 PGD	0.227999985	0.522999972	0.571999967	0.573999971	0.657999992
28	Linf PGD	0.229499936	0.548999965	0.530499965	0.596999973	0.667499989
29	PGD	0.209999979	0.565999985	0.564499974	0.600499988	0.664999992

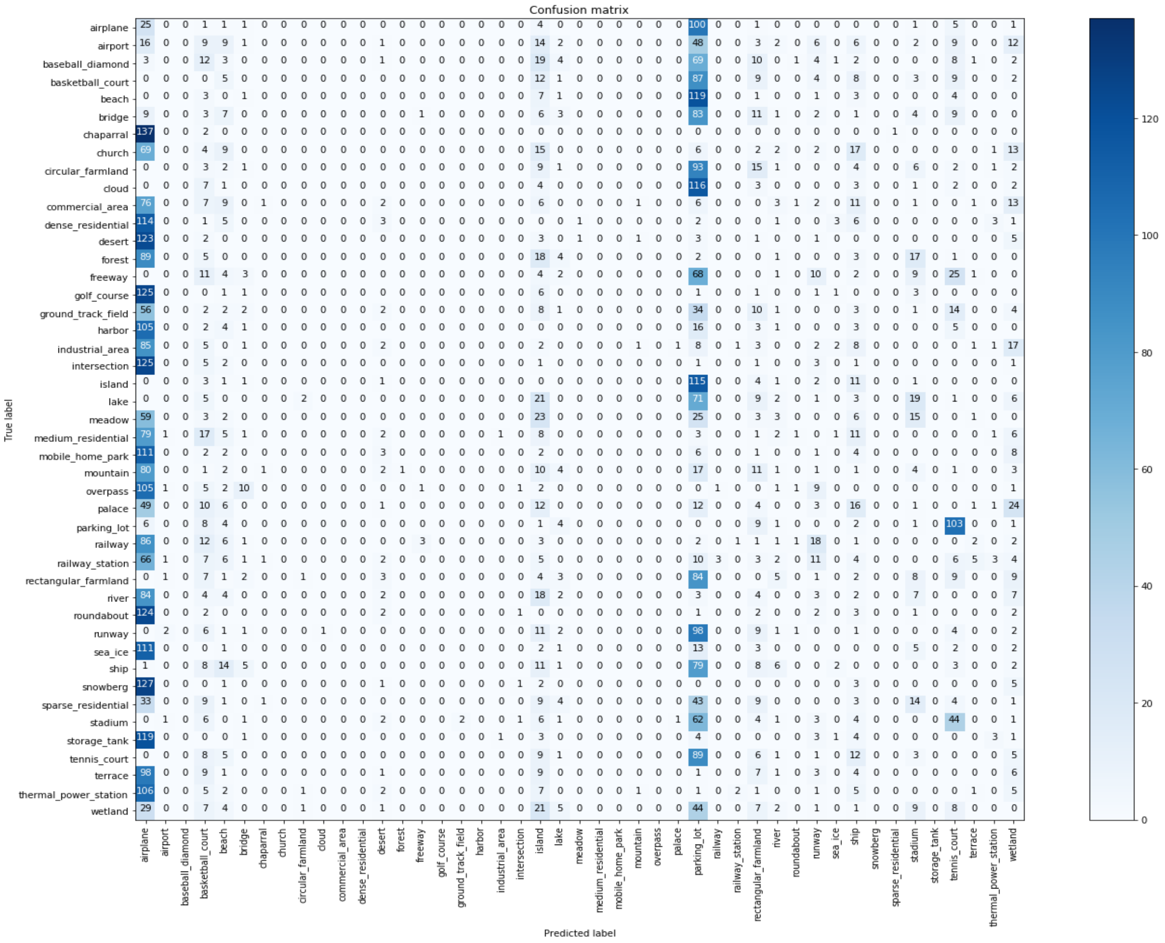
Confusion Matrix for Virtual Adversarial Attack on AlexNet



- Epsilon=1.0

	Original Accuracy	0.747058824	0.885373609	0.892686804	0.894117647	0.914149444
	Best Attack Accuracy	0.718600959	0.848807633	0.841494441	0.885055646	0.505723357
	Percentage of Performance Drop	3.809320504	4.130005188	5.734638752	1.013513243	**44.67826232**
Attack No.	**Attack Name**	**AlexNet**	**Resnet50**	**Resnet101**	**MobileNet_v2**	**DensetNet**
1	**L2 Contrast Reduction Attack**	0.744038165	0.887599364	0.896820351	0.89125596	0.91462639
2	**Virtual Adversarial Attack**	0.746263921	0.887758344	0.894117646	0.891732909	0.915103339
3	**DDNA Attack**	0.750238478	0.886645466	0.895389505	0.892368838	0.915262319
4	**L2 Projected Gradient Descent Attack**	0.748012722	0.886327505	0.892686807	0.891096979	0.915739268
5	**Linf Projected Gradient Descent Attack**	0.746740848	0.883942768	0.892209858	0.889984101	0.916375197
6	**L2 Basic Iterative Attack**	0.747694761	0.886804454	0.895707473	0.892368838	0.917329095
7	**Linf Basic Iterative Attack**	0.74244833	0.8836248	0.894753575	0.892209858	0.915580288
8	**L2 Fast Gradient Attack**	0.743720204	0.884737678	0.895707473	0.892686807	0.914467409
9	**Linf Fast Gradient Attack**	0.740381569	0.883465819	0.892050877	0.887122415	0.912718602
10	**Attack**	0.743402213	0.886804454	0.896184422	0.893640697	0.914308429
11	**Attack**	0.746581882	0.884101748	0.895548493	0.893004768	0.91478537
12	**L2 Clipping Aware Repeated Additive Gaussian Noise Attack**	0.745469004	0.884737678	0.894594595	0.893958665	0.913036563
13	**L2 Clipping Aware Repeated Additive Uniform Noise Attack**	0.745310009	0.885532595	0.893958665	0.891891889	0.91399046
14	**Attack**	0.746422887	0.888553262	0.89793323	0.889348172	0.91399046
15	**Newton Fool Attack**	**0.718600959**	**0.848807633**	**0.841494441**	**0.89062003**	**0.505723357**
16	**Linf Deep Fool Attack**	0.741176486	0.884737678	0.889507152	0.888871223	0.915103339
17	**Salt And Pepper Noise Attack**	0.747853726	0.887758344	0.884737492	0.888871223	0.915262319
18	**L2 Deep Fool Attack**	0.747853726	0.883783787	0.887421242	0.890937999	0.916693166
19	**L2 Additive Gaussian Noise Attack**	0.743402213	0.884578697	0.894117646	0.892209858	0.912400633
20	**L2 Additive Gaussian Noise Attack**	0.750397459	0.888394274	0.893163756	0.89062003	0.916057236
21	**L2 Clipping Aware Additive Gaussian Noise Attack**	0.74562797	0.888394274	0.896025434	0.891096979	0.917170115
22	**Attack**	0.74594596	0.884101748	0.897774242	0.892368838	0.916057236
23	**Linf Additive Uniform Noise Attack**	0.744992048	0.885532595	0.892368838	0.890779011	0.915421307
24	**L2 Carlini Wagner Attack**	0.747694761	0.884101748	0.895230524	0.894117646	0.915898249
25	**FGM**	0.746581882	0.887122415	0.894594595	0.89062003	0.913354531
26	**FGSM**	0.740540534	0.880286172	0.88966614	0.885055646	0.913513511
27	**L2 PGD**	0.747535765	0.887281403	0.892209858	0.891732909	0.914308429
28	**Linf PGD**	0.747853726	0.884737678	0.893958665	0.891096979	0.914467409
29	**PGD**	0.744833082	0.887440383	0.892845787	0.893640697	0.916216217

Confusion Matrix for Newton Fool Attack on DenseNet121

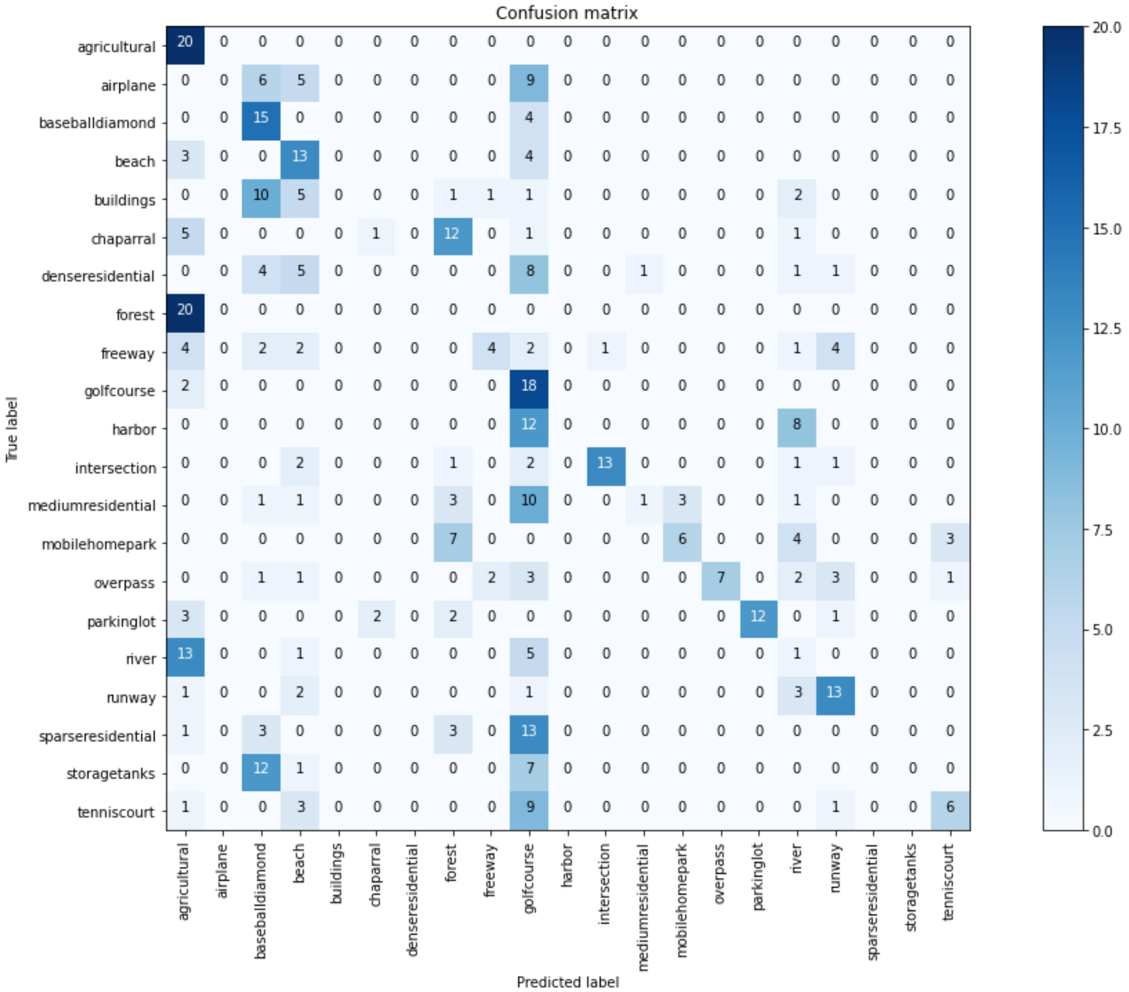


UC Merced Land Use Dataset

- Epsilon=0.05

	Original Accuracy	0.653937947	0.723150358	0.754176611	0.813842482	0.801909308
	Best Attack Accuracy	0.627684951	0.706443906	0.732696891	0.799522668	0.785202861
	Percentage of Performance Drop	**4.01460059**	2.310232169	2.84810214	1.759531398	2.083333731
Attack No.	**Attack Name**	**AlexNet**	**Resnet50**	**Resnet101**	**MobileNet_v2**	**DensetNet**
1	**L2 Contrast Reduction Attack**	0.642004758	0.708830535	0.766109779	0.809069201	0.811455846
2	**Virtual Adversarial Attack**	0.670644373	0.708830535	0.751789972	0.818615749	0.794749394
3	**DDNA Attack**	0.634844869	0.715990454	0.747016698	0.809069201	0.799522668
4	**L2 Projected Gradient Descent Attack**	**0.627684951**	0.720763713	0.756563231	0.825775653	0.801909298
5	**Linf Projected Gradient Descent Attack**	0.649164677	0.715990454	0.763723135	0.818615749	0.806682572
6	**L2 Basic Iterative Attack**	0.644391388	0.725536972	0.761336505	0.825775653	0.801909298
7	**Linf Basic Iterative Attack**	0.646778017	0.708830535	0.773269683	0.813842475	0.806682572
8	**L2 Fast Gradient Attack**	0.651551306	0.713603795	0.756563231	0.830548912	0.801909298
9	**Linf Fast Gradient Attack**	0.656324565	0.708830535	0.766109779	0.811455846	0.797136024
10	**Attack**	0.651551306	0.723150343	0.758949876	**0.799522668**	0.799522668
11	**L2 Repeated Additive Uniform Noise Attack**	0.649164677	0.706443906	0.761336505	0.806682572	0.799522668
12	**L2 Clipping Aware Repeated Additive Gaussian Noise Attack**	0.651551306	0.725536972	0.754176602	0.823389009	0.809069201
13	**L2 Clipping Aware Repeated Additive Uniform Noise Attack**	0.644391388	0.718377084	0.758949876	0.804295942	0.801909298
14	**Linf Repeated Additive Uniform Noise Attack**	0.646778017	0.723150343	0.756563231	0.818615749	0.797136024
15	**Newton Fool Attack**	0.649164677	0.711217165	0.732696891	0.816229105	0.794749394
16	**Linf Deep Fool Attack**	0.663484484	0.730310261	0.766109779	0.809069201	0.801909298
17	**Salt And Pepper Noise Attack**	0.646778017	0.715990454	0.763723135	0.818615749	0.797136024
18	**L2 Deep Fool Attack**	0.649164677	0.723150343	0.768496409	0.816229105	0.801909298
19	**L2 Additive Gaussian Noise Attack**	0.639618129	0.727923632	0.780429587	0.801909298	0.804295942
20	**L2 Additive Gaussian Noise Attack**	0.653937936	0.708830535	0.761336505	0.809069201	0.794749394
21	**L2 Clipping Aware Additive Gaussian**	0.634844869	0.727923632	0.758949876	0.806682572	0.804295942
22	**L2 Clipping Aware Additive Uniform Noise**	0.639618129	0.720763713	0.763723135	0.804295942	0.78758949
23	**Linf Additive Uniform Noise Attack**	0.658711195	**0.706443906**	0.751789972	0.811455846	0.801909298
24	**L2 Carlini Wagner Attack**	0.651551306	0.715990454	0.766109779	0.821002379	0.801909298
25	**FGM**	0.658711195	0.718377084	0.758949876	0.813842475	0.804295942
26	**FGSM**	0.63245821	0.713603795	0.773269683	0.813842475	0.797136024
27	**L2 PGD**	0.63245821	0.730310261	0.768496409	0.816229105	0.801909298
28	**Linf PGD**	0.637231499	0.708830535	0.758949876	0.809069201	**0.785202861**
29	**PGD**	0.642004758	0.725536972	**0.732696891**	0.816229105	0.806682572

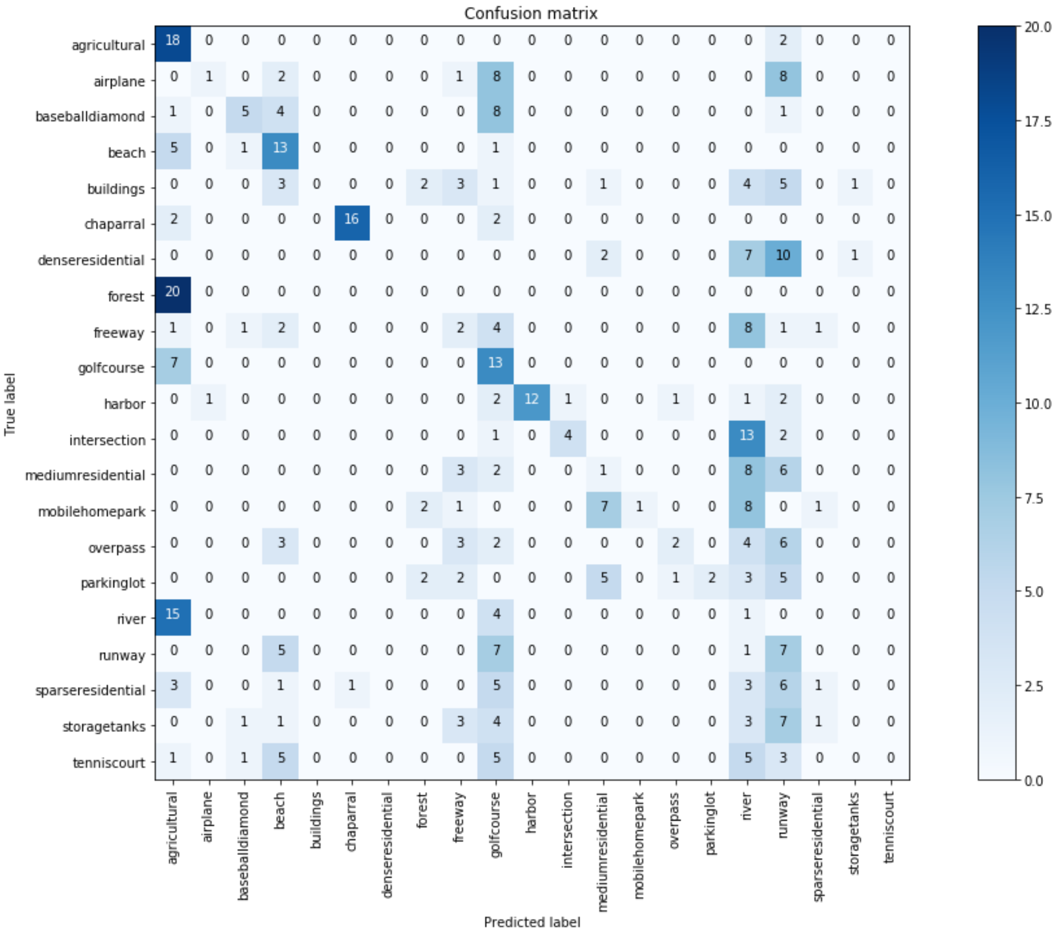
Confusion Matrix for PGD Attack on MobileNetV$_2$

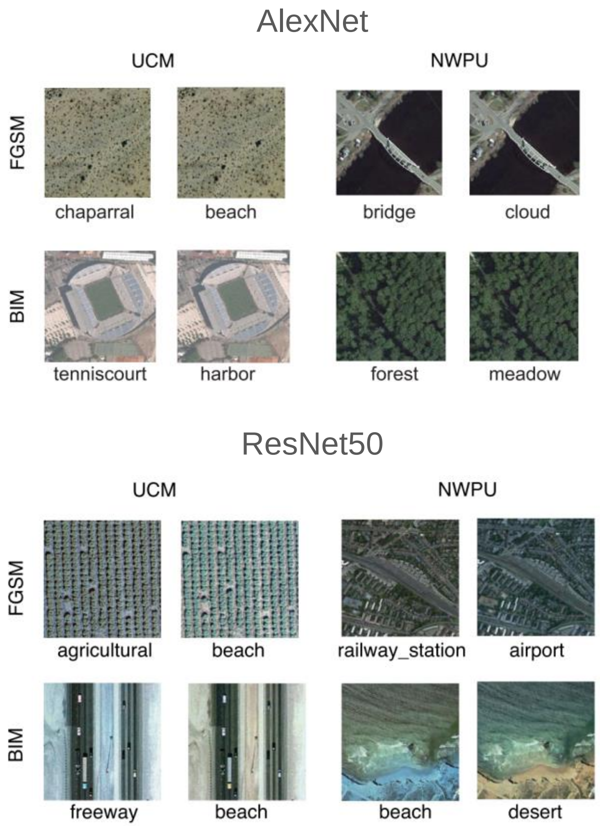


- Epsilon=1.0

	Original Accuracy	0.653937947	0.723150358	0.754176611	0.813842482	0.801909308
	Best Attack Accuracy	0.119331717	0.095465362	0.095465362	0.014319777	0.028639555
	Percentage of Performance Drop	81.7518286	86.79868425	87.34177636	**98.24047309**	96.4285793
Attack No.	Attack Name	AlexNet	Resnet50	Resnet101	MobileNet_v2	DensetNet
1	L2 Contrast Reduction Attack	0.656324565	0.723150343	0.766109779	0.556085914	0.968973747
2	Virtual Adversarial Attack	0.649164677	0.720763713	0.756563231	0.584725529	0.973747015
3	DDNA Attack	0.625298321	0.696897358	0.751789972	0.556085914	0.973747015
4	L2 Projected Gradient Descent Attack	0.651551306	0.725536972	0.766109779	0.572792351	0.973747015
5	Linf Projected Gradient Descent Attack	**0.119331717**	**0.095465362**	0.097851992	0.019093037	0.040572762
6	L2 Basic Iterative Attack	0.637231499	0.715990454	0.756563231	0.558472544	0.973747015
7	Linf Basic Iterative Attack	0.121718347	0.102625251	0.102625251	0.016706407	**0.028639555**
8	L2 Fast Gradient Attack	0.646778017	0.713603795	0.758949876	0.572792351	0.971360382
9	Linf Fast Gradient Attack	0.138424814	0.136038125	0.138424814	0.100238621	0.045346022
10	Attack	0.651551306	0.708830535	0.768496409	0.563245803	0.973747015
11	Attack	0.653937936	0.711217165	0.751789972	0.565632433	0.968973747
12	L2 Clipping Aware Repeated Additive Gaussian Noise Attack	0.642004758	0.720763713	0.751789972	0.560859174	0.971360382
13	L2 Clipping Aware Repeated Additive Uniform Noise Attack	0.658711195	0.715990454	0.766109779	0.568019062	0.971360382
14	Attack	0.36992836	0.288782775	0.403341293	0.124104977	0.97613365
15	Newton Fool Attack	0.644391388	0.723150343	0.727923632	0.520286381	0.26491642
16	Linf Deep Fool Attack	0.403341293	0.300715983	0.408114552	0.164677799	0.957040571
17	Salt And Pepper Noise Attack	0.651551306	0.720763713	0.756563231	0.565632433	0.245823383
18	L2 Deep Fool Attack	0.646778017	0.723150343	0.763723135	0.558472544	0.966587111
19	L2 Additive Gaussian Noise Attack	0.649164677	0.718377084	0.756563231	0.563245803	0.971360382
20	L2 Additive Gaussian Noise Attack	0.656324565	0.715990454	0.761336505	0.556085914	0.971360382
21	Noise Attack	0.649164677	0.725536972	0.744630069	0.572792351	0.968973747
22	Attack	0.642004758	0.713603795	0.766109779	0.553699255	0.973747015
23	Linf Additive Uniform Noise Attack	0.415274441	0.331742227	0.453460574	0.152744591	0.284009516
24	L2 Carlini Wagner Attack	0.651551306	0.720763713	0.761336505	0.551312625	0.973747015
25	FGM	0.644391388	0.715990454	0.768496409	0.57756561	0.968973747
26	FGSM	0.143198073	0.140811443	0.133651495	0.10501188	0.047732651
27	L2 PGD	0.653937936	0.718377084	0.770883054	0.572792351	0.973747015
28	Linf PGD	0.136038125	0.097851992	**0.095465362**	0.026252925	0.038186133
29	PGD	0.124104977	0.102625251	0.100238621	**0.014319777**	0.033412874

Confusion Matrix for L_2-Projected Gradient Descent Attack on AlexNet





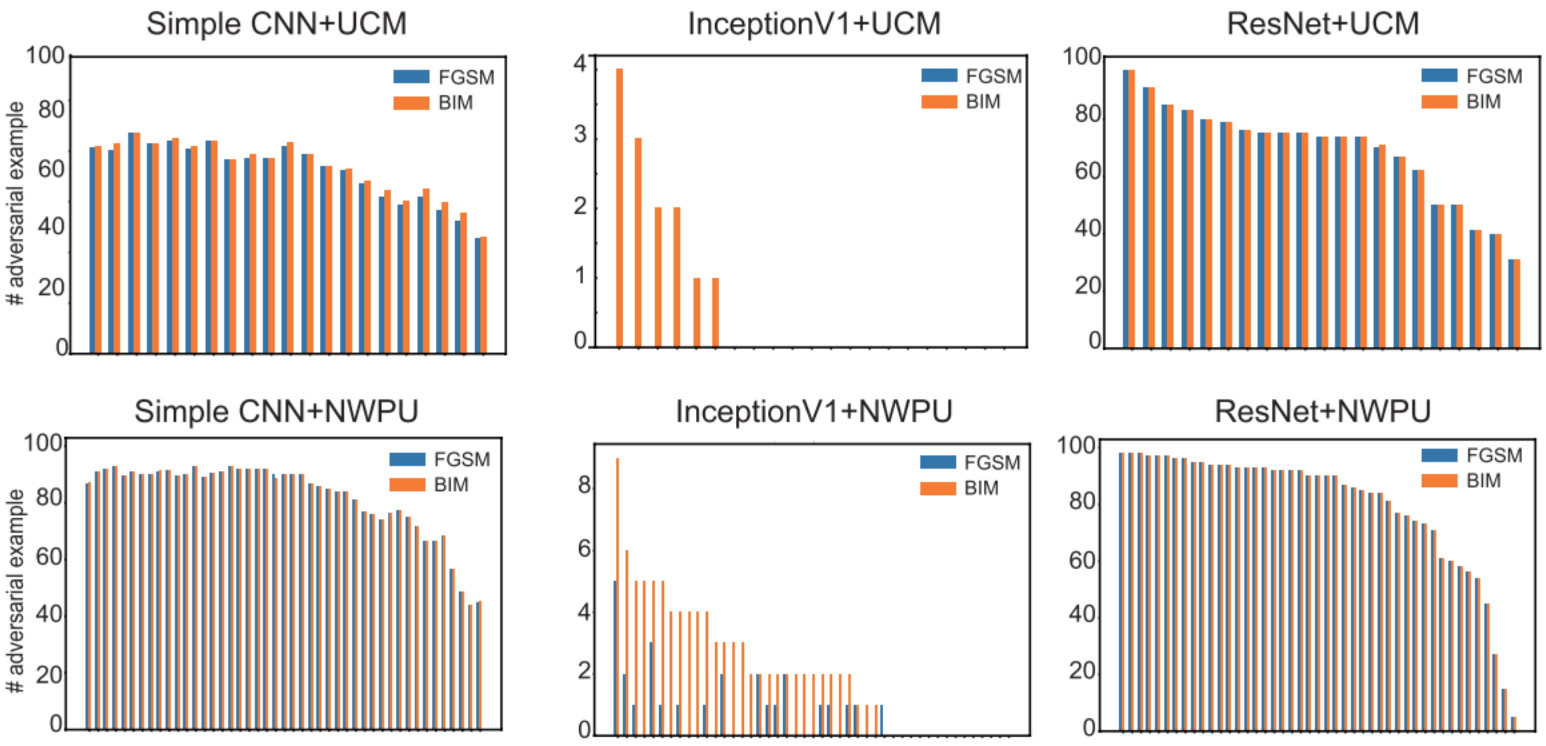
Demonstration of adversarial examples. Under different datasets, models and attack algorithms, we show adversarial examples of attacked RSIs. Each group is represented in pairs. The left image represents the original input image, and the right image represents the corresponding adversarial example. We find that the classes of the adversarial examples are misclassified.

Observations

- In the experiment, our results show that CNNs of RSI scene classification are also vulnerable to adversarial examples, and some of them have a fooling rate of over 80%.
- The model vulnerability to the adversarial example is multifactorial, and are affected by the architecture of CNNs and the type of RSI dataset.
- The result shows the vulnerability of several mainstream CNNs and demonstrates the importance of adversarial examples in RSI scene classification
- We find that adversarial examples of RSIs have a property of attack selectivity, which means that the classes of adversarial examples are not random and usually focus on a few specific classes.

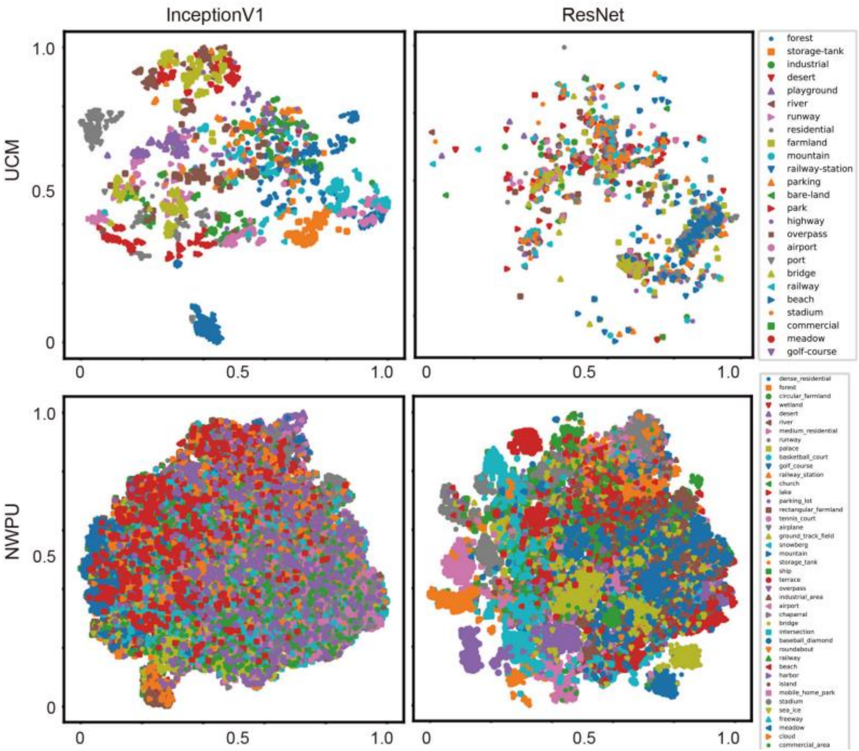
IMPACT OF THE RSI MODEL:

RSI scene classification model structure is the first factor that can affect the adversarial examples. Therefore, we calculate the adversarial examples obtained by all models in each class, as shown in Figure. We find that differences in the model structures have a large impact on the adversarial examples.



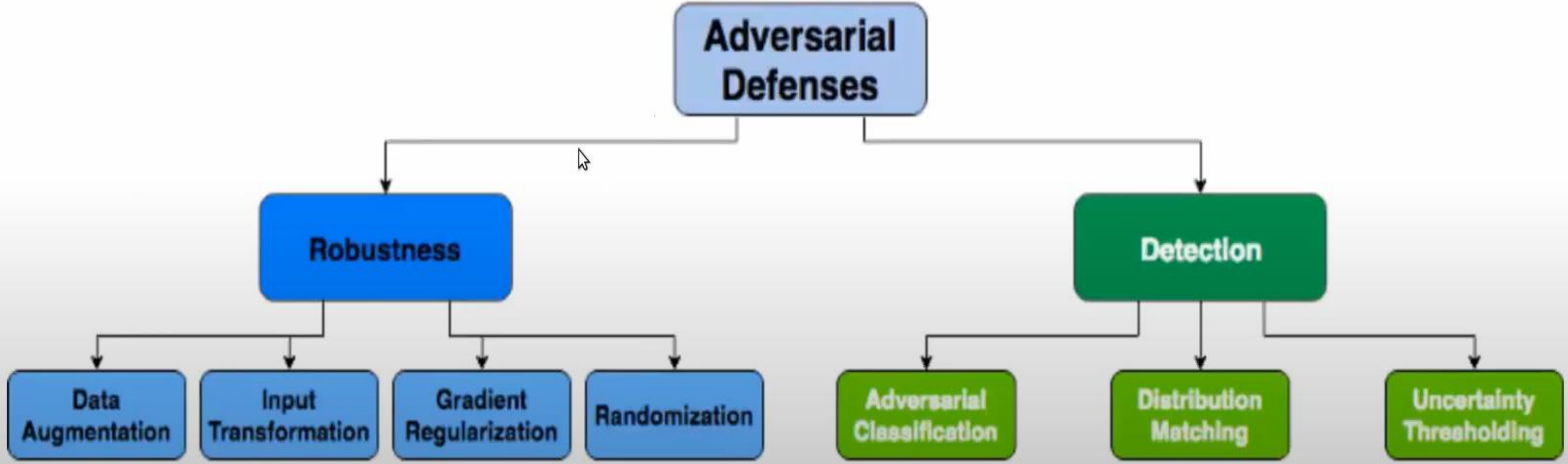
IMPACT OF THE RSI DATASET:

We find that differences in the training datasets also affect the model vulnerability of RSI scene classification systems. From Table 3, the fooling rates of the two attack models for CNN models trained on UCM datasets exceed 80%, whereas the fooling rates of CNN models trained on the NWPU is less than 20%. This result is mainly determined by the number of features in the training dataset. The UCM dataset has 21 classes, and the total number of images is 2,100. NWPU has 45, and the total numbers of images are 31,500, i.e., their contents are more substantial than those of UCM.

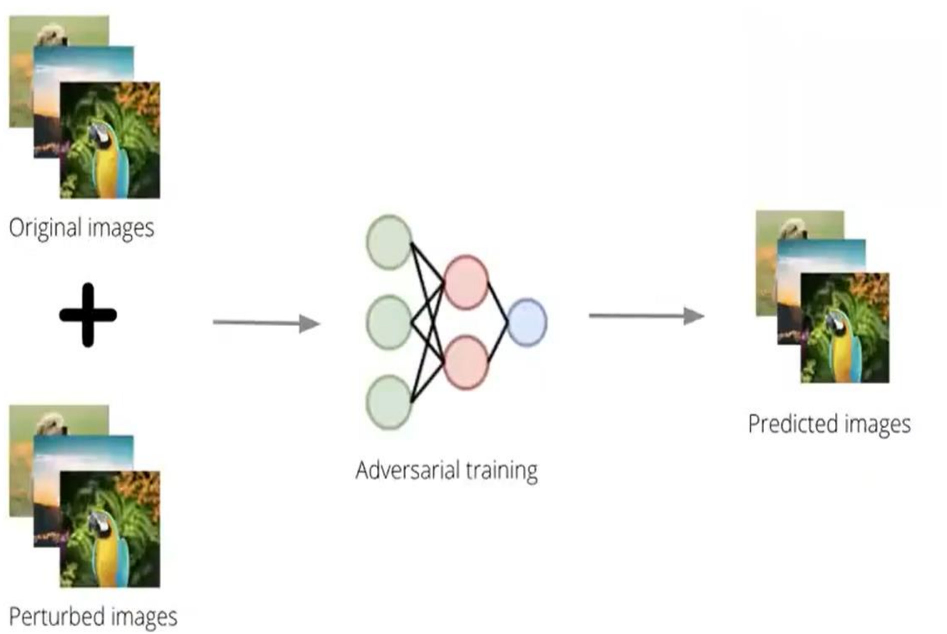


How can we defend against adversarial attacks?

1. Detect whether an input is adversarial or not
2. Modify the input so that it is no longer adversarial in nature



Adversarial Training



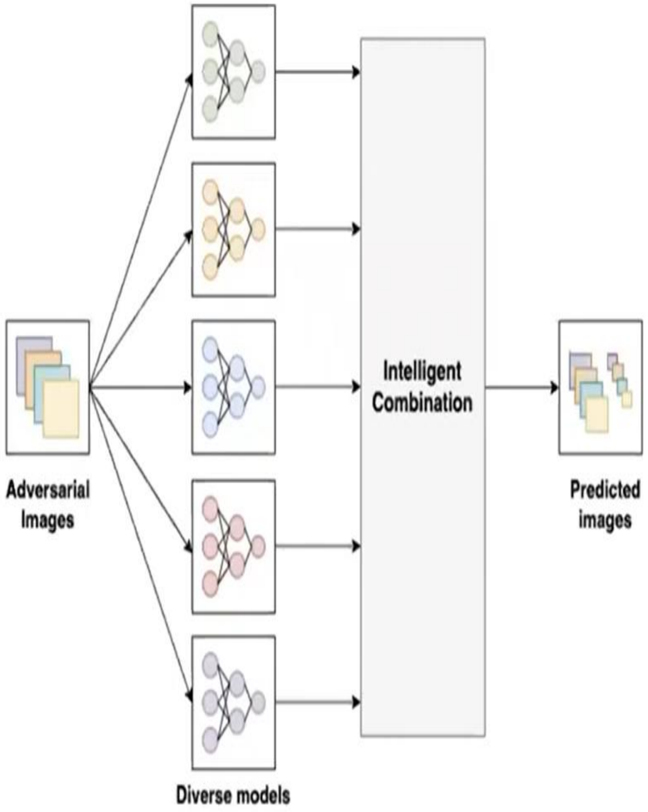
- Generate **adversarial data** based on clean data
- Use both clean data and adversarial data during training

Challenges:

- Model will learn the perturbation from adversarial data
- The accuracy of model on clean data will decrease after adversarial training

Can we make a stronger defense?

- Have multiple defense networks instead of one
- Ensure these networks are all vulnerable to the same attack, therefore are diverse
- Adversarially train these and then, perform SAP on them



Random Initialization

The attacker's noise cannot succeed in all networks

Pros

- Offer some diversities: different initialization lead to different local minimum
- Easy to implement and easy to train
- Effectively defense adversarial attack

Cons

- Requires more computational power
- Not diverse enough
- Affect the accuracy of normal image

Epochs	Dev acc	MV acc	Average attck acc	MV when attack
4	71.54%	80.90%	64.07%	73.69
10	81.31%	88.65%	64.29%	71.95
15	83.90%	90.67%	63.65%	69.79
20	85.70%	91.90%	63.26%	68.99
30	87.43%	92.61%	63.97%	69.67

Stochastic Activation Pruning

- Similar to dropout
- Prune nodes with a probability proportional to their magnitude of activations in each layer
- Scale the remaining nodes' activations to maintain the dynamic range of activations for the input to the subsequent layer

Advantages

- Does not reduce accuracy
- Saves computation: no retraining required

Algorithm 3: Stochastic Activation Pruning (SAP)

Data: Input datum x, neural network with n layers, with i^{th} layer having weight matrix W_i, non-linearity i and number of samples to be drawn r^i

Result: New activation map

1 Calculate activation vector for layer i, $h^i \leftarrow \phi^i\left(W^i h^{i-1}\right)$;

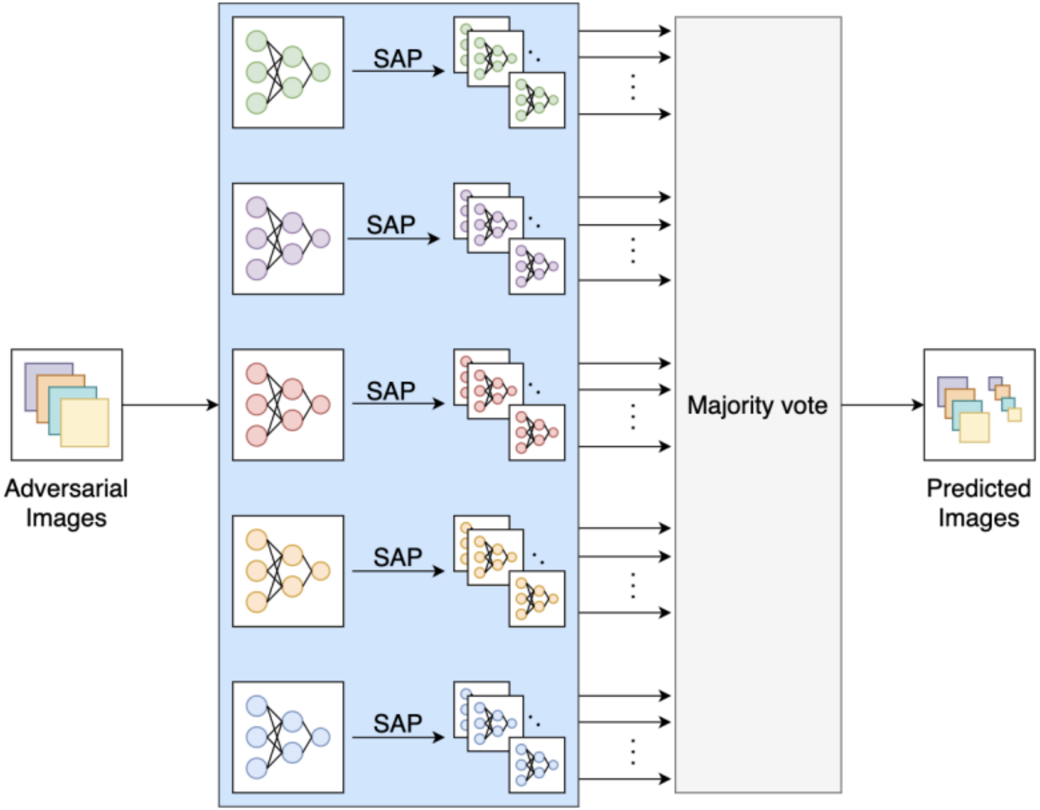
2 Normalize activations on to the simplex with multinomial probability distribution,

$$p_j^i \leftarrow \frac{\left|\left(h^i\right)_j\right|}{\sum_{k=1}^{a^i}\left|\left(h^i\right)_k\right|}, \forall j \in \left\{1, \ldots, a^i\right\};$$

3 Draw a set of indices of activations to be kept and prune the left based on the distribution;

4 Scale up survived activations, $\left(h^i\right)_j \leftarrow \frac{\left(h^i\right)_j}{1-\left(1-p_j^i\right)^{r^i}}$.

Multi-SAP Adversarial Defense



Defense Strategy – 1

- In our first defense strategy, we use one base network, here, the base ResNet18 model and adversarially train it.
- Different models are produced every time SAP is applied, owing to the variation in sampling at each iteration.
- Therefore, we applied 50 random seeds and obtained 50 differently pruned models of the base adversarially trained model. We then obtained a majority vote among the classification results from each of these 50 models.

Defense Strategy – 2

- In our second defense strategy, we utilize 5 different base networks that are obtained using different splits of training data and weight initializations.
- These 5 networks are then adversarially trained. Now, using 10 random seeds, we create 10 diverse networks from each of these five adversarially trained networks, thereby producing 50 diverse networks.
- These combined with the 5 non-SAP base adversarial networks produced a total of 55 networks.

Multi-SAP Defense Results

Defense Strategy – 1

L∞ TORCH ATTACKS	Single Non adversarially trained model	Single adversarially trained model	Adversarially trained our SAP defense
BIM	46%	55.70%	**56.70%**
RFGSM	33.70%	50.10%	**51.50%**
APGD	41.30%	54.60%	**55.00%**
TPGD	36%	53.80%	**54.60%**
FFGSM	37.90%	54.80%	**55.13%**
MI-FGSM	41.60%	55.20%	**56%**

Defense Strategy – 2

L∞ TORCH ATTACKS	Single Non-adversarially trained model	Single Adversarially trained model	Adversarially trained our SAP defense
BIM	46%	55.70%	**62.40%**
RFGSM	33.70%	50.10%	**56.10%**
APGD	41.30%	54.60%	**60.70%**
TPGD	36%	53.80%	**61.20%**
FFGSM	37.90%	54.80%	**60.60%**
MI-FGSM	41.60%	55.20%	**61.30%**

Defense using GANs

- We denoise the adversarial image by adding learnt noise to it. (i.e. “Counter the adversary”)
- This noise distribution is learnt using a Generative Adversarial Network, where the generates the anti-adversarial noise.
- We experiment with various architectures and hyperparameters

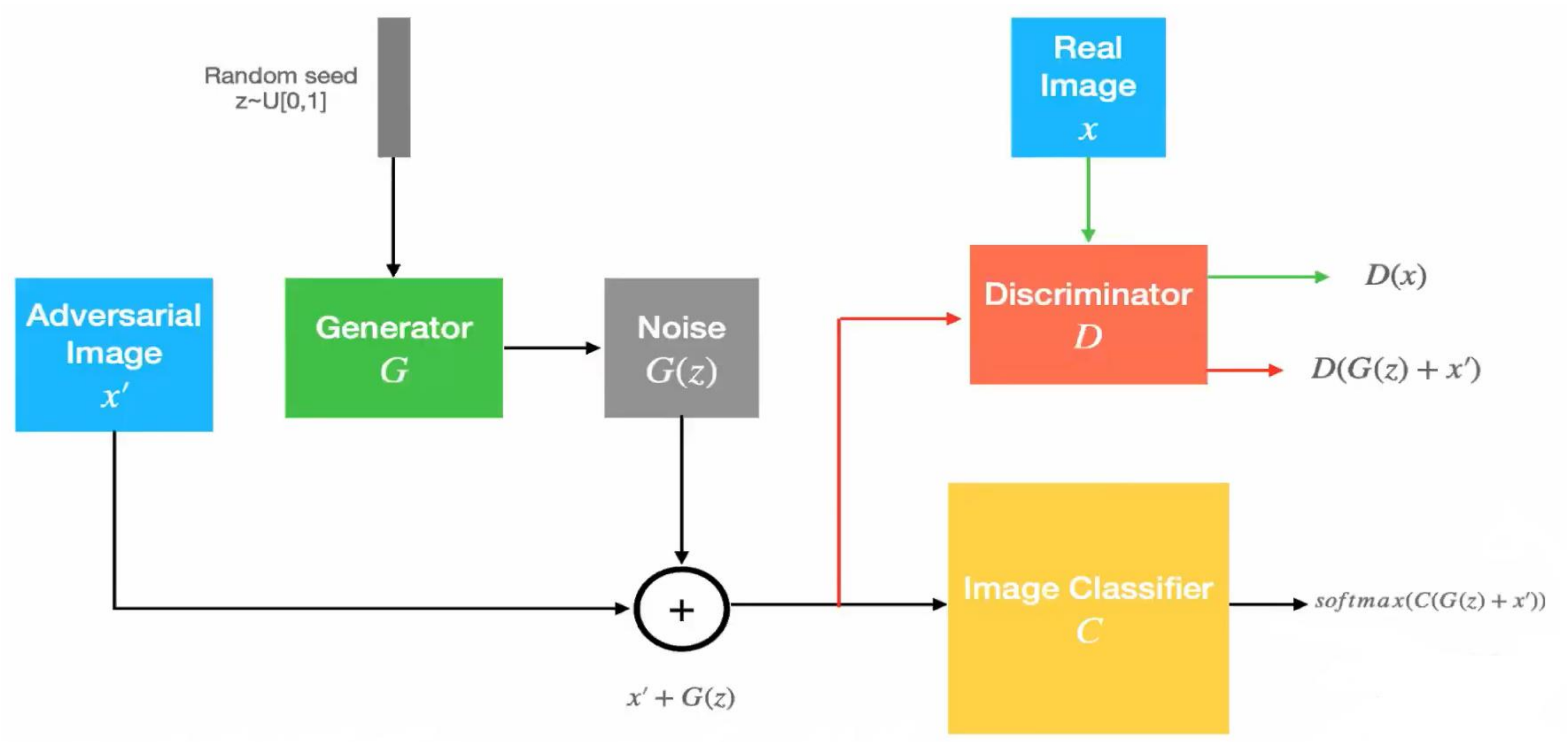
We know that …

Image + Adversarial Noise = Adversarial Image

Our idea …

Image + Adversarial Noise + ***Learned Noise*** = Normal Image

GAN Architecture



GAN Defense Results

Experiment	Linear+BCE	Linear+MSE	Conv+MSE	Conv+BCE	Linear + WGAN-GP	DefenseGAN
Adv Classifier accuracy	1.03%	1.03%	1.03%	1.03%	1.03%	6%
Adv Defense Accuracy	14.88%	11.87%	13.03%	12.47%	10.70%	37%
Real Classifier accuracy	92.21%	92.21%	92.12%	92.14%	92.21%	80%
Real Defense Accuracy	31.01%	28.38%	35.15%	35.27%	30.87%	42%

Conclusion

- Random initialization add more diversities which make DNNs more robust towards adversarial samples
- Stochastic Activation Pruning like methods also improve the defense by stochastic pruning activation and reduce computation by sharing weights and bias with the pretrained model
- Multi-SAP combined with adversarial training outperforms PGD adversarial training, one of the most powerful defenses against L-inf attacks
- Improved accuracy by 6-7% against multiple L-inf attacks
- Using a counter noise for negating the adversarial noise is helpful to some extent. Fully connected layers might work better with images than CNNs for generating noise

References

- https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9146660
- https://ieeexplore.ieee.org/document/9339955
- https://www.hindawi.com/journals/scn/2021/6663028/
- https://ieeexplore.ieee.org/document/9119167
- https://arxiv.org/pdf/1805.10997
- https://www.tensorflow.org/datasets/catalog/resisc45
- https://ieeexplore.ieee.org/document/7891544
- https://www.tensorflow.org/datasets/catalog/uc_merced https://faculty.ucmerced.edu/snewsam/papers/Zhu_SIGSPATIAL15_LandUseClassification.pdf
- https://github.com/bethgelab/foolbox*(GitHub)*
- https://foolbox.jonasrauber.de
- https://arxiv.org/abs/1707.04131
- Ren, K., Zheng, T., Qin, Z. and Liu, X., 2020. Adversarial attacks and defenses in deep learning.Engineering.
- Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, et al. Intriguing propertiesof neural networks. 2013. arXiv:1312.6199.
- Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014. arXiv:1412.6572.
- Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world. 2016. arXiv:1607.02533.
- Dong Y, Liao F, Pang T, Su H, Zhu J, Hu X, et al. Boosting adversarial attacks with momentum.In: Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition; 2018 Jun 18–23; Salt Lake City, UT, USA; 2018. p. 9185–193.
- Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: Proceedings ofthe 2017 IEEE Symposium on Security and Privacy; 2017 May 22–26; San Jose, CA, USA; 2017. p.39–57.\
- Zheng T, Chen C, Ren K. Distributionally adversarial attack. 2018. arXiv:1808.05537.
- Moosavi-Dezfooli SM, Fawzi A, Fawzi O, Frossard P. Universal adversarial perturbations. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition; 2017 Jul 21–26; Honolulu, HI, USA; 2017. p. 1765–73.
- Xiao C, Li B, Zhu JY, He W, Liu M, Song D. Generating adversarial examples with adversarialnetworks. 2018. arXiv:1801.02610.
- Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprintarXiv:1706.06083 (2017).
- Jan Hendrik Metzen, Tim Genewein, et al. "On Detecting Adversarial Perturbations. " arXiv preprintarXiv:1702.04267 [stat.ML] (2017).
- Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-gan: Protecting classifiers againstadversarial attacks using generative models." arXiv preprint arXiv:1805.06605 (2018).
- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems.2014.
- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarialexamples." arXiv preprint arXiv:1412.6572 (2014
- Alec Radford, Luke Metz, Soumith Chintala, "Unsupervised Representation Learning with Deep Convolu-tional Generative Adversarial Networks". ICLR 2016
- Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprintarXiv:1411.1784 (2014).
- Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale imagerecognition." arXiv preprint arXiv:1409.1556 (2014).
- https://github.com/chengyangfu/pytorch-vgg-cifar10Mao, Xudong, et al. "Least squares generative adversarial networks." Proceedings of the IEEE internationalconference on computer vision. 2017.
- Martin Arjovsky, Soumith Chintala, and Leon Bottou. “Wasserstein GAN”. In:ArXiV abs/1701.07875(2017) https://arxiv.org/abs/1701.07875
- Nicholas Carlini, David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten DetectionMethods. 2017.
- Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, Dawn Song.(Revised version: 14 Feb 2019)Generating Adversarial Examples with Adversarial Networks arXiv:1801.02610v5 [cs.CR]
- Puneet Mangla, Surgan Jandial, Sakshi Varshney, Vineeth N Balasubramanian.(Revised version: 23 Dec2019) AdvGAN++ : Harnessing latent layers for adversary generation. arXiv:1908.00706v2 [cs.CV]
- Yann LeCun, Corinna Cortes, and Christopher JC Burges. 1998. The MNIST database of handwritten digits.(1998)
- Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

- Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprintarXiv:1706.06083 (2017).

- an Hendrik Metzen, Tim Genewein, et al. "On Detecting Adversarial Perturbations. " arXiv preprintarXiv:1702.04267 [stat.ML] (2017).

- Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-gan: Protecting classifiers againstadversarial attacks using generative models." arXiv preprint arXiv:1805.06605 (2018).
- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems.2014.
- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarialexamples." arXiv preprint arXiv:1412.6572 (2014

- Alec Radford, Luke Metz, Soumith Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". ICLR 2016

Thank You!

Questions?