Deep Learning Lab - Final Project Report

2019142002 Youjin Song

[a] *Yonsei University, Department of Electronic and Electrical Engineering,*

Abstract

This paper propose a method to detect *Out-of-Distribution(OOD)* in MNIST classification task. By simultaneously training denoising autoencoder and classifier model for in-distribution data and adding loss functions to have even probability for wrong prediction, the model was able to successfully detect unfamiliar data during validation process.

1. Introduction

Deep Neural Network has enormously enhanced the performance in various computer vision tasks such as classification and object segmentation. Fundatmentals behind the achievement is deepening of the layer, which allowed the model to learn feature vector that effectively imply semantic information of a given input. While well-trained Deep Neural Network shows excellent performance in tasks with familiar data, in real world situation, we have less control over range of various data the models would be exposed to. In that case, it is logical for the model to notice the unfamiliarity and perform poorly for learned task, because it has never been trained to perform the task for that alien image. However, it has been shown that the well-trained model often shows high confidence for unfamiliar data as well which might possibly cause serious problem in real world application. Therefore, *Out-of-Distribution(OOD)* detection task aims to detect whether an example is in error or abnormal for not being used during the training process.

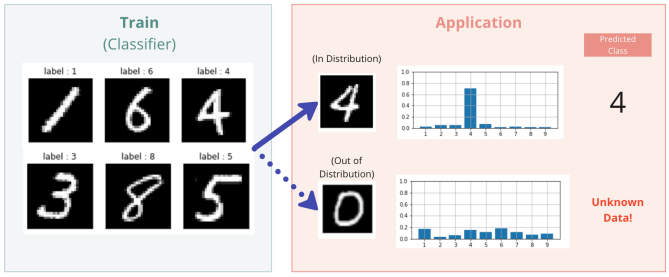

Figure 1: The model should be aware of and and correctly indicate the fact that data('Zero' instance in the figure above) that has not been used during training process.

The goal of the project is to perform the OOD detection task in MNIST classifier model. First training a classifier only with '1' to '9' labeled image, and then putting images including '0', the trained classifier should correctly figure out whether the given image is from in- or out-of- distribution. This report first briefly summarizes some previous works that is related to detecting OOD (section 2) and then explains the model's architecture and algorithm in detail (section 3). Then, experiments results are provided(section 4), along with several ablation studies to show how each component of designed model has contributed to the performance.

2. Related Works

2.1. OOD detection tasks

There were several attempts to successfully detect OOD. Hendrycks and Gimpel (2016) first brought the idea of the task and introduced a simple baseline that uses softmax scores. If the maximum probability from softmax is lower than certain threshold, the data is likely to be out-of-distribution. They utilized the Area Under the Receiver Operating Characteristic curve (AUROC) metric for evaluation the detection performance. Liang et al. (2017) suggested effective and simple method to detect OOD data samples, by *temperature scaling* and *adding small perturbation*s. By scaling output with certain temperature value T before putting it into Softmax computation, it polarizes the softmax logit from in-distribution and OOD. Lee et al. (2017) concentrates more on the *training* process compared to the previous mentioned works, which mainly focused on *inference* after training. They added *confidence loss* that induces unfamiliar data to have uniform-distribution-like softmax scores. Moreover, they also added OOD data(generated from GAN) that is very similar to in-distribution data.

2.2. Autoencoder

An autoencoder is a special type of neural network that is designed to gain representation in *unsupervised* manner. The most basic auto-encoder has bottleneck structure two main parts : encoder and decoder. Encoder tries to compress the knowledge representation from input while decoder tries to recover the original input from the extracted feature. Since it does not require data label during training, reconstruction error between original input and reconstructed output is used for network optimization. There has been some works that utilized the idea of autoencoder for anomaly detection. Zhou and Paffenroth (2017) used deep autoencoder for eliminating outliers and noise without access to any clean training data.

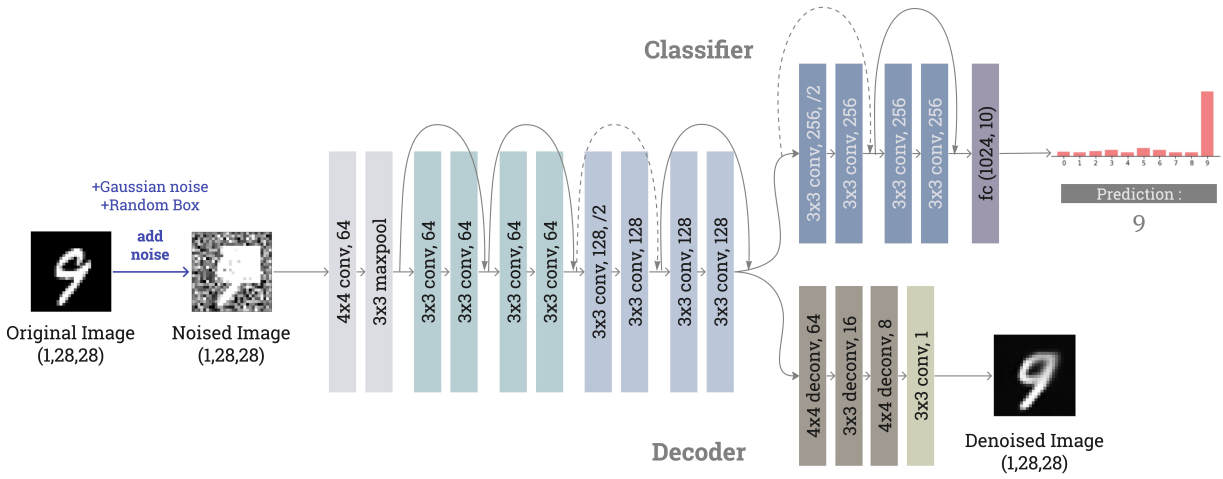

Figure 2: Model Architecture

2.3. Focal Loss

Lin et al. (2017) introduces a novel but simple loss called *Focal Loss* function to solve class imbalance problem commonly faced during object detection tasks. Their modification came from the idea of putting more weights on fewer, difficult examples (low prediction target probability $\hat{p}_t$) and down-weighting to well-classified, easier samples.

$$L_{\text{focal}}(t, \hat{\boldsymbol{p}}) = -\sum_{i}^{C} \mathbb{1}_{t=i}(1 - \hat{p}_i) log(\hat{p}_i) \tag{1}$$

Denote that t is a target class label, $\hat{p}_i$ is a probability of an object classified as class i. If $\hat{p}_t$ is small, it means that the model has less confidence for classifying the object as correct target. Therefore, by multiplying $(1 - \hat{p}_t)$, more difficult examples with lower $\hat{p}_t$ would be considered more significantly than easier examples with higher $\hat{p}_t$. By simply multiplying this single term at commonly used Cross Entropy loss equation, $L_{\text{CE}}(t, \hat{\boldsymbol{p}}) = -\sum_i^C \mathbb{1}_{t=i}\ log(\hat{p}_i)$, it successfully avoided easy negatives(Background, for example) from overwhelming during training process.

3. Proposed Method

The major assumption for the method is that OOD data should show unbiased, even score(logit) distribution from trained classifier. Fairly ideal case is shown in figure 1 (from introduction) : '4' is from in-distribution, so it has high confidence for its prediction whereas '0' is out-of-distribution, so it shows evenly-low scores for predicting as each classes. Although there is certain maximum point —'6' is the highest prediction score here— the model is not confident enough so it should be classified as out-of-distribution. Therefore, the proposed method tries to make evenly-low scores for unconfident input image by loss functions. If it gets to do so, we would be able to successfully identify OOD by thresholding by the maximum logits — say, 0.3, in this example.

Another assumption is that trained auto-encoder would exclusively perform well(denoising in this example) for in-distribution data, but not for foreign data. Reconstruction loss between original and denoised image could be another criterion for distinguishing in/out distribution data.

Taking these into account, the method tries to train classifier that well-distinguishes within given train data but not being *too confident* about difficult samples. At the same time, the autoencoder must be able to perform its denoising skill for learned images, but perform less well for foreign images.

The task goes through two main phases. The first phase is to train the deep neural network using 1 to 9 labeled MNIST dataset. Here, the model is trained to simultaneously optimize auto-encoder and classifier. The second phase is to find optimal threshold of classification scores(softmax logit) and reconstruction error(MSE loss) that could effectively split between in/out distributions, by using validation set. Validation set, of course, only has image labeled with 1 to 9, but some denoised images that were reconstructed from intensely noised input, can be utilized as out-of-distribution data. More details are explained in sectiont 4.

3.1. The network Architecture

As shown in figure 2, the network is consist of 2 main part : **classifier** that aims to correctly match predicted value with data label and an **auto-encoder** that tries to reconstruct the noised image into the original image. The model uses structure from resnet 18 (He et al. (2016)), but the last layer (with 256 channels) are omitted, since the given MNIST dataset with 1×28×28 size is too small for given kernel sizes, nor deepening more layer gave meaningful improvement in its performance. The first 10 layers are shared for classifier and encoder part of auto-encoder. This means that the deepest shared feature embedding

would contain both higher-level semantic information for classification, and some lower-level information for reconstruction in the end. In fact, it was necessary for auto-encoder to have sufficient depth, since denoising must not be limited to lower-level task like simple de-blurring.

3.2. Adding noise for Auto-encoder training

In this model, auto-encoder functions as denoiser - remove gaussian noise and random square. Since the random square was generated to heavily occlude the given image, the auto-encoder should understand some amount of semantic information for quality recovery. Adding only gaussian noise or having smaller square that does not significantly harm the original image so it did not improve semantic understanding of the model.

After all the experiments, the noised '0' label data was fed into the trained auto-encoder, and the it actually seems like reconstructing the image considering high level semantic information as shown in figure 3.

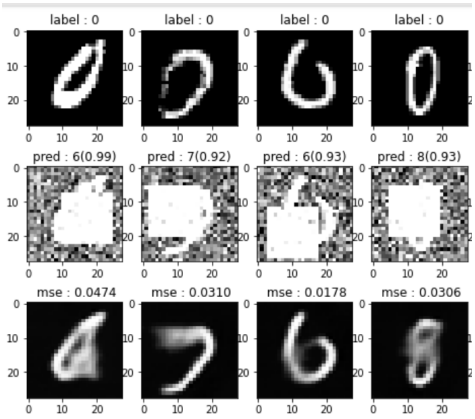

Figure 3: The top row is the original input image, middle row is noised image (gaussian noise and random square where its size ranging from 10 to 14) and the bottom is the denoised output from auto-encoder that has trained with 1 to 9 data.

The reconstructed image resembles one of the in-distribution images. It is noteworthy that it also somehow matches its classification prediction — if it predicted class of the noised image as '6', it reconstructed the image as if they are actually '6'.

3.3. loss functions

The model is optimized by minimizing 3 loss functions. One loss function for auto-encoder is quite straightforward : MSE(Mean Square Error) loss is computed between original image(without noise) y and denoised output $\hat{y}$.

$$L_{\text{den}}(y, \hat{y}) = \frac{1}{N} \sum_{i,j}^{N} (y_{ij} - \hat{y}_{ij})^2 \tag{2}$$

The other two loss functions are for training classifier. It tries to obtain accurate classification prediction but avoid being over-confident during training. To be more specific, those two loss functions were specifically designed to have *biased, high* logits for confident samples and *evenly-low* scores for difficult samples.

As a baseline of improving classification accuracy, Cross Entropy Loss $-\sum_{i}^{C=10} \mathbb{1}_{t=i} log(\hat{p}_i)$ is adopted. The loss penalizes when score of target class $\hat{p}_t$ is close to 0, so induces the classifier to make the score higher for target.

However, since it almost ignores scores predicted for other classes, we adopt the similar logic from *Focal Loss* (Lin et al. (2017)), but in opposite way. Instead of penalizing more for unconfident(difficult) correct-class score as in equation 1, this method tries to penalize more for overconfident wrong-class score. Coefficient k is multiplied to this additional term, and from experiments, k around 1 best optimizes in intended way.

$$L_{\text{focal}}(t, \hat{\boldsymbol{p}}) = -\sum_{i}^{C=10} \mathbb{1}_{t=i} log(\hat{p}_i) + k\, \mathbb{1}_{t\neq i}(\hat{p}_i) log(1 - \hat{p}_i) \tag{3}$$

The last loss function, named *StdLoss*(Standard Deviation Loss) is introduced for making the scores even. In a similar vein, Lee et al. (2017) has adopted KLLoss to make distribution similar to uniform distribution. We apply this loss for only instances that has made wrong classification decision.

$$L_{\text{std}}(t, \hat{\boldsymbol{p}}) = -\mathbb{1}_{t\neq \texttt{pred}} \textbf{std}(\hat{\boldsymbol{p}}),\ \texttt{pred} = \underset{i}{\text{argmax}}\ \hat{p}_i \tag{4}$$

By introducing the loss functions in addition to Cross Entropy loss, the score distributions has changed as shown in figure 4. OOD data was added for this evaluation, to see how the logits of OOD would be distributed.

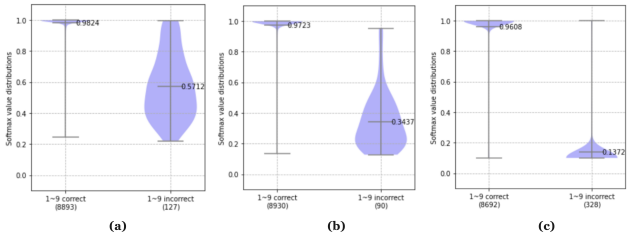

Figure 4: Score(Maximum Softmax Value) distributions when (a) only Cross Entropy Loss was used for optimization(coefficient to additional loss term is set to 0) (b) coefficient is 1 (c) coefficient is 5.

While the correctly classified data still has high confidence, adopting those losses lowers the confidence for wrong predicted data. However, if std loss has overly larger coefficient, classification accuracy decreases, so it is important to set adequate weight to the term.

3.4. Thresholding

After the first training phase, the model is now able to classify in-distribution data with very high accuracy($\sim$ 99%). We have already trained the model to have low and evenly distributed scores in difficult samples, so maximum softmax score for foreign data would be lower than in-distribution data. However, when the model is exposed to OOD, it stills outputs prediction with the highest softmax score. Therefore, we should set certain threshold : i.e. if the maximum softmax score is below certain point, its softmax scores are ignored and decided as OOD.

Determining the optimal threshold is now directly related to the ultimate accuracy. This is where validation data is needed. By putting in-distribution 1 ~ 9 data with some OOD data, we can draw ROC curve depending on the thresholds, and then figure out the best threshold that distinguishes data from in- and out-of-distribution. Now, another problem rises : where is that OOD data? Of course, random noise that has same size with the input (1,28,28) is certainly an OOD, but it would too trivial(it would not help setting best threshold). We should look for data that is very close to in-distribution data, but actually not.

Noticing that our trained auto-encoder outputs image from noised input image, we can easily earn OOD that looks foreign with slight modification. By putting rotated digit image (90 or 270 degree, but not 180 since some digits like 8 looks the same when rotated 180 degree) or occluding the digit even heavier with larger square, the auto-encoder is able to generate some untrivial but clearly OOD images.

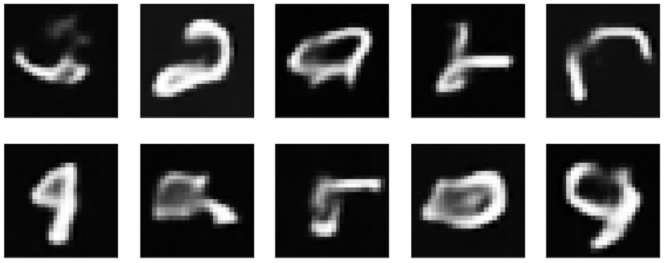

Figure 5: Denoised image from heavily noised data(rotated-noised digit) can be considered as OOD.

Finally, considering that OOD has larger reconstruction loss, minimum reconstruction loss for OOD can also be a criterion for determining whether given data is from in/out distribution.

4. Results

Among 54077 MNIST images(label 1 to 9), 5000 images were not used for training so that it could later be used on validation process. Following the pipeline described in previous section 3, the network is trained with the 49077 train data.

4.1. Performance of Denoiser

Qualitative denoising performance of the denoiser model is shown in figure 6. Those samples were randomly extracted, and we could easily confirm that there is significant disparity in denoising ability between in and out-of-distribution data. Quantitatively speaking, when sorted the top 200 MSE loss by its label, most of the data that has worse reconstruction error was from out-of-distribution (figure 8). From figure 7, out-of-distribution data is showing larger MSE value than other digits. digit '1' is easier for reconstruction thanks to its simpler shape, whereas other digits often shows larger MSE value in average.

4.2. Setting Threshold

The final step is to set final threshold of maximum softmax scores. Plotting ROD curve as above, setting 0.659 gave the

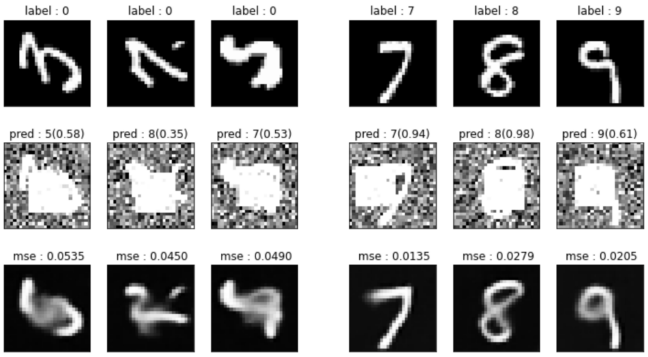

Figure 6: The left three columns shows generated OOD data and its reconstructed image and the right three columns are from in distribution data.

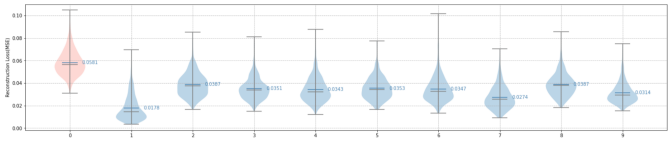

Figure 7: MSE loss distribution for reconstruction by each labels. Out-Of-Distribution data are all labeled as 0.

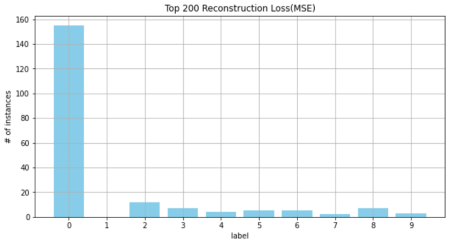

Figure 8: MSE loss top 200 at validation

best accuracy for detecting OOD and classification accuracy within In-distribution data. Since the model has now set the threshold for detecting OOD, if its maximum softmax score is below the threshold, its class is now determined as 0 regardless of its previous softmax scores.

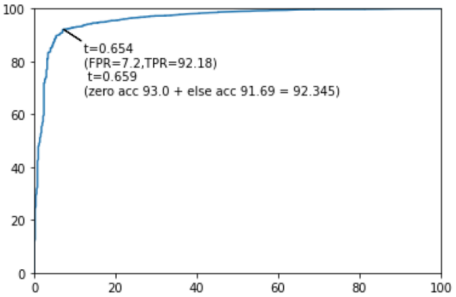

Figure 9: ROC curve by thresholding maximum softmax value. Setting

Moreover, minimum MSE loss value from generated OOD from validation dataset is set as auxiliary criterion for deciding OOD. By setting the threshold by all this process, **OOD detection accuracy(zero accuracy) was 96%, and Classification**

accuracy(else accuracy) 90% .

5. Conclusions

The proposed method polarized softmax scores by adopting two loss functions during training process, and set reasonable threshold of maximum softmax scores that distinguishes OOD from in-distribution data. Auto-encoder that were simultaneously trained and shared deep features with the classifier provided some meaningful OOD images for setting such thresholds. There are some possible weakness in this method. The generated samples (during validation process)from trained auto-encoder still might be somehow trivial — it might be far from in-distribution. Moreover, for more complex input data with larger size or containing more semantic information, auto-encoder might not perform well in higher level. Nevertheless, the overall process was able to detect unfamiliar data while successfully performing classification task for learned data.

References

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Hendrycks, D., Gimpel, K., 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136 .

Lee, K., Lee, H., Lee, K., Shin, J., 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. arXiv preprint arXiv:1711.09325 .

Liang, S., Li, Y., Srikant, R., 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. arXiv preprint arXiv:1706.02690 .

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988.

Zhou, C., Paffenroth, R.C., 2017. Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 665–674.