High-speed Rail Operating Environment Recognition Based on Neural Network and Adversarial Training

Xiaoxue Hou*, Jie An*, Miaomiao Zhang*§, Bowen Du†§, Jing Liu‡§
*School of Software Engineering, Tongji University, Shanghai, China
†Department of Computer Science, University of Warwick, Coventry, England
‡School of Software Engineering, East China Normal University, Shanghai, China

***Abstract*—Neural network is one of the key technologies for deep learning. Experiments on some standard test datasets show that their recognition ability has reached the level of human beings. However, they are extremely vulnerable to adversarial examples, that is, adding some subtle perturbations to the input example can cause the model to give a wrong output with high confidence. In this paper, we propose a non-contact approach based on neural network and adversarial training to recognize the high-speed rail operating environment. We first built the environment dataset and trained neural network models to do the recognition. We found that our model had high prediction accuracy, but with poor security since it was easy to attack our model using Basic Iterative Methods (BIM). To improve its security, we performed adversarial training based on the adversarial training dataset we built. The evaluation experiments indicated that this approach could improve the security of our model at the same time ensuring the prediction accuracy on the original test dataset.**

***Index Terms*—Recognition, Neural network, Adversarial attack, Defense**

I. Introduction

Neural network is one of the deep learning algorithms [1]. It mimics the structure and operation mechanism of biological neural network. In the past ten years, the research work of neural networks has been deepened, and great progress has been made. Neural network has successfully solved many practical problems in the areas of pattern recognition, intelligent robotics, automatic control, predictive estimation, biology, medicine and economy [2].

However, when neural network distinguishes adversarial examples, the effect is extremely unstable. Just deliberately adding some subtle perturbations to the input example that is undetectable by humans can cause the model to give a wrong output with high confidence [3]. Therefore, it is of great significance to further study the reasons for the adversarial examples and defense methods of them.

In this paper, we focus on the task of high-speed rail operating environment recognition. Nowadays, Chinese high-speed rail construction industry has entered a period of vigorous development, which is particularly important to the diagnosis and treatment of high-speed rail failures. However, different high-speed rail faults have different discrimination and treatment methods in different operating environments [4]. Take the arc detection in pantograph-catenary system as an example [5]. The light brightness has obvious affect on the arc detection, and in many arc detecting methods, we always have to distinguish the operating environemnt before detecting the magnitudes of arc. At present, only a few high-speed rail systems are equipped with environmental sensors to monitor the operating environment, most of them do not have sensor devices [6]. And environmental sensors often fail. In actual work, this task is mainly fulfilled by the staff playing back the monitoring video to manually recognize the environment based on their observations and experiences. This kind of diagnosis is not rigorous since it is not continuous in time and space, and there are some hidden dangers to make mistakes. Therefore, creating a non-contact method to recognize the high-speed rail operating environment automatically is of vital importance.

Here is the structure of this paper. In section II, we summarize the related researches of convolutional neural networks, adversarial examples and their defense methods, to lead our approach. And then in section III, we build dataset from monitoring videos of the high-speed rail operating environment to support the task. Next in section IV, we train several neural network models and choose one with best prediction accuracy to do the recognition. In section V, to improve the security of the model we obtained before, we perform adversarial attacks and adversarial training. In section VI, we give summary, limitations and future work of our research, and finally we give conclusion in section VII.

II. State Of The Art

Deep learning [7] is a method based on characterizing and learning data which uses unsupervised or semi-supervised feature learning and hierarchical feature extraction algorithms instead of manual extraction to get features. The goal of characterization learning is to find better representations and create better models to learn these representations from large-scale unlabeled data. The expression is similar with the advancement

Supported by NSFC 61472279, 61972284, 61972150, 61572195.
§Corresponding authors: miaomiao@tongji.edu.cn, B.Du@warwick.ac.uk, jliu@sei.ecnu.edu.cn

2375-0197/19/$31.00 ©2019 IEEE
DOI 10.1109/ICTAI.2019.00120

of neuroscience [8]. So far there have been several deep learning frameworks, such as deep neural networks, convolutional neural networks and deep belief networks as well as recurrent neural networks, which have been successfully applied in the fields of computer vision, speech recognition, natural language processing, audio recognition and bioinformatics [9]–[12].

The concept of adversarial examples was first proposed by Christian Szegedy et al. in 2013 [3], he pointed out that when deliberately adding some subtle perturbations on the input example which were noticed by humans could cause the model to give a wrong output with high confidence.

Goodfellow et al. [13] explained that the reason why it is difficult for neural networks to resist against disturbances is the linear characteristics of neural networks, and they designed Fast Gradient Sign Method (FGSM) to quickly generate adversarial examples, which confirms the high-dimensional linearity of the design of modern deep neural networks. FGSM is a common white-box attack method with a very simple idea. It mainly uses the loss function of the model to obtain the adversarial perturbation for the input gradient, and then adds it to the input to generate adversarial examples. We define an adversarial example $\tilde{x}$ of an input x as $\tilde{x} = x + \eta$, where η is defined as: $\| \eta \|_\infty \leq \epsilon$, and ϵ is required to be small enough to be discarded by the neural network model. We consider the dot product between a weight vector ω and an adversarial example $\tilde{x}$ as:

$$\omega^T \tilde{x} = \omega^T x + \omega^T \eta \tag{1}$$

As can be seen from equation 1, a small adversarial perturbation can significantly increase the output of neurons. Goodfellow pointed out that if the amount of change in the adversarial perturbation was exactly the same as the direction of the gradient, then there would be a maximal change in the classification results. We assume that the dimension of the weight vector is n and the mean of the weight vector is m, then the maximum value of $\omega^T \eta$ is ϵmn. And now $\eta = \epsilon sign(\omega)$, in which sign function guarantees the amount of change in the adversarial perturbation is the same as the direction of the gradient.

We consider that θ is the parameter of a model, y is the target with input x, and $J(\theta, x, y)$ is the loss of the neural network, so the max-norm perturbation is given as equation 2, which is also the definition of the fast gradient sign method:

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{2}$$

The one-step method performs image perturbation by increasing the loss function of the classifier by a large-scale operation [14], and it can be directly extended to increase the variation of the loss function through multiple small steps, so that we get Basic Iterative Methods (BIM).

Recently more and more researchers start to study the defense of adversarial examples. Although people's understanding of adversarial examples has been improved, the related defense methods also have been improved, but so far, there is no universal and complete solution.

Adversarial training is a way to deal with adversarial examples. The network's robustness is gradually improved by continuously entering new types of adversarial examples and performing adversarial training. In order to ensure effectiveness, this method requires the use of high-intensity adversarial examples, and the network architecture must have sufficient expressive power. This method requires a large amount of training data and thus it is called brute force adversarial training. This brute force adversarial training can regularize the network to reduce overfitting [15].

Image denoising [16] is a way to remove the noise exists in the image and restore the image to make the classifier give a correct prediction output. There are some traditional denoising methods include median filter and BM3D, and also we can use neural networks to denoise.

Image compression is used to reduce the redundant information of the image while maintaining the significant information, since there is a strong similarity and correlation between adjacent pixels in the local structure of image. Dziugaite et al. [17] used JPG image compression to reduce the impact of adversarial perturbations to improve the prediction accuracy. This method is effective for some part of the attack methods, but also reduces the accuracy of original inputs while compressing images.

There are also some other defense methods, and the research direction of defense methods is full of controversy and changes with each passing day.

III. Dataset Building and Data Preprocessing

Our original data source is the monitoring video of the high-speed rail operating environment in the areas of Chinese Jinan and Chongqing. This video covers a variety of topographical features, shooting angles, vehicle types, time periods and weather conditions. We transferred the video into images with 224×224 pixels. Through the analysis of the video and the key factors in the recognition of the high-speed rail operating environment, we divided the original dataset into 6 categories according to four indicators, as indicated in table I, which are whether daytime, whether foggy, whether rainy and whether in the tunnel with the labels of 0, 1, 2, 3, 4 and 5.

TABLE I: Classification Basis

Indicator	Daytime	Foggy	Rainy	In the Tunnel
0	Y	Y	N	N
1	Y	N	N	N
2	Y	N	Y	N
3	N	N	N	N
4	N	N	Y	N
5	N	N	N	Y

Actually in the monitoring videos there are also many other indicators, for example, the vehicle types include 380A and 380D. However, these kinds of indicators are less important to the recognition since we can take similar actions to deal with different problems happened with different vehicle types. The

indicators we choose are based on weather recognition, time period recognition and specific cases recognition.

The weather conditions our videos include are foggy, rainy and sunny, which will affect the air humindity, light intensity and whether to enable the emergency mode [18]. Environmental sensor is another choice to recognize them, while it is very fragile in harsh environments, and also expensive to install. In this case the weather recognition is an important part of high-speed rail operating environment recognition. The time periods in our videos include daytime, dusk and night. Considering that in the daytime and dusk the light is much brighter than it is at night, we only take daytime and night to category images, so the images in the dusk are also categoried as in the daytime. As for specific cases, we only consider whether it is in the tunnel, since the monitoring screen overexposures when the rail enters and leaves the tunnel, which should be excluded from the normal analysis.

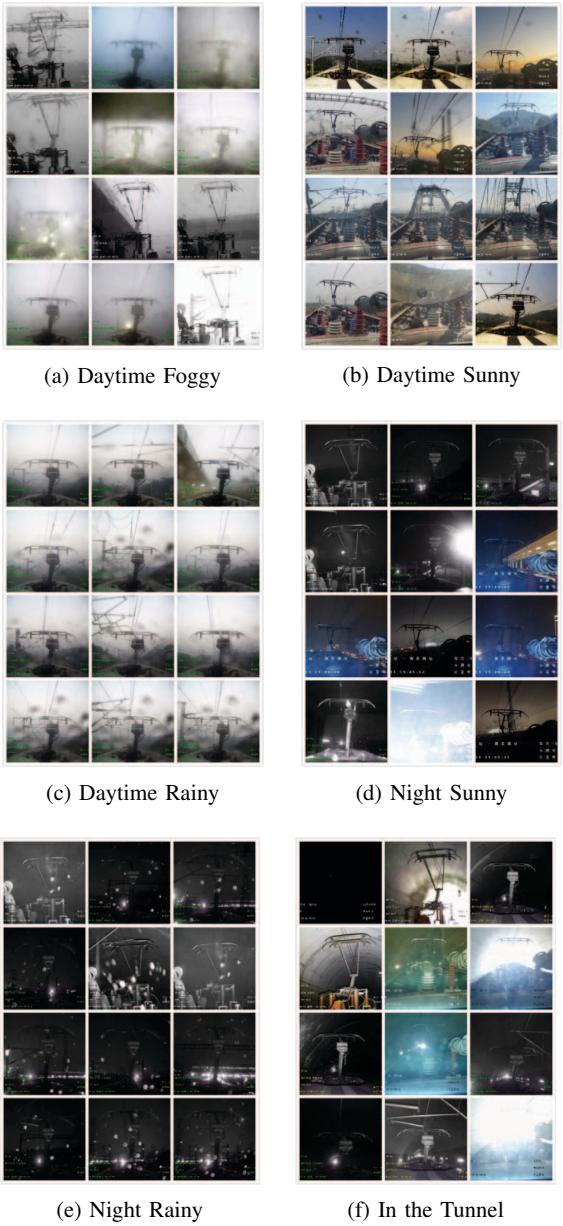
According to the analysis above, we get the dataset with 30,000 images belonging to 6 categories. Figure 1 shows a part of images belonging to each of the six categories.

The division of the dataset plays a key role in neural network training. It is a way to divide the dataset into training dataset and test dataset [19], training dataset is for the training of the model and test dataset is for the model evaluation of generalization ability. However, there are also many hyper-parameters inside the neural network that needs to be selected manually, such as the number of layers of the model and the number of neurons in each layer of the model. We have to adjust those hyper-parameters to get the model with better generalization ability, and this is why we introduce validation dataset. Goodfellow in his book [20] pointed out the training dataset, validation dataset and test dataset could be divided in a ratio of 6:2:2 if the amount of the data was about 10,000, since we have totally 30,000 images in our dataset, we get training dataset, validation dataset and test dataset separately with 21600, 7200 and 7200 images.

The recognition task is regarded as a multi-classification problem. In this task, features are not continuous values, but categorical values. Here we use numbers of (0, 1, 2, 3, 4, 5) to separately represent the six categories, which will greatly improve the computing efficiency. However, even after converting them to digital representations, the integer feature representation cannot be directly used in the classifier. Because for such continuous input, the classifier considers the classes to be ordered, but actually they are unordered. We binarize the integer labels so that they are regarded by the classifier as a vector from the euclidean space.

Normalization is another important part when we preprocess the data. It is mainly for the speed of data processing, and it transfers dimensioned expressions into dimensionless expressions so that indicators of different units or magnitudes can be compared and weighted. Since the range of image pixels is between 0 and 255, we divide the image pixels by 255 to realize normalization.

Till now the dataset for the neural network model training is finely prepared.



(a) Daytime Foggy (b) Daytime Sunny

(c) Daytime Rainy (d) Night Sunny

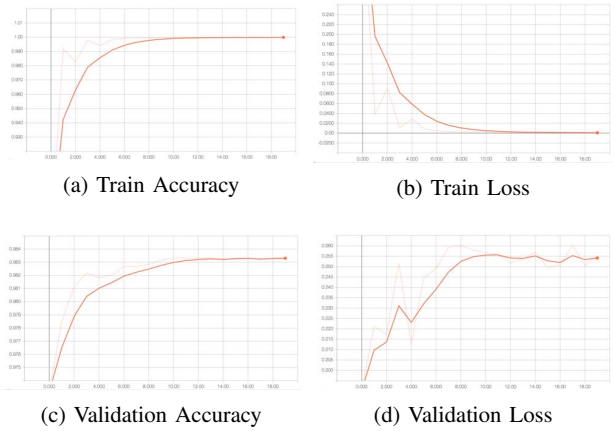
(e) Night Rainy (f) In the Tunnel

Fig. 1: Examples of Six Categories

IV. Neural Network Model Training

After building the dataset, we begin to train neural network models to solve the recognition task. Convolutional neural network [21] is a kind of feedforward neural networks with convolutional computation and deep structure. It is one of the representative algorithms of deep learning and has achieved very good results in the fields of computer vision. So far there has been many popular convolutional neural network models, such as LeNet, AlexNet, ZFNet, VGG-16 and ResNet [22].

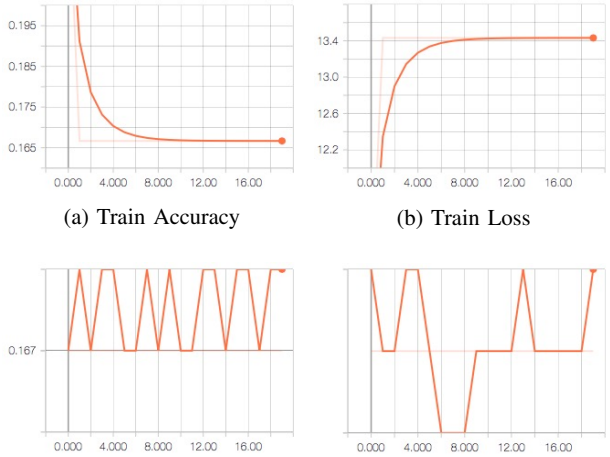
LeNet [23], born in 1994, is one of the classic structures of convolutional neural networks. It uses features such as



(a) Train Accuracy (b) Train Loss

(c) Validation Accuracy (d) Validation Loss

Fig. 2: Results of LeNet Model



(a) Train Accuracy (b) Train Loss

(c) Validation Accuracy (d) Validation Loss

Fig. 3: Results of AlexNet Model

convolution, parameter sharing and pooling to extract features, avoiding a large amount of computational cost, and finally applys fully connected neural networks for classification and identification.

In 2012, Krizhevsky [24] used the convolutional neural network to win the ILSVRC 2012 image classification contest and proposed the AlexNet model, which spurred the wave of neural network research with many innovative methods.

In this paper, we tried both LeNet model and AlexNet model to train. We adopted Keras framework [25] with the version of 2.2.4 to build our neural network models. Keras is a highly modular neural network library supported by Google, whose backend is based on Tensorflow and Theano. It is written in Python and supports both GPU and CPU. The first release of Keras was in March 2015, and it is favored for the ease of use and simple syntax, making development faster [26].

After 20 epochs training, the precision accuracy and the loss function of training and validation of LeNet model are shown on the Tensorboard in Figure 2. It shows that the prediction accuracy of training and validation are both rising gradually, and they finally close to 1, the loss function of training and validation are tending to convergence, which indicates that the training effect is stable.

And after 20 epochs training, the precision accuracy and the loss function of training and validation of AlexNet model are shown on the Tensorboard in Figure 3.

In our experiments, we found that AlexNet model was over-fitting on our dataset, since both the loss function curve and accuracy function curve of validation have severe concussion, which shows that the model has very poor stability. So in this case, we choose LeNet model to solve the task. We applied LeNet model for the prediction on the test dataset, and obtained the prediction accuracy of 0.9837, which shows that the fitting degree of the model is acceptable.

The LeNet model consists of two convolutional layers, two maxpooling layers and two fully-connected layers. Figure 4 is the structure of our model, which shows the architecture, input and output of layers.

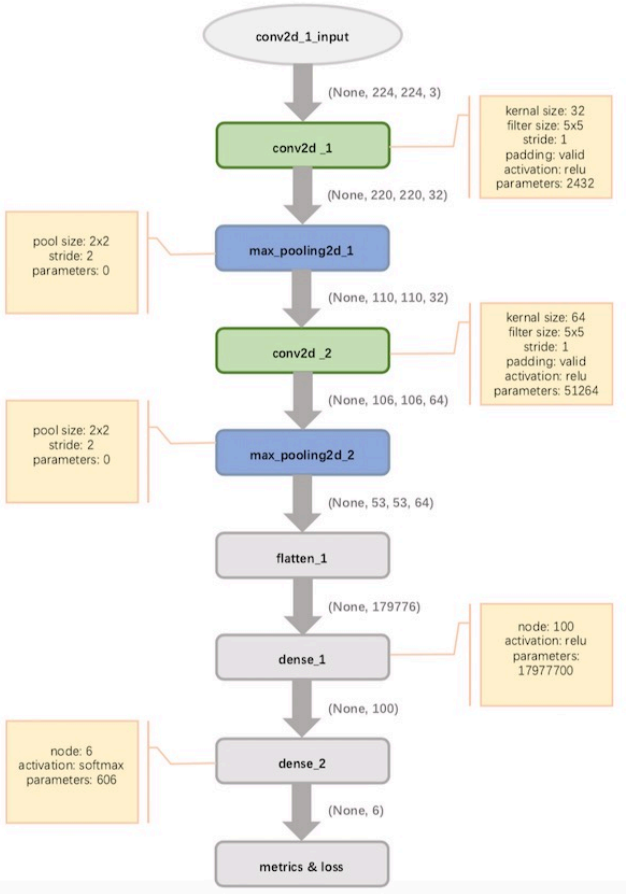


Fig. 4: Structure of LeNet Model

The conv2d_1_input is the input layer, which provides the original images with the input shape of 224×224×3 for the model, that is, the pixels of the images are 224×224, and they are color images with channel number of 3. The conv2d_1 is the convolutional layer, and it has 32 kernels with the size of 5×5 and stride size of 1. Since the it does not use full zero complement, the output size of this layer is 224-5+1=220, with the depth of 32, that is 220×220×32. The total number of parameters of this layer is 5×5×1×32×3+32=2432. The maxpooling2d_1 is the down-sampling layer with pool size of 2×2 and the stride size of 2, so the output size of this layer is 110×110×32. This layer uses the image principle of local correlation to subsample the images, which could reduce the amount of data processing while retaining useful information. The conv2d_2 is the convolutional layer with 64 kernels, the size of them is 5×5 and the stride size is 1. It does not use full zero complement, neither. The output size of this layer is 110-5+1=106, with depth of 64, that is 106×106×64. The total number of parameters of this layer is 5×5×1×64×32+64=51264, 32 is the depth of the input. The layer maxpooling2d_2 is also the down-sampling layer with pool size of 2×2 and stride of 2, so the output size of this layer is 53×53×64. The flatten_1 layer flattens all the nodes, so the output is 53×53×64=179776. The first fully-connected layer dense_1 has 100 nodes, so the output size is 100 and the total number of parameters of this layer is 179776×100+100=17977700. The output layer dense_2 is also the fully-connected layer with 6 nodes, so the output size is 6, and the total number of parameters of this layer is 100×6+6=606. Total number of parameters is 2432+51264+17977700+606=18032002.

V. Adversarial Attacks and Adversarial Training

In order to check the security of the LeNet model we obtained in the previous section, we apply adversarial attack methods to attack the model. Actually there are many attack methods, while considering the complexity of algorithms and our hardware devices, we take both BIM method and FGSM method to attack the model, which are introduced in the section I before.

There are two types of adversarial attack methods, that are untargeted adversarial attacks and targeted adversarial attacks. Untargeted adversarial attacks just want to confuse the model to give a wrong output, while targeted adversarial attacks want the model to give a wrong desired output. In this section, we implement untargeted adversarial attacks since we just want to confuse the model.

We set the *iteration* number as 50, and *epsilon* as 0.1, which is the allowable fluctuation range of the sample. We conducted the experiments for many time and finally found these appropriate values, that is, we regarded the adversarial examples and original images as the same category when *epsilon* is 0.1.

As can be seen in table II, we take 12 sets of examples to show in the paper, and each set contains the original image and adversarial examples generated by BIM method and FGSM method. Specific images can be found in Appendix A.

TABLE II: Original images and their corresponding adversarial examples generated by FGSM and BIM method.

Number	Original Class	FGSM		BIM	
		New Class	Confidence	New Class	Confidence
1	0	3	0.62005836	3	0.9932944
2	1	5	0.6902759	5	0.7287956
3	1	0	0.63797605	0	0.7664373
4	1	5	0.39251974	5	0.44168004
5	1	5	0.8008037	5	0.8255589
6	2	0	0.6066581	0	0.8976936
7	3	0	0.7571429	0	0.85842884
8	3	4	0.5396405	1	0.5554841
9	3	4	0.9026469	4	0.78059787
10	4	3	0.87523174	3	0.82832044
11	5	3	0.886363	3	0.831025
12	5	3	0.74626285	3	0.944448

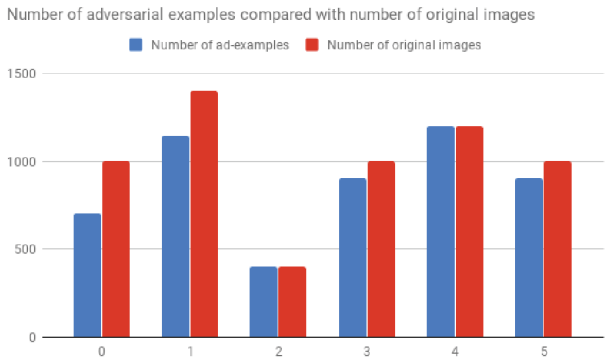


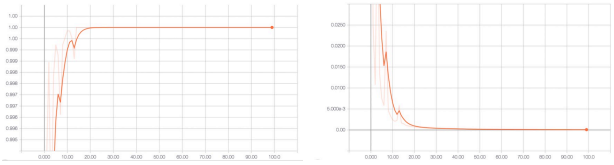
Fig. 5: Number of Adversarial Examples Compared With Number of Original Images

The attack experiments on the test dataset show that almost all of the adversarial examples created by BIM look very similar to the original examples, while there are some differences between original examples and adversarial examples created by FGSM, which proves that our model is vulnerable to the BIM method, and BIM method is more effective than FGSM method on our model.

Figure 5 shows the number of adversarial examples generated by BIM in each category compared with the number of original images. The attack success rate of BIM on our test dataset is 5351/6000=0.89.

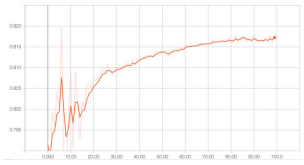
To improve the security of our model, we perform adversarial training.

Firstly, we built the adversarial example dataset, which is made up of both original images and adversarial examples generated by BIM method. We got 5351 adversarial examples after using BIM method to attack our test dataset, with the number of each class as 705, 1147, 400, 904, 1200 and 995.

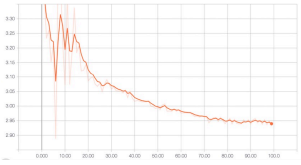


(a) Train Accuracy

(b) Train Loss



(c) Validation Accuracy



(d) Validation Loss

Fig. 6: Results After Adversarial Training

We picked the original images with the same number in each class, and mixed them up, so in our adversarial example dataset, we got 10702 images, with the ratio of orginal images and adversarial examples as 1:1. The reason why we dont directly use the adversarial examples, but adding original images into the adversarial training dataset is that we want to improve the security of the model at the same time ensuring that the prediction accuracy of the model would not decline.

We split the adversarial dataset into adversarial training dataset, adversarial validation dataset and adversarial test dataset in a ratio of 6:2:2, the same as the model training before, and performed adversarial training.

The predicion accuracy and the loss function of the adversarial training dataset and adversarial validation dataset are shown on the Tensorboard in Figure 6. For the prediction accuracy of the retained model, the value is 0.9992 on the original test dataset, and is 0.9977 on the adversarial test dataset. It shows that the retrained model has better security without reducing the prediction accuracy on the original test dataset compared with the previous model.

VI. Discussion

In this paper, we build a neural network model with high prediction accuracy and high security to deal with the task of high-speed rail operating environment recognition. Here we summarize our work as follows:

1) Dataset Building
   We built dataset containing 30,000 images within six categories, and split the dataset into training dataset, validation dataset and test dataset.
2) Neural Network Model Training
   We obtained a neural network model with high prediction accuracy to solve the task.
3) Adversarial Attacks and Adversarial Training
   We applied adversarial attacks to check the security of the model we trained before, and the results of experiments indicated that our model has poor security under BIM attack. To improve the security of our model, we applied adversarial training.

Since we already know that the deep learning models are extremely vulnerable to adversarial examples, we want to improve the security of our model. Security problem of deep learning is a hot topic, and there are many adversarial attack methods which have very good effect to attack the deep learning models, and then defense methods come behind. However, there is no universal and complete solution. We tried some experiments, and want to make a small research contribution to the defense of adversarial attacks.

There are mainly two aspects that restrict our experiments, and for each one we give future work.

1) Our hardware devices are not ideal enough. All of the experiments in this paper were performed on the MacBook Pro with the processor of 2.9 GHz Intel Core i5, which limited the training speed of the neural network to some extent. If we could apply GPU processor to train neural network models, we would be able to train far more than 20 epochs, maybe 500 epochs, so as to achieve higher prediction accuracy. In addition, better processor makes it possible for us to do adversarial training more times to continually improve the security of our model.
2) Our dataset could be more abundant. The types of environmental conditions that currently contained in our dataset are relatively rich, but they are not enough to cover all the weather and geographical conditions. Especially in our dataset, the images in the category of daytime rainy are much simpler compared with the images in other categories. In the future, if it is possible, we will try for more monitoring videos of high-speed rail operating environment, and then we could adjust the classification of the current dataset to get our research approach applied.

VII. Conclusion

In this paper, we built a neural network model with high prediction accuracy and high security to deal with the task of high-speed rail operating environment recognition. Firstly, we established the dataset based on the monitoring video during high-speed rail operation, and then we trained neural network models to solve the task. Secondly, we tried both FGSM method and BIM method to attack the model we trained before. The results of experiments indicated that BIM method had better attacking effect on our model, so we collected adversarial examples attacked by BIM method and built dataset for adversarial training. Finally, we retrained the model based on the adversarial training. The evaluation experiment proves that the retrained model has higher security without reducing the prediction accuracy on the original test dataset compared with the previous model.

To sum up, this kind of non-contact approach based on neural network and adversarial training can be applied to recognize the high-speed rail operating environment. It is also possible to extend our research results to more neural

network systems that have higher security requirements, such as automatic car driving, face recognition and medical health.

REFERENCES

[1] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[2] Abidha Pandey, Manish Puri, and Aparna Varde. Object detection with neural models, deep learning and common sense to aid smart mobility. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 859–863. IEEE, 2018.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[4] Zhiwei Han, Zhigang Liu, Guinan Zhang, and Hong-mei Yang. Overview of non-contact image detection technology for pantograph-catenary monitoring. *Journal of the China Railway Society*, 35(6):40–47, 2013.

[5] Shize Huang, Yachan Zhai, Miaomiao Zhang, and Xiaoxue Hou. Arc detection and recognition in pantographcatenary system based on convolutional neural network. *Information Sciences*, 501:363 – 376, 2019.

[6] Ilhan Aydin, Selahattin B Celebi, Sami Barmada, and Mauro Tucci. Fuzzy integral-based multi-sensor fusion for arc detection in the pantograph-catenary system. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 232(1):159–170, 2018.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[8] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[9] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[10] Sholom M Weiss and Casimir A Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*, volume 123. Morgan Kaufmann San Mateo, CA, 1991.

[11] Priya Persaud, Aparna S Varde, and Stefan Robila. Enhancing autonomous vehicles with commonsense: Smart mobility in smart cities. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1008–1012. IEEE, 2017.

[12] Salvatore Sorce, Vito Gentile, and Antonio Gentile. Real-time hand pose recognition based on a neural network using microsoft kinect. In *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 344–350. IEEE, 2013.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[16] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[17] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[18] Regina Ruby Lee Clewlow. *The climate impacts of high-speed rail and air transportation: a global comparative analysis*. PhD thesis, Massachusetts Institute of Technology, 2012.

[19] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[21] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[22] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[23] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20, 2015.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[26] Nikhil Ketkar. Introduction to keras. In *Deep Learning with Python*, pages 97–111. Springer, 2017.

APPENDIX A
ADVERSARIAL EXAMPLES GENERATED BY BIM, FGSM AND THEIR ORIGINAL IMAGES