Journal of Artificial Intelligence Practice (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/medsc.2022.030208
ISSN 2371-8412 Vol. 6 Num. 8

Machine learning: Training model with the case study

Tianyue Jiang[1,*], Lihong Zhang[1]

[1]*Central University of Finance and Economics, Beijing, China*
[*]*Corresponding author*

Keywords: Machine learning, Deep learning, CNN, GAN

Abstract: In recent decades, making machines learn from a massive dataset has become a prevalent task. It is significant to build a suitable deep learning model which would make its own decisions after training. This project aims to deal with a massive dataset with the trained deep learning model. More specifically, this project applies a Convolutional Neural Network (CNN) model to classify massive text data. It applies Generative Adversarial Network (GAN) model to regress image data. Furthermore, we start by building the model and enhance the model with necessary modifications for satisfying performance. Then we train and test the model to verify that the model achieves its aim. While the whole process implements based on Python and the Tensor Flow environment.

1. Introduction

With the technology developing, the human has to deal with huge datasets, and the traditional methods do not satisfy the modern demand, and the machine shows advantages in handling big data. Artificial Intelligence (AI), which aims to ensure machine learning without humans, has been widely used. While machine learning is a primary method to achieve AI, and deep learning is a subfile of machine learning based on particular neural network architecture. Deep learning contains many neural network structures to complete different data format tasks. While classification and regression are the most significant tasks. The project mainly contains several sections. First part is the basic theory of machine learning and deep learning. The second section focuses on completing case study one, which builds and enhances a CNN model to classify text data. While the third part is case study two that builds and enhances a GAN model to regress image data. Case studies are implemented in Python 3.6.0, and I applied Anaconda to build the virtual Tensorflow environments. In addition, I also applied TensorBoard as the visualizing and supervising tool, and all operations of the project completed in the macOS platform. Finally, the final part is a summary section.

2. Literature review

2.1 Introduction of Machine Learning and Deep Learning

Machine learning has regarded as an effective method to solve massive dataset tasks by relaying on algorithms to detect the rules. While the computer has huge memory space and fast computation speed. It generally needs learning time until it achieves satisfying results since training is the most critical step, and it is significant to choose a suitable model. It contains supervised, semi-supervised

and unsupervised learning. In this project, we focus on supervised learning which trains the model with the given correct output so that the model knows the correct answer in the training process; hence, it shows benefits on classification problems. In contrast, unsupervised learning model learns the information without a given correct answer, so it is good at finding disciplines problem.

Deep learning uses nonlinear and straightforward models which is good at dealing with complex nonlinear tasks and solving high-dimensional problems. The machine learns ideally by relying on massive training and high operation speed, which is impossible for the typical human. For example, the AlphaGo play go with human since it has been trained more than ten million times, which cannot be achieved by humans. Hence, the human brain is good at generalization based on a limited dataset, and it contains lots of neurons which can quickly inspire others. While neurons send and receive information, and channels connect them. Deep learning applies a neural network (NN) structure similar to the human brain structure. All the neurons are divided into multiple layers, and neurons in one layer can inspire other layers. The neurons have their own learnable biases and weight, which determines the state of the next layer neuron. More specifically, we number the neurons with X1, X2, X3…Xi in one layer. While each neuron has its own weight W1, W2, W3…Wi, and the system uses different weight to represent the importance. Then the system determines the state of the next neuron following the result of function 1, referenced from [1]. If the result of function 1 is equal to or smaller than the threshold value, the output will be zero, which means that the state of the next layer neuron is inactive. While the next layer neuron's state is active when the result of function 1 is greater than the threshold, and the output will be one.

$$\Sigma\,(W_i \times X_i) \qquad (1)[1]$$

2.2 Convolutional Neural Network (CNN)

Yann LeCun, Wei Zhang first put forward the Convolutional Neural Network(CNN) in 1989 [1]. It is the representative neural network with multiple layer structure, and it applies the convolution operation and the pooling operation to improve the efficiency. Case study one builds a complete CNN model, which can classify positive or negative meaning sentences. First, it is essential to build necessary layers, and the arguments and structures should be modified for better performance. Then, the CNN is supervised learning, so we divided the training dataset into the positive and negative categories, and threw them into the CNN model to extract features while it requires some time to learn the features. When the model achieves perfect accuracy, we input random sentences into the model to verify the classification performance. During the CNN operation process, it drops data into the input layer neurons, then transfer the neurons to convolutional layer to extract features. While the pooling layer delete unnecessary features, and it is connected with activation layer. The fully connected layers and output layer are the ends of the whole structure. The following architecture is the foundation of CNN, so we reference Figure 1 from [2] to show the relationship.

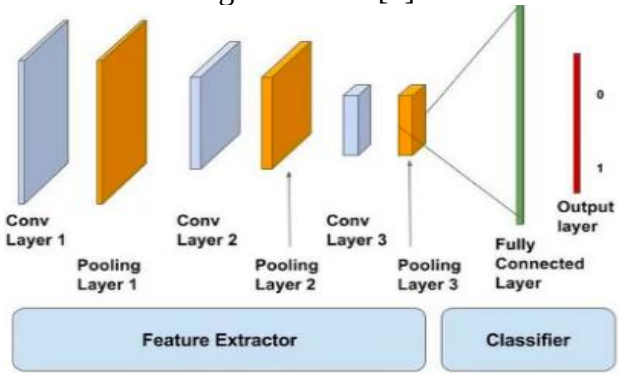

Figure 1: The structure of Convolutional Neural Network(CNN) [2].

2.2.1 The structure of CNN

The input layer is the first layer, and it receives the original data. While it divides the input data into several pixels value, and it organizes the pixels into an array. While the CNN is constructed by lots of neurons, so each pixel value connects with one neuron in this layer. Moreover, the neurons in the input layer connect with the convolutional layer neurons using channels. The CNN can only operate the numerical format, and the input layer uses an integer number to mark every word based on the word's frequency. We generally use embedding to represent the relationship of the several words since the embedding contains many vectors, and each vector represents one word.

Then, convolutional layer is the most representative layer. It aims to extra features of input data, and it contains a convolutional operation that is the linear operation and activation function that is a nonlinear part. While the convolutional operation applies many filters, which seem like a small window used to observe a part of the data so that the feature can be extracted more specifically. In this project, filters aim to observe embeddings representing some words, and we set that one filter can slide three, four or five words. While the filter will move to the next new area, and it defines the stride to represent the length of each moving step. When the filter slide over all the vectors, it generates a result which is called a feature map. The convolutional operation is followed by a nonlinear activation function, which adds nonlinear factors to the previous linear features, and I apply the Rectified Linear Unit (ReLU) function to convert linear into nonlinear. If it only operates linear function, the output of the multilayer is similiar to the single layer. All the neurons with negative input values move to zero at high speed, and it is a piecewise function with a gradient is equal to 1, it has been widely applied to avoid gradient explosion and gradient disappearance.

The pooling layer is unique, and it reduces the number of insignificant computations and parameters. It controls the spatial size and remains the essential features; however, pooling operations may result in overfitting problems. In this projext, I applied the max-pooling which partitions the feature map into several areas with a fixed size, and it only picks the maximum value which reprsents the most useful feature from each area to generate the new feature map.

Generally, the next layer is fully connected layer, and it solve the nonlinear problem. It aims to achieve classification, and it converts the feature into neurons by using convolution operation. The Figure 2, referenced from [9], describes the fully connected layer structure. It shows that the left part are several feature maps, resulting from convolution and pooling operations. The middle part shows two fully connected layers, while each layer contains 4096 neurons which need 4096 convolution operations. After computation, each neuron represents one subclass, and the convolution operations reduce the effect of feature position. However, the fully connected layer may result in an overfitting problem. Then it always applies logistic regression to classify which is the softmax method since it is good at classify multiple classes. Each feature has a particular weight w, and we get output y= xw. Hence, different classes are represented by different probabilities.

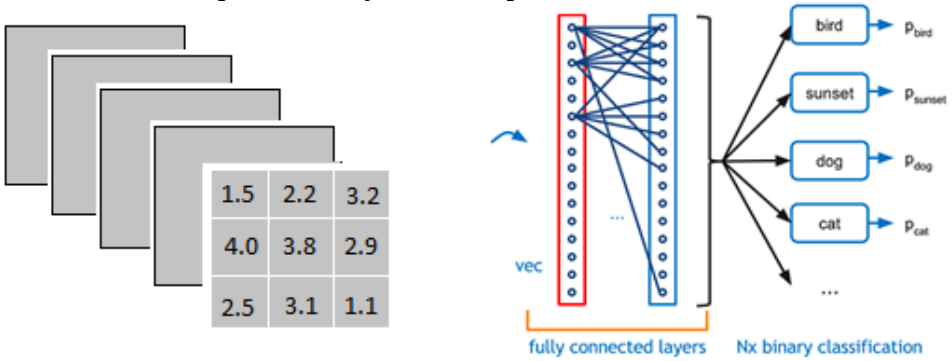

Figure 2: The structure of the fully connected layers [9].

The output layer is the last layer of CNN. Since each neuron already has its own probability value, it determines the output by choosing the maximum value among all neurons. While the output layer applies the cross-entropy function, which is the loss function to predict and calculate the error. It applies backpropagation to update weight and bias after the whole CNN operation.

3. Design detail -case study 1: classification based on CNN

3.1 Design the CNN Sentences classification in Matlab

The Matlab tutorial of the CNN model is practical to learn basic knowledge and implement the operations quickly [8], so I built a traditional CNN structure in Matlab. However, the training accuracy result is 70%, which is dissatisfied. Hence, I modified both the structure and arguments to enhance the traditional CNN structure for better performance. For the structure, I added several batch normalization layers, which was put forward by Sergey in 2015, and it aims to normalize input data so that each layer has the same mean value and variance, and it solves gradient disappearance and overfitting. I built a CNN structure with the input layer, convolutional layer, batch normalization layer, Relu layer, maxing pooling layer, fully connected layer, softmax layer and output layer. For the arguments, significant arguments determine the training result accuracy. For example, filter_Size defines the filter's height and weight, which should set as the same value in each convolutional layer. While the numFilter represents the number of filters. In each max pooling layer, I set the same value for pool_Size and stride. After adjusting the argument, I tested a 10000 images dataset that referenced from [3], and I got 99% accuracy after training process. Moreover, the training result shows in Figure 3, where the blue line represents the accuracy.

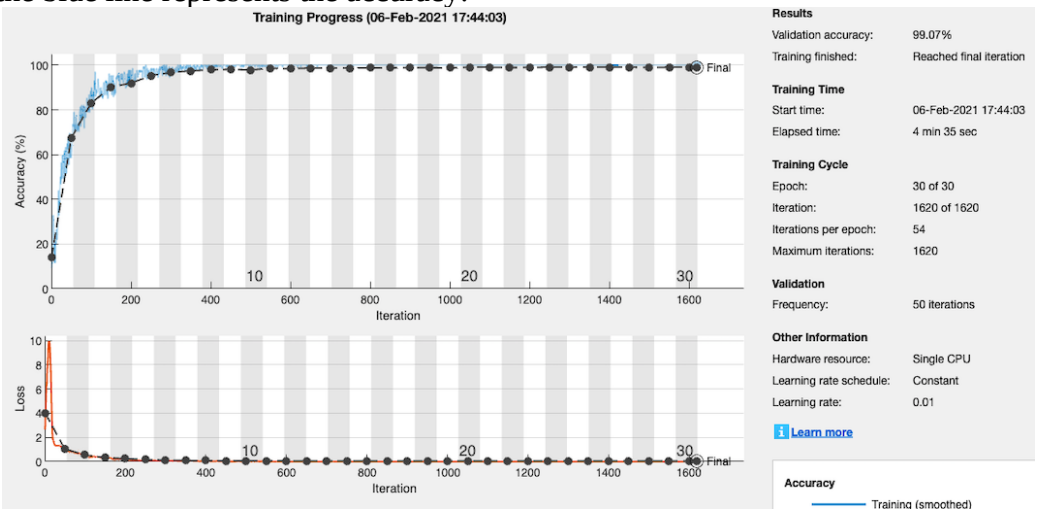

Figure 3: The training result of CNN model in Matlab.

3.2 Design the CNN Sentences classification in Python

I built the CNN structure following Matlab, and I chose the max-pooling method in the pooling layer and the ReLU function in the activation layer, and I applied the combination of the fully connected layer and softmax layer at the end. For the Python syntax, I also refer to the significant arguments, which I learned from Matlab, so that the model could get better performance. For the dataset, I used 5000 lines of sentences containing positive and negative reviews as the whole dataset, and 80% of the dataset is the training set. In contrast, the other 20% dataset is recognized as the test dataset. However, the original sentences cannot be directly used, so I deleted all the special characters and spaces; moreover, it also needs to split the whole sentences into words and label each word with

positive or negative labels. The model requires time to learn and train so that it can classify ideally, and the training result shows in Figure 4. The whole training process takes few hours until the result cannot improve. The blue line in left part shows the accuracy of the CNN model, and the blue line in right part is the loss result. After 3000 iterations, the accuracy remains above 95%, and the loss achieves under 1%. In general, the training result is satisfying.

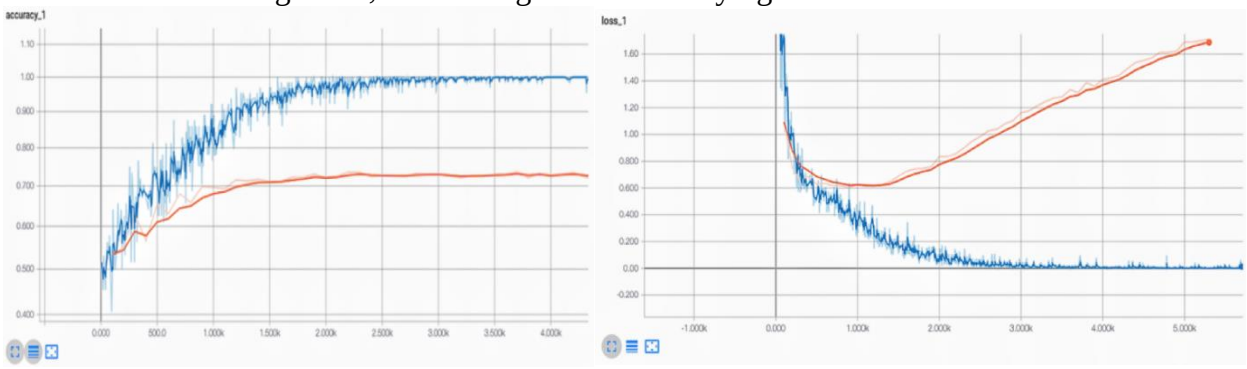

Figure 4: The accuracy and loss result of training process.

4. Testing and verification

After training process, the model shows perfect accuracy result which means that we already got the trained model. Then I test the model to verify the performance, so I randomly pick eight sentences from the test dataset as the input, which contain both negative and positive reviews, and we hope that the trained CNN model classify the sentences into two categories, and I set binary one to represent the positive category and binary zero to represent the negative category, and the test result shows in Figure 5. It is clear that all test sentences are divided into two categories where the upper part sentences are positive comments of the film with the binary one; moreover, the down part sentences have binary zero, and they all negative comment sentences. In conclusion, the CNN model takes few time to be trained, almost satisfying the classification demand.

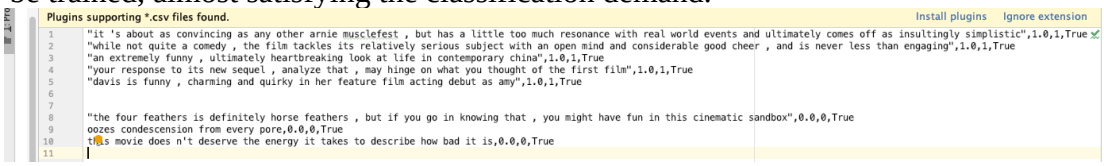

Figure 5: The test result of trained CNN model.

5. Literature review

5.1 Generative Adversarial Network （GAN）

The Generative Adversarial Network(GAN) was first put forward by Ian Goodfellow in 2014[7], and it can generate new data that is similar to the original input data. Actually, the GAN relays the generator and discriminator compete for each other to optimize the result. While the generator can be recognized as a faker who tries to generate a fake banknote based on mimicing real banknote features. While the discriminator is recognized as police who identify the banknote is real or fake. When the discriminator identifies the generated banknote as fake, the generator will keep learning to generate a new fake which is more similar to the real, and both generator and the discriminator improved ability. While the training process stops when the discriminator cannot identify the fake, and it means that the generator generates a perfect fake banknote. However, the GAN has shortcomings which could result in a vanishing gradient and mode collapse problem if the generator or the discriminator

is stronger than the other. In case study two, we build a trained GAN model, and we provide enough images as input dataset. Then, it needs some time to learn the input data features, and it trains both the generator and the discriminator simultaneously. The training process stops when the generated image is almost the same as the input image, and we get a trained GAN model. Finally, we modify the structure by applying subpixel super-resolution (SPSR) in the upsample part of the generator to improve the quality of generated images. The modifications aim to develop the resolution of generated images from low resolution to high resolution.

5.1.1 The structure of GAN

The generator and the discriminator are neural network while I referenced Figure 6 from [6]. The left part is the generator network while it connects to discriminator network. The generator receives input data, and converted into a vector format so that the model can recognize, and it generates a sample that is called a mask, which is sent to the discriminator as the input. The generator optimizes the mask's quality based on the output of the discriminator.While the traditional generator plays a mapping function, where the input vector maps to the training dataset, and the generator applies the deconvolutional operation to convert the input vector data back into 2D image data. The discriminator classifies the generated data, and it outputs the identifying result. The real data is input images, and the fake data results from the generator. Generally, it applies CNN to build the discriminator structure, and the generator keeps generating a better mask by learning the feedback from the discriminator and keep sending the new mask to the discriminator during the training process. Meanwhile, the discriminator learns to identify the real and fake data so that it classify the mask, and it keeps updating the weight. Furthermore, the training process finishes when discriminator cannot identify the mask which means that the generator has created a perfect mask.

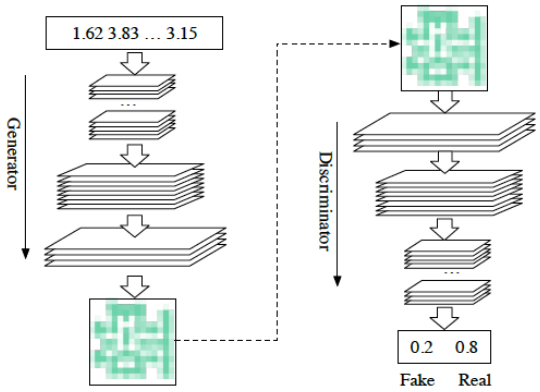

Figure 6: The structure of GAN [6].

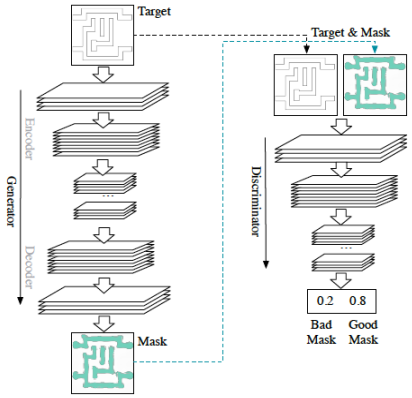

Figure 7: enhancement GAN structure [6].

6. Design details- Case study 2: Regression based on GAN

6.1 Structure of enhancement GAN

For improving the quality of the mask, I referenced Subpixel Super Resolution (SPSR) theory from [6] to enhance the traditional GAN model. The two main enhancement focuses on changing the generator architecture. First, it adds the decoder part after the original generator, which is recognized as the encoder part; hence, the new generator architecture consists of the encoder and decoder part, which seems like a U Net structure. I referenced Figure 7 from [6] to directly show the enhanced generator structure while the original encoder part and discriminator remain, and the target is the real image input. However, the new structure is deeper than the previous GAN, and the gradient backpropagation requires more time. While the U Net is an effective solution since it reduces the loss

and gradient vanishing. In addition, the second enhancement applies SPSR to replace the traditional deconvolution operation in the decoder part. Since the deconvolution operation may cause extra noise and reduce operational efficiency. Moreover, the SPSR is a standard method in solving super-resolution problems. It has been widely applied in dealing with super-resolution image area to get better resolution images. Basically, it contains convolutional layers and periodic shuffling which is a significant structure, and it upscales the size of the original feature map. The computation theory shows in the functions 2, and it is referenced from [5].

$$t^{hr}_{i,j,k} = t^{lr}_{i',j',k'} \quad (2)[5]$$

While i', j', k' are low resolution images where the i'represents i/r, the j'represents j/r, and the k' represents C×r×mod(j, r)+C×mod(i,r)+k. The C is the number of channels, and the H represents the height of the feature map, and the W is the width. While the main purpose of periodic shuffling is to scale the feature map r times larger, and the two steps show in Figure 8. While convolutional layers generate feature map with the original size H×W×C, and the periodic shuffling upscales the original feature map size. If the feature map upscale r, the first layer of periodic shuffling remain the original size, but it generates r^2 channels so that the total size of this layer is H×W× r^2C. All the feature map send to the next layer in one channel, so the size of new feature map is rH×rW×C; hence, the size converts to rH×rW, so it upscale the size r times larger, and the pixel position remains in different feature maps. Appling SPSR has the benefit of improving the operation speed since the convolution kernel scan fewer channels, and the noises of the mask reduced.

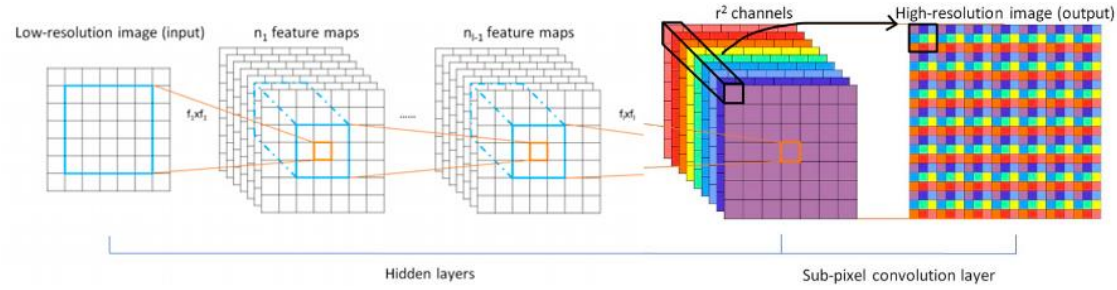

Figure 8: The process of SPSR[4].

6.2 Image regression based on enhanced GAN

I implemented the traditional GAN model in Python, and I applied 2D convolution layers in the downsample part of the generator. It connected with the ReLU function and batch normalization layers for better performance. Moreover, I called Conv2D transpose function to build upsampling part of the generator, it completes a deconvolutional operation. Furthermore, the discriminator structure uses several 2D convolution layers. Then, I used five hundred images as the training set to train the model, and I leave the model a few times for learning and training. The training process shows in Figure 9, where the epoch represents the number of iterations. The training process will stop when the 80 epochs complete, and the regression predicting image of shows in Figure 10(a).

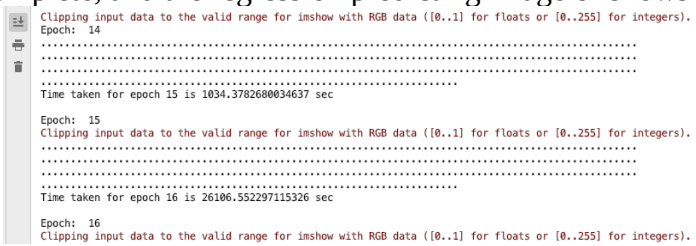

Figure 9: The training process of traditional GAN model.

I modified the traditional GAN by adding SPSR to the upsample part of the generator while the downsample part remains 2D convolution layers. The SPSR function converts the image shape into [batch_size, width, height, number of channels] which is convenient to operation periodic shuffling operation. The r is the upscale number of the image which is represented by shape[1]*=self. r in Python. It operates the SPSR to convert the input shape from [batch_size, width, height, number of channels $\times r^2$] to [batch_size, width $\times$r, height $\times$r, number of channels]. We call the 2D convolution layer function and the SPSR function when we build upscale architecture. For the image, I set the input data IMG_WIDTH=256,IMG_HEIGHT=256,OUTPUT_CHANNELS=3. The discriminator structure still uses 2D convolution layers, ReLU function and batch normalization layers. The enhancement GAN model uses the same dataset to train. While Figure 10 (a) is the traditional GAN model regression result, and Figure 10 (b) is the result of the enhancement GAN model with SPSR. The ground truth is the final aim that the model predict, and the predicted image is the current predicted result in the training process. While the ground truth and predicts image of the enhancement GAN model are more precise than the traditional model; moreover, the enhancement model's predict image format is similar to the output of the SPSR layer, shown in previous Figure 8. It means that the enhancement GAN model shows better performance in generating high-resolution images. However, the predicted image of the SPSR model is dissatisfied since it needs loop training for better results, which we will improve in the future work section.

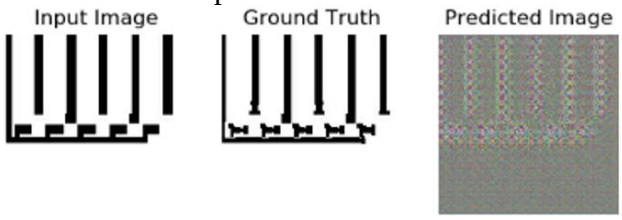

Figure 10(a): Traditional GAN model result.

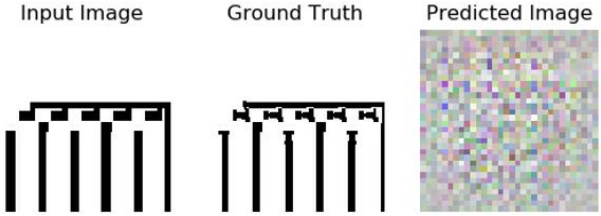

Figure 10(b): Enhancement GAN model result.

7. Testing and verification

After training the GAN model with 500 images and 80 epochs, the GAN model has been trained well. It means that the test dataset should verify the model performance. I took five images as the test dataset; moreover, the test result shows in Figure 11. While the ground truth is the final expected result of the model if the training iteration enough, and the predicted image is the current generated image with fewer training iteration. It shows that the current generated image is generally similar to the original input image, meaning that the GAN model works. However, the performance of the current generated image is not perfect since the training iteration is not enough, and the generated image would be closer to the original input when the model has sufficient training time.

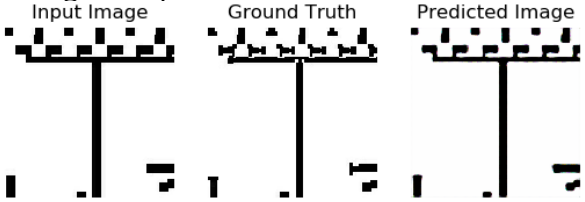

Figure 11: The test result of GAN model.

8. Conclusion

In conclusion, the Convolutional Neural Network(CNN) and the Generative Adversarial Network(GAN) are subfiles of deep learning; moreover, deep learning is also a subfile of machine learning. To enhance the training model, modifying the network structure and modifying the network

arguments are effective methods. The Batch Normalization (BN) layer can improve the accuracy and can speed up the CNN model. Moreover, Subpixel Super-Resolution (SPSR) shows better performance in dealing with super-resolution tasks.

References

[1] Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 3642–3649). https://doi.org/10.1109/CVPR.2012.6248110

[2] Deshpande Adit. (2016). A Beginner's Guide To Understanding Convolutional Neural Networks. Github, 19, 1–11. https://doi.org/10.1097/MD.0b013e31822403e9

[3] Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database (pp. 248–255). Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ cvprw. 2009.5206848

[4] Davis, Ian. 2009. "The 'super-Resolution' Revolution." In Biochemical Society Transactions, 37:1042–1044. doi: 10.1042/BST0371042.

[5] Foroosh, H., Zerubia, J.B., Berthod, M., 2002. Extension of phase correlation to subpixel registration. IEEE Transactions on Image Processing 11, 188–199. doi:10.1109/83.988953

[6] Li, X., Chen, L., ... Tong, W., 2019. SCGAN: Disentangled Representation Learning by Adding Similarity Constraint on Generative Adversarial Nets. IEEE Access.7, 147928–147938. doi:10.1109/ACCESS.2018.2872695

[7] Pan, Z., Yu, W., ... Zheng, Y., 2019. Recent Progress on Generative Adversarial Networks (GANs): A Survey. IEEE Access 7, 36322–36333. doi:10.1109/ACCESS.2019.2905015

[8] MathWork, 2020. Create simple image classification network using deep network designer. https://www. mathworks.com/help/deeplearning/gs/create-simple-image-classification-network-using-deep-network-designer.html

[9] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation, 1(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541