भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Adversarial Attacks on Aerial Scene Classification

Shorya Sharma
ss118@iitbbs.ac.in

Abstract

Remote sensing image (RSI) scene classification is the foundation and important technology of ground object detection, land use management and geographic analysis. During recent years, convolutionalneural networks (CNNs) have achieved significant success and are widely applied in RSI scene classification. However, crafted images that serve as adversarial examples can potentially fool CNNs with high confidence and are hard for human eyes to interpret. For the increasing security and robust requirements of RSI sceneclassification, the adversarial example problem poses a serious problem for the classification results derivedfrom systems using CNN models, which has not been fully recognized by previous research. In this study, 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 (NWPU-RESIC45 and UC Merced Land Use Dataset) to fool CNNs (i.e., AlexNet, ResNet50, ResNet101, MobileNetV_2 and DenseNet121). We also explored the transferability properties of Transferable Sparse Adversarial Attack (a bench-mark black box adversarial attack with high transferability rate under L_0 conditions) in this study. Attack selectivity reveals potential classes of adversarial examples and provides insightsinto the design of defensive algorithms in future research.

1. Introduction

With the advancement of remote sensing technology,the automatic interpretation of remote sensing images (RSIs)has greatly improved [1]–[4]. RSIs with higher resolution have led to the wide use of RSI scene classification systems in crop classification [5]–[7], forest resource surveys [8]–[10], land cover classification [11], [12], building detection [13], [14] and other fields [15], [16]. Automatic interpretations of RSIs with robust and high accuracy can deliver high economic efficiency [17]–[19].

A good image classification algorithm can effectively extract target features and quickly locate new targets, which is the key to RSI interpretation. Therefore, the variabilityin features can be massive among different classification tasks. most traditional RSI interpretation algorithms design the hand-craft features for different applications, such as road detection [20], [21], vegetation measurement [22], drought monitoring [23], etc. [24]–[26]. 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 [27]–[29]. And most traditional classification algorithms have a low classification accuracy and calculation inefficiency, which makes it difficult to meet thesystem requirements.

In recent years, deep learning has achieved remarkable progress in the field of computer vision [30]. Many excellent algorithms, especially convolutional neural networks (CNNs), have been introduced into the RSI scene classification systems [5], [11], [15]. Chen *et al.* [31] used a single-layer autoencoder (AE) and multi-layer stacked autoencoder (SAE) to learn the features of RSIs and proposed a classification method using the extracted spatial features. This method performs better than do support vector machines (SVM)and K-nearest neighbor (KNN) classification models. Chen *et al.* [32] proposed a deep belief network (DBN)-based image classification framework that combined the spectral and spatial features of RSIs and used a CNN model toextract robust features from images, thereby greatly reducing the number of hyperparameters. To alleviate the overfitting problem of CNNs, Cheng *et al.* [33] added the metric learning regularization term to the CNN model by optimizing the discriminant objective

function. This approach efficiently reduced classification error. Zhang *et al.* [34] proposed a gradient boosting random convolution network that reused the weight of each CNN model and reduced the parameters, thusmaking feature extraction more efficient. Chaib *et al.* [35] used the CNN model in feature extraction and discriminant correlation analysis (DCA) for data fusion. These RSI sceneclassification models perform much better in terms of accuracy than traditional RSIs and in the efficiency of RSI sceneclassification achieved using high-performance computing. Meanwhile, previous research has also demonstrated the excellent feature extraction abilities of CNNs.

However, CNNs have many limitations and still haveroom for improvements, especially concerning security problems [36]–[39]. Szegedy *et al.* [40] find that a small adversarial perturbation can be achieved on a trained model with agradient algorithm. This happens when we deliberately add noises to the input images to fool the CNN classifier and prompt it to make wrong predictions with high confidence. The modified images are called adversarial examples.

The problem of adversarial examples exists in various application areas of deep learning [41]. The classification problem of RSIs presents a great security risk, especially forapplications in the military and automatic driving fields [42].As shown in Figure 1, in military applications, an attacker can generate an adversarial perturbation for a target such as an airplane to camouflage the object using a physical object.The adversarial example of the target can easily escape the detection by the military target detection systems. In this case, the originally robust target would be wrongly classified as the wrong class with high confidence. State-of-the-art RSI sceneclassification systems depend heavily on CNNs to extract features from the target. Unlike RGB images, RSIs have unique properties, such as spectra, bands, etc. In addition, RSIs are usually obtained from the nadir perspective, without the foreground and back views of natural images. Therefore,it is necessary to study the characteristics of adversarial examples in RSI scene classification systems.

In the experiment, we train several CNN models withthe widest application in RSI scene classification systems.In these high-accuracy CNNs, we use a variety of attack algorithms to generate different adversarial examples. The result shows the vulnerability of several mainstream CNNs and demonstrates the importance of adversarial examples in RSI scene classification. Furthermore, we find that the fundamental issues related to adversarial examples of RSIs are the model vulnerability and attack selectivity. This means that different RSI scene classification CNNs have differ- ent security characteristics, so the cost to obtain adversarial examples varies. We also find that attack selectivity is related to the model structure and the training data, producing a clustered pattern of errors in generating adversarial examples. The misclassified adversarial examples are highly similar to the correct original scenes. These properties also reflect the characteristics of adversarial examples in RSI sceneclassification.

The main contributions of this article are listed as follows:

- We implemented white-box adversarial attacks using 5 different neural network architectures on different aerial scene classification datasets under two different configurations: eps=0.0005 and eps=1.0.
- We tabulated our results and inferred the best-attack on each dataset using a particular network architecture. Confusion Matrix were also plotted for evaluation purposes.
- We implemented the Transferable Sparse Adversarial Attack (TSAA) and deployed it on our dataset to infer the transferability of this black-box attack on our datasets using different neural network architectures.
- We implemented state-of-the-art black box attacks on our datasets using Adversarial Attack libraries.

2. Related work

In recent years, CNNs have been applied in many tasks in the field of remote sensing, such as landmark classification [43], land use classification [44] and change detection [45]. CNNs have produced outstanding results in these areas. However, most of the research focuses on how to improve the model accuracy and design good CNN structures [46]–[48], and insufficient attention has been placed onthe adversarial example problem of RSI scene classificationsystems. Since the adversarial example was proposed by Szegedy *et al.* [40], the security of deep learning has been the subject of widespread discussion.

The adversarial example refers to an input image formed by adding a special adversarial perturbation, which results in the model producing a false output with high confidence. Generally, a small adversarial perturbation is not perceived by the human eye. Goodfellow *et al.* [37] stated that adversarialexamples in CNNs are produced due to the linear operationsof the model in a high-dimensional space. For feature vectoroperations in a high-dimensional space, small perturbations can cause substantial errors by reducing the robustness of themodel. This problem not only is encountered in computer vision tasks but also exists in many fields such as natural language processing and sentiment analysis [49]. It is well known that RSIs consist of high-dimensional data, so the problem of adversarial examples in RSI scene classificationsystems is also unavoidable.

The attack algorithms of adversarial examples are also diverse. According to the number of iterations, attack algorithms can be divided into the one-step attack and iterative methods. These attack algorithms are gradient-based optimization methods that try to maximize the loss of the objective function. The fast gradient notation (FGSM) method is a one-step attack proposed by Goodfellow *et al.* [37], which first computes the gradient direction of the model loss function and subsequently increases the

value of the loss function by adding a small adversarial perturbation in that direction. Thus, the prediction result of the model deviates from the real class. This attack algorithm is simple, but the fooling rate of deceived models is low. Therefore, an iterative method named basic iterative method (BIM) is proposed [38]. The algorithm is an extension of the FGSM, and with multiple iterations, it attempts to add noise in each step to increase the value of the loss. In each iteration, the changed image pixels are controlled within a certain neighborhood of the original picture. Different from the FGSM, the BIM algorithm can effectively improve the fooling rate of attacks. FGSM and BIM are the two most widely applied attack algorithms.

Attack algorithms can also be divided into other types. Attack algorithms can be divided into white-box attacks and black-box attacks [36] according to the model information acquired by the attacker. In a white-box attack, the attacker knows the type of CNN, the number of layers, and the optimization method used for training and can obtain the training data and even discern the hyperparameters of the model. Attackers can launch additional targeted attacks based on the details of the model. In contrast to white-box attacks, attackers in black-box attacks are unaware of the model details. They can use only the model application background and outputs to analyze the vulnerability of the model. According to the characteristics of the attack algorithm, Prem-latha *et al.* [50] and Chakraborty *et al.* [51] classified attack scenarios into three categories: evasive attacks, poisoned attacks and exploratory attacks. Attackers in an evasive attack attempt to avoid detection by the system by constructing malicious input. In this scenario, the attacker cannot influence the training data of the model. The attacker in a poisoned attack scenario attempts to inject constructed malicious data to poi-son the model during the training process. In the exploratory attack scenario, the attacker cannot affect the training dataset or the details of the model and acquires knowledge of the learning algorithms and training data by testing the response of the model to the input data, thereby constructing effective adversarial examples. According to whether the attack class is clear, attack algorithms are also divided into targeted attacks and untargeted attacks [36]. Targeted attacks attempt to make the model classify the adversarial example as a specific class, whereas untargeted attacks do not care about the prediction labels of adversarial examples. Untargeted attacks require only the model to misclassify data that were originally correctly classified.

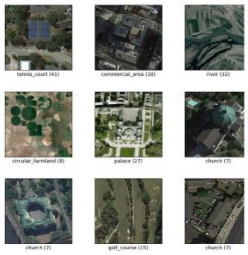
Above all, adversarial example attacks might occur during training and testing of a model. The type of attack is related to the model details and training data. The type of attack algorithm depends on the setting of the attack scenario. For example, FGSM and BIM can be both white-box and black-box attack algorithms. It depends on whether the information (e.g., model structure, hyperparameters) of the attacked model is known before the attack. Similarly, FGSM and BIM can also be targeted and untargeted attacks, depending on whether the image is misclassified into a specific class. Under different attack types, the object function of the attack algorithm changes. For example, the FGSM targeted attack algorithm only needs to change the loss function into the optimization of a specific class. In our research, we use all white-box, untargeted attack algorithms.

Currently, RSI scene classification involves many CNN-based applications that might also have adversarial example problems. These problems could have highly serious consequences for the application of RSI scene classification. Moreover, since RSIs have spatial and spectral properties, they are different from the RGB images. Therefore, the adversarial example problem is of great importance in RSI scene classification.

3. Data

Two Datasets were used:

- 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 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



NWPU-RESIC 45

UC Merced Dataset

4. Method

This section gives a brief introduction of CNNs and the most widely used robust CNN model structure in RSI scene classification. Subsequently, it explains how to use attack algorithms to generate an adversarial example of an RSI.

A. CONVOLUTIONAL NEURAL NETWORKS

Current RSI scene classification systems have better classification capabilities, and they are mostly based on CNNs. CNNs usually consist of convolutional layers, pooling layers, and fully connected layers [52]. Each layer has a different function. Through these structures, RSIs are continuously reduced in dimension, and semantic features are gradually abstracted. Eventually, the model implements the classification of the RSIs.

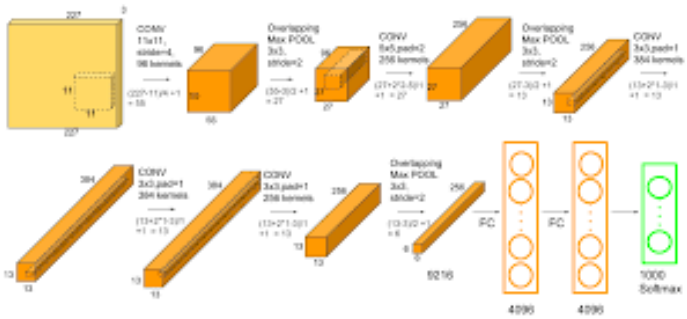
Pooling operations can reduce the dimensions of the RSI to reduce the computational complexity and improve the generalization ability of the model. Finally, A^l denotes the feature maps produced after the convolution and pooling operations. When the feature maps are fed into the fully connected layer, all of the feature maps need to be flattened and trans- formed into vectors. Finally, A classifier can predict a class y based on the final outputs. When the prediction is not correct, we use the cross-entropy loss function to compute the distance between the predicted results and the real labels. The model can be updated by the backpropagation algorithm [52], which aids the model in reducing the loss value. The model gradually converges in the process of iteration

1) AlexNet

In 2012, *Alex Krizhevesky* and others proposed a deeper and wider CNN model compared to LeNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [7]. AlexNet achieved state-of-the-art recognition accuracy against all the traditional machine learning and computer vision approaches. It was a significant breakthrough in the field of machine learning and computer vision for visual recognition and classification tasks and is the point in history where interest in deep learning increased rapidly.

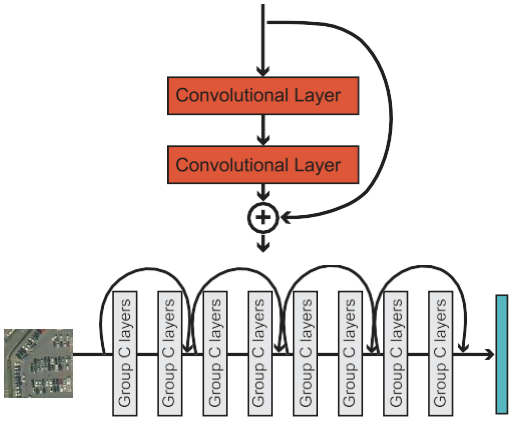
The architecture of AlexNet is shown in Fig. 14. The first convolutional layer performs convolution and max pooling with Local Response Normalization (LRN) where 96 different receptive filters are used that are 11×11 in size. The max pooling operations are performed with 3×3 filters with a stride size of 2. The same operations are performed in the second layer with 5×5 filters. 3×3 filters are used in the third, fourth, and fifth convolutional layers with 384, 384, and 296 feature maps respectively. Two fully connected (FC) layers are used with dropout followed by a Softmax layer at the end. Two networks with similar structure and the same number of feature maps are trained in parallel for this model. Two new concepts, Local Response Normalization (LRN) and dropout, are introduced in this network. LRN can be applied in two different ways: first applying on single channel or feature maps, where an N×N patch is selected from same feature map and normalized based one the neighborhood values. Second, LRN can be applied across the channels or feature maps (neighborhood along the third dimension but a single pixel or location).

AlexNet has 3 convolution layers and 2 fully connected layers. When processing the ImageNet dataset, the total number of parameters for AlexNet can be calculated as follows for the first layer: input samples are 224×224×3, filters (kernels or masks) or a receptive field that has a size 11, the stride is 4, and the output of the first convolution layer is 55×55×96. According to the equations in section 3.1.4, we can calculate that this first layer has 290400 (55×55×96) neurons and 364 (11 ×11×3 = 363 + 1 bias) weights. The parameters for the first convolution layer are 290400×364 = 105,705,600. Table II shows the number of parameters for each layer in millions. The total number of weights and MACs for the whole network are 61M and 724M respectively



2) Resnet

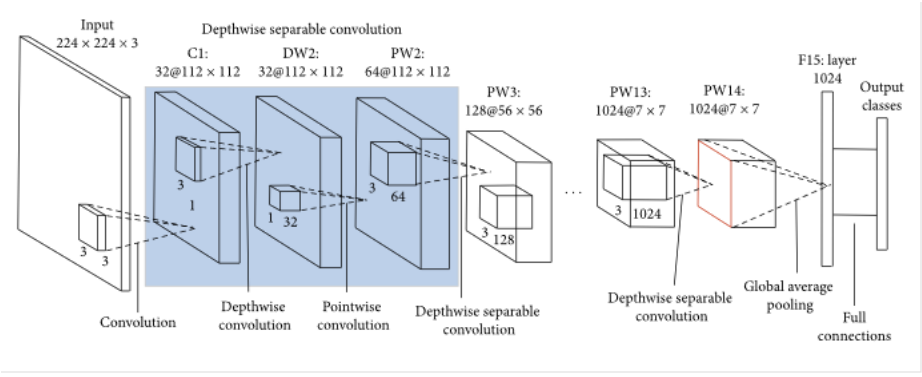
Another model structure, ResNet [57], won the 2015 ILSVRcompetition, as shown in Figure 3(b). The model takes the depth to extreme values, and for the first time, the model achieved a depth of 152 layers. ResNet50 is the 50-layer version of ResNet. To train deeper structures, ResNet introduced a residual module, as shown in Figure 3(a), which implements the delivery of information between layers through a shortcut connection. Without adding parameters, this model can directly place shallow features into the upper layer, thus greatly increasing the training speed of the model. ResNet achieved a milestone in the ImageNet competition [56] and was the first model to surpass human classification in the ImageNet classification task. Moreover, as the main backbone of the detection model, ResNet has achieved good results for the VOC Pascal [58] and COCO [59] datasets and in other detection tasks [60]. Many RSI detection applications are based on ResNet [61].



3) MobileNet

MobileNet model is designed to be used in mobile applications, and it is TensorFlow's first mobile computer vision model. MobileNet uses depthwise separable convolutions**.** It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations: Depthwise Convolution and Pointwise Convolution. MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

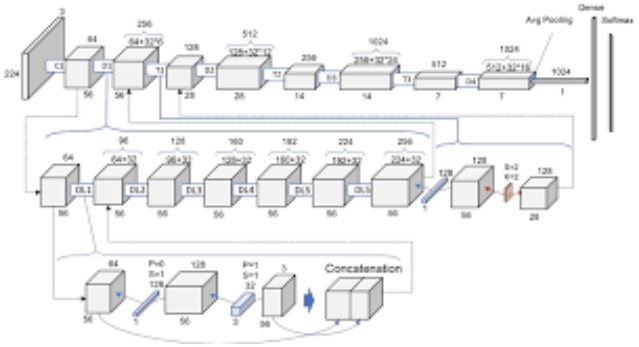


4) DenseNet

In a normal feedforward network, an output from a layer is passed on to only the next occurring layer and as the network goes deep it would have access to only the higher-level features but not those from lover levels closer to input. In Densenet every subsequent layer receives input from all the previous layers in the chain as can be observed from above figure. All these values are concatenated so that information is not lost. The concatenated feature map is passed through a composite function consisting of Batch Normalization, Relu and 3x3 Convolution. The output is then passed on to every

subsequent layer and the process is repeated.

But simple concatenation can lead to very huge number of parameters in a deep network. So the architecture is divided into multiple dense blocks each separated by a block called transition block. In dense block the information flows in the way described in above paragraph. Transition block is the place where downsampling occurs to prevent blowing up. The transition layer consists of batch normalization layers followed by 1x1 convolution layer which is followed by 2x2 average pooling layer.



B. ATTACK ALGORITHM

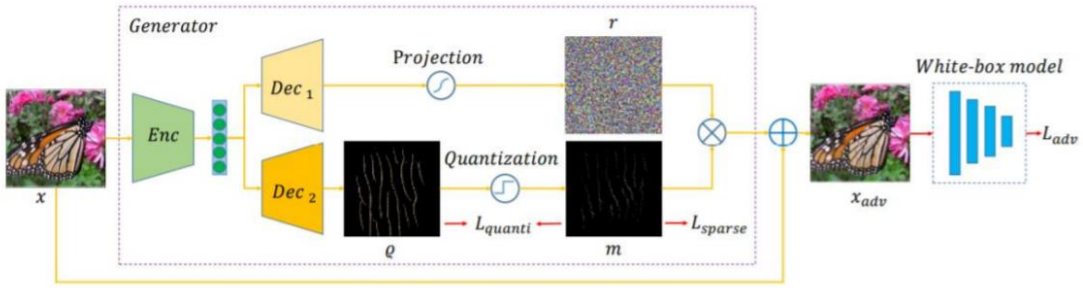
These excellent models still cannot escape the problem of adversarial examples. The goal of the adversarial example is primarily to make the model misclassify the minimally changed image when the original image can be classified correctly. The attack algorithms of adversarial examples arealso applicable to RSIs.

1) White-Box Adversarial Attacks

- Gradient-Based Attacks: Gradient Attack, Gradient Sign Attack (FGSM), Iterative Gradient Attack, Iterative Gradient Sign Attack, DeepFool L2 Attack, DeepFool L∞ Attack, L-BFGS Attack, SLSQP Attack, Jacobian-Based Saliency Map Attack
- Score-Based Attacks: Single Pixel Attack, Local Search Attack, Approximate L-BFGS Attack
- Decision-Based Attacks: Boundary Attack, Pointwise Attack, Additive Uniform Noise Attack, Additive Gaussian Noise Attack, Salt and Pepper Noise Attack, Contrast Reduction Attack, Gaussian Blur Attack, Precomputed Images Attack

2) Transferable Sparse Adversarial Attack (Black-Box Adversarial Attack)

TSAA is a sparse adversarial attack based on the L0 norm constraint, which can succeed by only modifying a few pixels of an image. Prior sparse attack methods achieve a low transferability under the black-box protocol because they methods rely on the target model's accurate gradient information or its approximation, causing the generated adversarial examples overfitting the target model. TSAA is a trainable generator-based architecture which tends to alleviate the overfitting issue by learning to translate a benign image into an adversarial example and thus efficiently craft transferable sparse adversarial examples



3) Fooling Rate

To evaluate the effects of attack algorithms, we introduce the definition of the fooling rate [36], which reflects the model'sratio of misclassified images to attack images and can be represented as follows,

$$\text{fooling rate} = \frac{N_{misclassification}}{N_{attack}} \times 100\%,$$

where $N_{misclassification}$ represents the number of misclassified images after the attack. Similarly, N_{attack} represents the number of images attacked. The fooling rate represents how many adversarial examples can successfully fool the model. It canbe used to evaluate both models and attack algorithms. For the same model, a strong attack algorithm can obtain more adversarial examples that can fool the model with a higher fooling rate. For the same attack algorithm, a high fooling rate indicates that the model has poor robustness.

5. Experiments

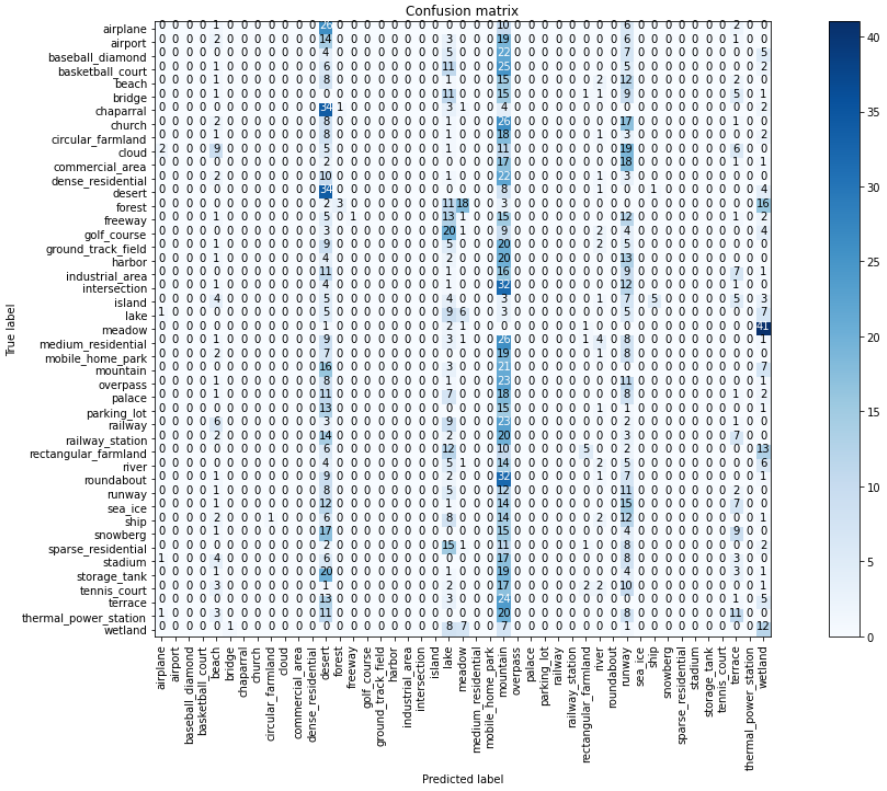
In this section, we first use the white-box attack algorithms to attack multiple converged RSI scene classification models under different datasets. Second, we analyze the transferability of TSAA. The attack object of the adversarial example is a modelwith high accuracy that has been converged. In the experiment, we use three commonly used RSI datasets, including UC Merced Land Use [62], NWPU-RESISC45 [63]. For all datasets, we choose 80% as a training set and the remaining 20% as a test set during the training progress. For deep learning models, we choose AlexNet, ResNet50, Resnet101, MobileNetV_2 and DenseNet121 which are widely used in RSI scene classification systems. The experimental platform is based on Ubuntu 16.04, with a 12 × 3.20 GHz Core i5 CPU, Tesla M60 GPU and PyTorch Framework. Cross-Entropy Function was used as the loss function with Adam Optimizer being selected for optimizing the weights. Confusion Matrix was used as the evaluation Metrics along with accuracy.

A. White-Box Attacks

1) NWPU-RESIC45

- Epsilon: 0.0005

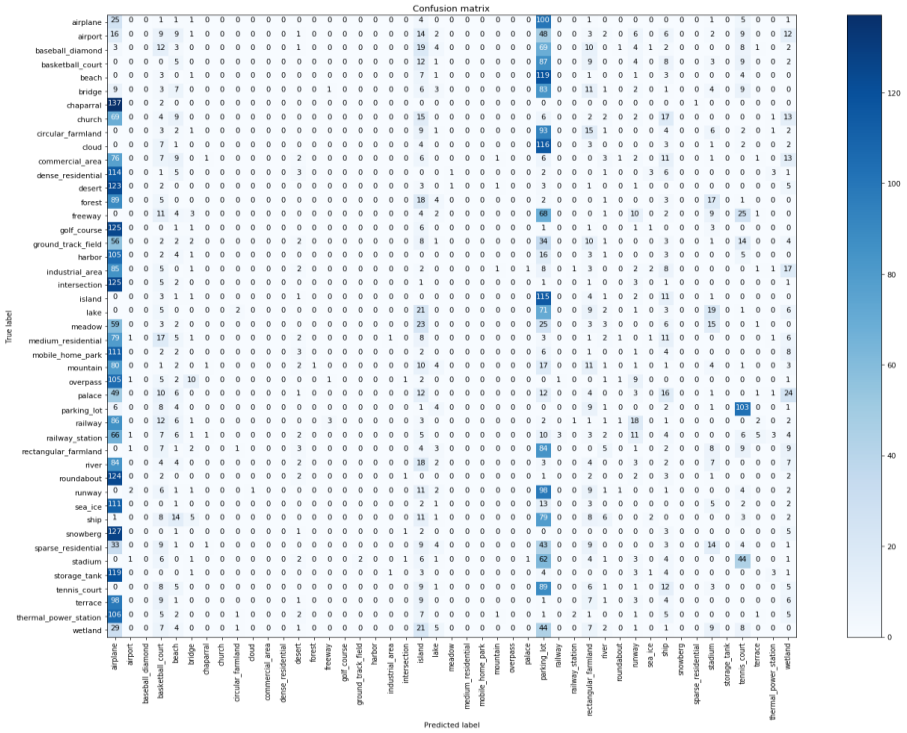
	Original Accuracy	0.747058824	0.885373609	0.892686804	0.894117647	0.914149444
	Best Attack Accuracy	0.185	0.479	0.381	0.573	0.449
	Percentage of Performance Drop	**75.23622047**	45.89854552	57.31985752	35.91447368	50.88330435
Attack No.	**Attack Name**	**AlexNet**	**Resnet50**	**Resnet101**	**MobileNet_v2**	**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

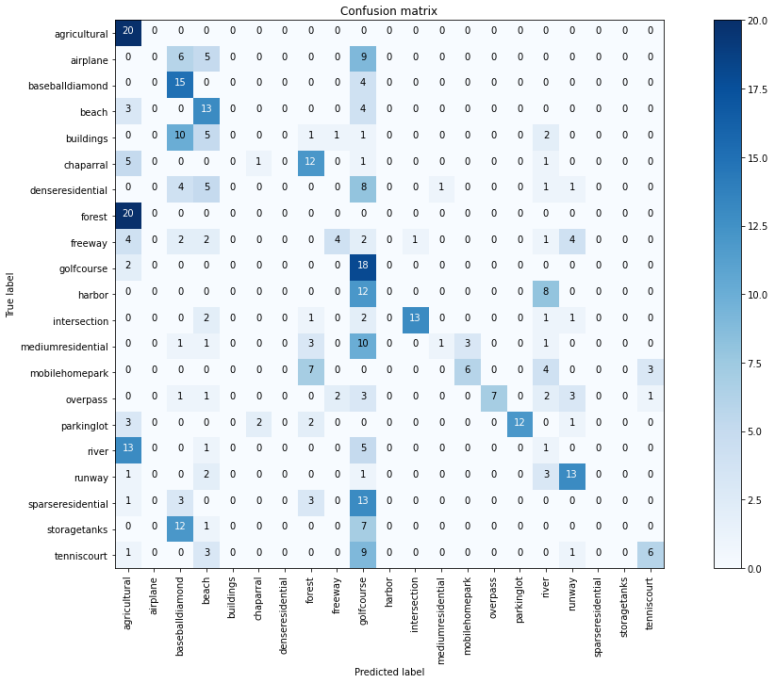


Confusion Matrix for Newton Fool Attack on DenseNet121

2) UC Merced Land Use Dataset

- Epsilon: 0.0005

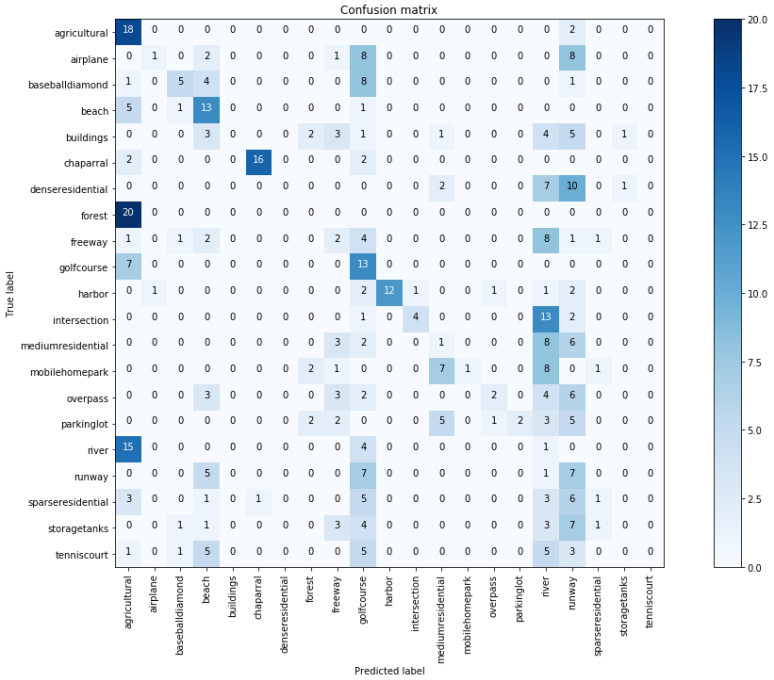
	Original Accuracy	0.653937947	0.723150358	0.754176611	0.813842482	0.801909308
	Best Attack Accuracy	0.119331717	0.095465362	0.095465362	0.014319777	0.028639555
	Percentage of Performance Drop	81.7518286	86.79868425	87.34177636	**98.24047309**	96.4285793
Attack No.	**Attack Name**	**AlexNet**	**Resnet50**	**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 PGD Attack on MobileNetV_2

- Epsilon: 1.0

	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



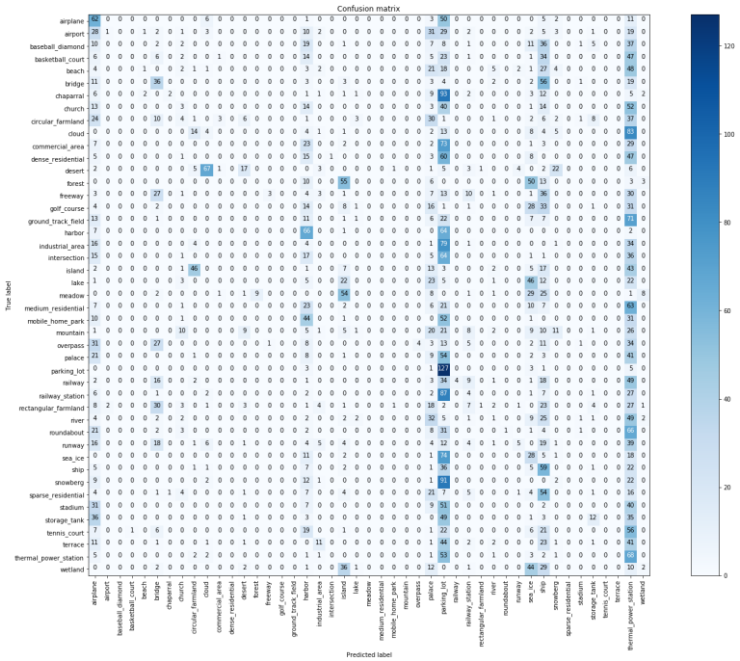
Confusion Matrix for L_2-Projected Gradient Descent Attack on AlexNet

B. Transferable Sparse Adversarial Attack (Black-Box Adversarial Attack)

1) NWPU-RESIC45

- Epsilon: 0.0005

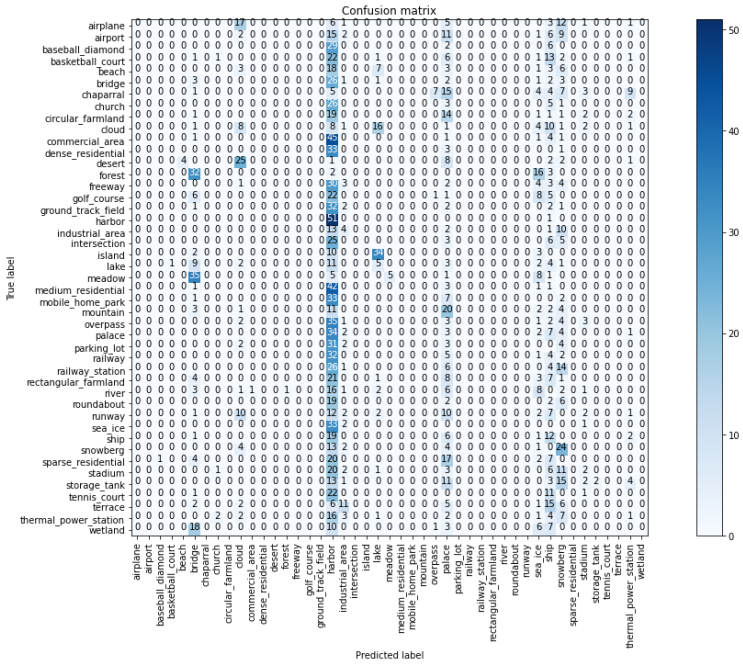
Source(Generator)	Model	L0-Norm	Time (ms)	Adversarial Accuracy (%)	Fooling Rate (%)	Robust Accuracy (%)
AlexNet	AlexNet*	0	0.534025853	17.774	75.262	24.73767886
	ResNet50	0	0.759529916	8.362	90.7	9.300476948
	ResNet101	0	1.037881371	10.874	90.509	9.491255962
	MobileNet_v2	0	1.476634477	12.258	66.121	33.87917329
	DenseNet121	0	1.144518761	5.231	73.466	26.53418124
ResNet50	AlexNet	0	1.502175172	17.488	75.66	24.34022258
	ResNet50*	0	1.260088889	8.458	90.604	9.395866455
	ResNet101	0	1.066947552	10.874	90.143	9.856915739
	MobileNet_v2	0	1.487173147	12.385	65.421	34.57869634
	DenseNet121	0	1.182302762	5.262	73.339	26.66136725
ResNet101	AlexNet	0	0.568486966	17.758	75.517	24.48330684
	ResNet50	0	1.283827344	8.426	90.7	9.300476948
	ResNet101*	0	1.169546458	10.859	90.477	9.523052464
	MobileNet_v2	0	1.401777715	12.544	65.803	34.19713831
	DenseNet121	0	1.221956092	5.39	73.577	26.42289348
MobileNet_V2	AlexNet	0	1.503463959	17.742	75.342	24.6581876
	ResNet50	0	1.246488947	8.347	90.541	9.459459459
	ResNet101	0	1.051601545	10.97	90.334	9.666136725
	MobileNet_V2*	0	1.446312221	12.496	65.135	34.86486486
	DenseNet121	0	1.244934792	5.31	73.752	26.24801272
DenseNet121	AlexNet	0	1.461738092	:17.361	76.169	23.83147854
	ResNet50	0	1.221987856	8.394	90.477	9.523052464
	ResNet101	0	1.088902909	10.938	90.604	9.395866455
	MobileNet_v2	0	1.514493832	12.544	65.215	34.78537361
	DenseNet121*	0	1.389218761	5.482	71.294	28.706



Confusion Matrix for ResNet101 Generator Attack on ResNet50

- Epsilon: 1.0

Source(Generator)	Model	L0-Norm	Time (ms)	Adversarial Accuracy (%)	Fooling Rate (%)	Robust Accuracy (%)
AlexNet	AlexNet*	50063.5586	0.421248555	8.15	59.55	40.45
	ResNet50	50055.8125	0.399045587	5.5	95.85	4.15
	ResNet101	50052.793	0.400970578	5.65	72.7	27.3
	MobileNet_v2	50048.793	0.383505344	10	68	32
	DenseNet121	50043.8477	0.384764671	3.1	76.2	23.8
ResNet50	AlexNet	0	0.424089432	8.45	58.85	41.15
	ResNet50*	0	1.145537853	5	96.65	3.55
	ResNet101	0	0.383425951	5.45	75.05	24.95
	MobileNet_v2	0	0.367376566	10.3	67.2	32.8
	DenseNet121	0	0.378295541	3.75	76.7	23.3
ResNet101	AlexNet	1147.067	0.422928572	7.3	60.05	39.95
	ResNet50	1176.4121	0.381159544	6	96.7	3.3
	ResNet101*	1165.7941	0.383220553	5.3	85.35	14.65
	MobileNet_v2	1131.2896	0.373892903	9.65	70.15	29.85
	DenseNet121	1155.4196	0.380117893	3.1	77.6	22.4
MobileNet_V2	AlexNet	0	0.434672475	8.3	58.35	41.65
	ResNet50	0	0.392568231	5.75	95.45	4.55
	ResNet101	0	0.406144023	4.8	75.35	24.65
	MobileNet_V2*	0	0.387378454	11.15	67.05	32.95
	DenseNet121	0	0.391313195	3.6	77.85	22.15
DenseNet121	AlexNet	2604.8662	0.421614766	9.5	59	41
	ResNet50	2622.8987	0.371679664	6.15	94.65	5.35
	ResNet101	2583.0205	0.374443054	5.2	70.35	29.65
	MobileNet_v2	2582.7012	0.367733002	10.15	72.2	27.8
	DenseNet121*	2649.0557	0.378239989	3.8	96.45	3.55

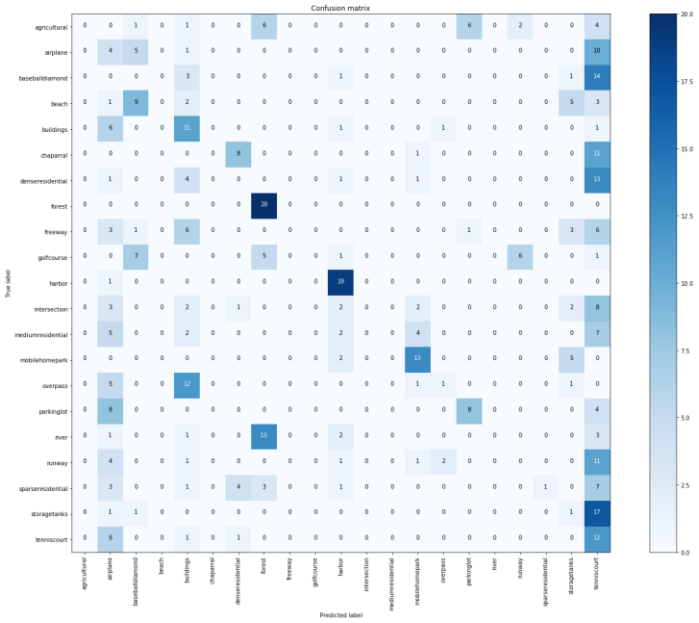


Confusion Matrix for ResNet101 Generator Attack on ResNet50

2) UC Merced Land Use Dataset

- Epsilon: 0.0005

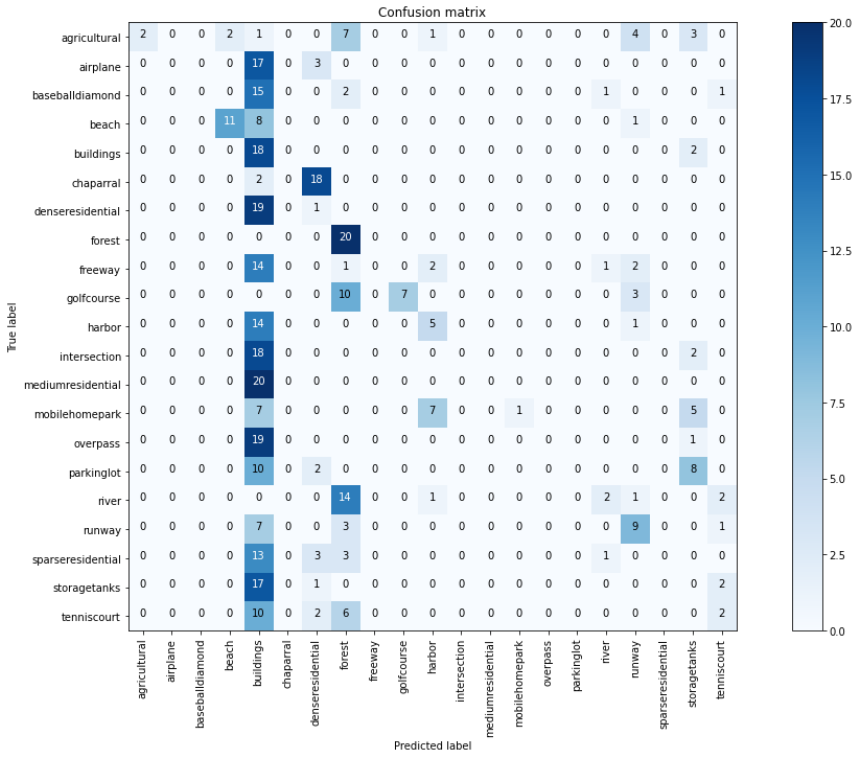
Source(Generator)	Model	L0-Norm	Time (ms)	Adversarial Accuracy (%)	Fooling Rate (%)	Robust Accuracy (%)
AlexNet	AlexNet*	9626.8281	0.693851553	36.277	48.926	51.07398568
	ResNet50	9661.6709	0.862086872	19.093	61.098	38.90214797
	ResNet101	9586.6016	1.251717456	23.866	78.998	21.00238663
	MobileNet_v2	9645.3301	1.358397536	11.456	16.706	83.29355609
	DenseNet121	9592.4707	1.187203892	31.981	69.69	30.31026253
ResNet50	AlexNet	2228.3818	1.555173095	36.516	47.494	52.50596659
	ResNet50*	2222.7686	1.353261579	19.332	58.95	41.05011933
	ResNet101	2241.4011	1.165145906	22.912	77.088	22.91169451
	MobileNet_v2	2234.9023	1.548439904	11.933	17.184	82.81622912
	DenseNet121	2228.0718	1.223687056	31.265	69.212	30.7875895
ResNet101	AlexNet	0	1.687926154	35.8	47.971	52.02863962
	ResNet50	0	1.32609381	19.809	58.473	41.5274463
	ResNet101*	0	1.067505247	23.628	78.043	21.95704057
	MobileNet_v2	0	1.533233465	11.933	14.797	85.20286396
	DenseNet121	0	1.185852474	31.265	69.928	30.07159905
MobileNet_V2	AlexNet	45019.4922	32.73339431	41.527	50.119	49.88066826
	ResNet50	45008.9688	32.4927025	17.9	63.246	36.75417661
	ResNet101	45020.0703	32.33842008	21.48	84.726	15.27446301
	MobileNet_V2*	45011.9414	32.47196635	11.695	25.776	74.22434368
	DenseNet121	45013.6094	32.65738886	29.594	67.542	32.45823389
DenseNet121	AlexNet	0	1.631933067	34.845	47.494	52.50596659
	ResNet50	0	1.28238241	19.093	61.098	38.90214797
	ResNet101	0	1.044955629	23.866	78.52	21.4797136
	MobileNet_v2	0	1.469535304	10.97	17.422	82.57756563
	DenseNet121*	0	1.244934792	5.31	69.215	30.78591291



Confusion Matrix for MobileNetV$_2$ Generator Attack on ResNet101

- Epsilon: 1.0

Source(Generator)	Model	L0-Norm	Time (ms)	Adversarial Accuracy (%)	Fooling Rate (%)	Robust Accuracy (%)
AlexNet	AlexNet*	9662.1025	0.285544088	15.752	67.542	32.45823389
	ResNet50	9664.9951	0.27224329	12.411	56.563	43.43675418
	ResNet101	9618.9404	0.389571406	19.809	75.179	24.82100239
	MobileNet_v2	9673.0264	0.372455911	33.174	52.506	47.49403341
	DenseNet121	9670.1074	0.375579251	14.32	46.778	53.22195704
ResNet50	AlexNet	2219.1479	0.245995055	35.561	53.938	46.06205251
	ResNet50*	2228.4966	1.061151022	18.616	93.556	6.443914081
	ResNet101	2235.4153	0.461475377	29.117	83.771	16.22911695
	MobileNet_v2	2226.2747	0.390163754	48.21	48.926	51.07398568
	DenseNet121	2238.2029	0.365764827	18.377	78.282	21.71837709
ResNet101	AlexNet	0	0.232030782	35.322	51.313	48.68735084
	ResNet50	0	0.262025433	19.809	61.098	38.90214797
	ResNet101*	0	2.595364905	22.912	79.47	20.52505967
	MobileNet_v2	0	0.244205493	45.823	46.778	53.22195704
	DenseNet121	0	0.502975573	24.344	65.155	34.84486874
MobileNet_V2	AlexNet	0	1.132587829	35.561	46.778	53.22195704
	ResNet50	0	1.151256174	20.048	59.666	40.33412888
	ResNet101	0	1.171488182	24.105	78.998	21.00238663
	MobileNet_V2*	0	1.260259555	46.53	46.301	53.69928401
	DenseNet121	0	1.124749605	23.866	64.439	35.56085919
DenseNet121	AlexNet	0	0.4171244	35.561	48.926	51.07398568
	ResNet50	0	0.404797192	19.332	58.95	41.05011933
	ResNet101	0	0.422845308	24.105	80.191	19.80906921
	MobileNet_v2	0	0.412999019	45.585	42.959	57.04057279
	DenseNet121*	0	0.425844033	24.582	65.394	34.60620525



Confusion Matrix for ResNet50 Generator Attack on ResNet50

6. Experiment Observations

Following Observations were made on the basis of the experiments conducted in the above study:

- On NWPU-RESIC45, the highest drop in accuracy was observed when Virtual Adversarial Attack **(75.23%)** and Newton Fool Attack **(44.67%)** were deployed under eps= 0.0005 and eps=1.0 respectively. to create adversarial examples on the AlexNet and DenseNet respectively.
- When TSAA was executed on NWPU dataset, highest fooling rate was **9.30%** when the generator was trained on ResNet101 architecture and the adversarial examples were created using the ResNet50 model under eps=0.0005 and **3.3%** when the generator was trained on ResNet101 architecture and the adversarial examples were created using the ResNet50 model under eps=1.0
- On UC Merced Dataset, the highest drop in accuracy was observed when L_2 Projected Gradient Descent Attack **(4.01%)** and PGD Attack **(98.24%)** were deployed under eps= 0.0005 and eps=1.0 respectively. to create adversarial examples on the AlexNet and MobileNetV_2 respectively.
- When TSAA was executed on UC Merced dataset, highest fooling rate was **15.27%** when the generator was trained on MobileNetV_2 architecture and the adversarial examples were created using the ResNet101 model under eps=0.0005 and **6.44%** when the generator was trained on ResNet50 architecture and the adversarial examples were created using the ResNet50 model under eps=1.0

7. Experiment Discussion

In this experiment, we verify that adversarial examples also exist in RSI scene classification systems. We find that the use of training datasets with different scales and different model structures has an impact on model vulnerability; even the same model exhibits different model vulnerabilities for different attack algorithms.

Adversarial examples of RSIs also have attack selectivity.From a geometric point of view, the RSI dataset is represented as a data point in the high-dimensional space in the CNN model, and the adversarial example is deviated from the boundary of the original cluster by adding small perturbations to the data points, which causes the models to misclassifyit. Therefore, data points at a cluster boundary are more likely to be successfully attacked than data points

near the center of the cluster. In addition, we find that most of the adversarial classes are beaches. Beaches might have special characteristics. We believe that this research can supply ideas for future research on security and defense methods concerning adversarial examples in RSI scene classification systems.

The research on adversarial examples of RSIs is still rare. Compared to natural images, adversarial examples of RSIs also have the same properties in some aspects. However, attack selectivity of adversarial examples is a property not explored in natural images. This property might be more helpful for understanding the correlation between ground objects. This study also has somelimitations: (1) in the real scenario, adversarial examples of RSIs are usually placed with some physical objects, which would change the impact of the adversarial example problem [69]. (2) RSI has spectral and spatial properties. Different spatial resolutions would produce different adversarial examples, and these adversarial examples would have different properties [70]. In further research, we will explore these issues. Importantly, we will design defensive algorithms to solve the adversarial example problem.

8. Conclusion and Future Work

In this article, we discuss the adversarial example problem in RSI scene classification for the first time. The experimentshows that adversarial examples are difficult for the human eye to recognize, and they allow the RSI scene classification system to achieve false results. This situation can causeserious problems for applications based on RSI scene classification systems.

Adversarial examples in the field of RSIs are a problem that should be addressed. In the field of RSI applications, many detection and segmentation tasks remain, such as building detection and land cover classification. The processing of these tasks is different from that of image classification. Theadversarial example problems of these tasks require further study, and this topic is of great significance for the design ofa secure and robust CNN-based remote sensing application system.

9. Acknowledgment

This work was carried out in part using computing resourcesat the Indian Institute of Technology Hyderabad as a part of my summer internship project. I would like to sincerely acknowledge Prof. C. Krishna Mohan and his lab members for giving me this opportunity to work on the above problem

10. Code

The code for the above project can be found out on: https://github.com/Shorya-Sharma/Adversarial-Attacks-on-Aerial-Scene-Classification/tree/main

For accessing the experiment results and pre-trained weights files, please use the following drive link: https://bit.ly/2VNWj7q

11. References

[1] T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote Sensing and Image Interpretation*. Hoboken, NJ, USA: Wiley, 2015.

[2] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, ‘‘Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning,’’ *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, Jun. 2015.

[3] G. Forestier, A. Puissant, C. Wemmert, and P. Gançarski, ‘‘Knowledge- based region labeling for remote sensing image interpretation,’’ *Comput., Environ. Urban Syst.*, vol. 36, no. 5, pp. 470–480, Sep. 2012.

[4] J. Zhang, ‘‘Multi-source remote sensing data fusion: Status and trends,’’ *Int. J. Image Data Fusion*, vol. 1, no. 1, pp. 5–24, Mar. 2010.

[5] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, ‘‘Deep learning classification of land cover and crop types using remote sensing data,’’ *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 5, pp. 778–782, May 2017.

[6] K. Jia, Q. Li, Y. Tian, B. Wu, F. Zhang, and J. Meng, ‘‘Crop classifica- tion using multi-configuration SAR data in the North China plain,’’ *Int. J. Remote Sens.*, vol. 33, no. 1, pp. 170–183, Jan. 2012.

[7] C. Conrad, S. Fritsch, J. Zeidler, G. Rücker, and S. Dech, ‘‘Per-field irrigated crop classification in arid Central Asia using SPOT and ASTER data,’’ *Remote Sens.*, vol. 2, no. 4, pp. 1035–1056, Apr. 2010.

[8] C. Yuan, Y. Zhang, and Z. Liu, ‘‘A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehi- cles and remote sensing techniques,’’ *Can. J. Forest Res.*, vol. 45, no. 7, pp. 783–792, Jul. 2015.

[9] J. C. White, N. C. Coops, M. A. Wulder, M. Vastaranta, T. Hilker, and P. Tompalski, ‘‘Remote sensing technologies for enhancing forest inven- tories: A review,’’ *Can. J. Remote Sens.*, vol. 42, no. 5, pp. 619–641, Sep. 2016.

[10] L. Tang and G. Shao, ''Drone remote sensing for forestry research and practices,'' *J. Forestry Res.*, vol. 26, no. 4, pp. 791–797, Dec. 2015.
[11] G. J. Scott, M. R. England, W. A. Starms, R. A. Marcum, and C. H. Davis, ''Training deep convolutional neural networks for Land–Cover classification of high-resolution imagery,'' *IEEE Geosci. Remote Sens. Lett.*, vol. 14,no. 4, pp. 549–553, Apr. 2017.
[12] K. Jia, S. Liang, N. Zhang, X. Wei, X. Gu, X. Zhao, Y. Yao, and X. Xie, ''Land cover classification of finer resolution remote sensing data integrating temporal features from time series coarser resolution data,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 93, pp. 49–55, Jul. 2014.
[13] M. Awrangjeb, M. Ravanbakhsh, and C. S. Fraser, ''Automatic detection ofresidential buildings using LIDAR data and multispectral imagery,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 65, no. 5, pp. 457–467, Sep. 2010.
[14] M. Vakalopoulou, K. Karantzalos, N. Komodakis, and N. Paragios, ''Build-ing detection in very high resolution multispectral data with deep learning features,'' in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*,Jul. 2015, pp. 1873–1876.
[15] A. Manno-Kovacs and T. Sziranyi, ''Orientation-selective building detec-tion in aerial images,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 108,pp. 94–112, Oct. 2015.
[16] G. Cheng, P. Zhou, and J. Han, ''Learning rotation-invariant convo- lutional neural networks for object detection in VHR optical remote sensing images,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12,pp. 7405–7415, Dec. 2016.
[17] X. Li, W. Li, A. Middel, S. L. Harlan, A. J. Brazel, and B. L. Turner, ''Remote sensing of the surface urban heat island and land architecturein Phoenix, Arizona: Combined effects of land composition and con- figuration and cadastral–demographic–economic factors,'' *Remote Sens. Environ.*, vol. 174, pp. 233–243, Mar. 2016.
[18] R. J. Shrivastava and J. L. Gebelein, ''Land cover classification andeconomic assessment of citrus groves using remote sensing,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 61, no. 5, pp. 341–353, Jan. 2007.
[19] Y. Ke and L. J. Quackenbush, ''A review of methods for automatic indi- vidual tree-crown detection and delineation from passive remote sensing,''*Int. J. Remote Sens.*, vol. 32, no. 17, pp. 4725–4747, Sep. 2011.
[20] J. Hu, A. Razdan, J. C. Femiani, M. Cui, and P. Wonka, ''Road net-work extraction and intersection detection from aerial images by tracking road footprints,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 12,pp. 4144–4157, Dec. 2007.
[21] G. Fu, H. Zhao, C. Li, and L. Shi, ''A method by improved circular projection matching of tracking twisty road from remote sensing imagery,''*Acta Geodaetica et Cartographica Sinica*, vol. 43, no. 7, pp. 724–730, 2014.
[22] R. Crippen, ''Calculating the vegetation index faster,'' *Remote Sens. Envi-ron.*, vol. 34, no. 1, pp. 71–73, Oct. 1990.
[23] S. K. McFEETERS, ''The use of the normalized difference water index (NDWI) in the delineation of open water features,'' *Int. J. Remote Sens.*, vol. 17, no. 7, pp. 1425–1432, May 1996.
[24] X. Huang and L. Zhang, ''Morphological building/shadow index for build-ing extraction from high-resolution imagery over urban areas,'' *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 1, pp. 161–172,Feb. 2012.
[25] J. Chen, M. Deng, X. Mei, T. Chen, Q. Shao, and L. Hong, ''Optimal segmentation of a high-resolution remote-sensing image guided by area and boundary,'' *Int. J. Remote Sens.*, vol. 35, no. 19, pp. 6914–6939,Oct. 2014.
[26] G. Liasis and S. Stavrou, ''Building extraction in satellite images using active contours and colour features,'' *Int. J. Remote Sens.*, vol. 37, no. 5, pp. 1127–1153, Mar. 2016.
[27] G. Cheng and J. Han, ''A survey on object detection in optical remote sensing images,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 117, pp. 11–28, Jul. 2016.
[28] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, ''Deep learning for remote sensing image classification: A survey,'' *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 8, no. 6, p. e1264, Nov. 2018.
[29] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, ''Deep learn-ing in remote sensing applications: A meta-analysis and review,'' *ISPRS J. Photogramm. Remote Sens.*, vol. 152, pp. 166–177, Jun. 2019.
[30] L. Deng, ''A tutorial survey of architectures, algorithms, and applications for deep learning,'' *APSIPA Trans. Signal Inf. Process.*, vol. 3, pp. 1–29, Jan. 2014.
[31] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, ''Deep learning-based classification of hyperspectral data,'' *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.
[32] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, ''Deep feature extrac- tion and classification of hyperspectral images based on convolutional neural networks,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10,pp. 6232–6251, Oct. 2016.
[33] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, ''When deep learn-ing meets metric learning: Remote sensing image scene classificationvia learning discriminative CNNs,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018.
[34] F. Zhang, B. Du, and L. Zhang, ''Scene classification via a gradient boosting random convolutional network framework,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1793–1802, Mar. 2016.
[35] S. Chaib, H. Liu, Y. Gu, and H. Yao, ''Deep feature fusion for VHR remotesensing scene classification,'' *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4775–4784, Aug. 2017.
[36] N. Akhtar and A. Mian, ''Threat of adversarial attacks on deep learning incomputer vision: A survey,'' *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, ''Explaining and harnessing adversarial examples,'' in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, Y. Bengio and Y. LeCun, Eds. May 2015. [Online]. Available: http://arxiv.org/abs/1412.6572
[38] A. Kurakin, I. Goodfellow, and S. Bengio, ''Adversarial examples in the physical world,'' 2016, *arXiv:1607.02533*. [Online]. Available: http://arxiv.org/abs/1607.02533
[39] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, ''Synthesizing robust adversarial examples,'' 2017, *arXiv:1707.07397*. [Online]. Available: http://arxiv.org/abs/1707.07397
[40] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, ''Intriguing properties of neural networks,''2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312. 6199
[41] N. Carlini and D. Wagner, ''Adversarial examples are not easily detected:Bypassing ten detection methods,'' in *Proc. 10th ACM Workshop Artif. Intell. Secur. (AISec)*, 2017, pp. 3–14.