Report of Task 3

MoeKid101

1. Introduction

In task two we've had some idea about the training of CNN on CIFAR-10, the generalization of artificial neural networks, and the class imbalance problem. Now in this task we turn to a simpler dataset, namely MNIST, and compare the performance of CNN and RNN, with class imbalance taken into account.

2. Baseline Models

We first present our baseline models including both the CNN baseline and RNN baseline. The CNN model has two convolution layers and two dense layers; the RNN model has two layers with hidden layer size 100, and the output of the last LSTM cell is directly connected to a fully connect layer whose output is exactly the predicted output.

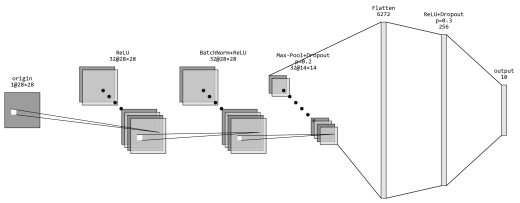

Figure 1. Architecture of baseline CNN model

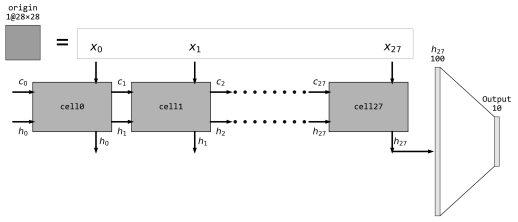

Figure 2. Architecture of baseline RNN model

On these two baselines, I simply used the original dataset (without data augmentation) and obtained an accuracy of 99.15% for CNN and 98.39% for RNN. These results are high enough for us to choose as baselines.

2.1. Data Augmentation

On MNIST, I took rotation with a random angle in $[-25°, 25°]$ as the only way to augment our data. The reason was that both CNN (which has square receptive fields) and RNN (which has 1×28 horizontal line receptive fields) are not designed to have the inherent ability to cope with rotations. Therefore, rotation forces the model to learn some robust features. An example is given in Fig.3.

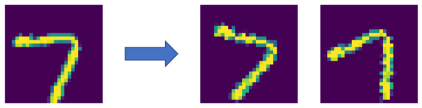

Figure 3. Example of random rotation

3. Class Imbalance

Class imbalance, a problem in neural networks defined as the phenomenon that collected training data of some labels have significantly more samples than that of other labels, have been covered in task 2. We would still use the definition of value ρ to describe how imbalanced the training set exactly is:

$$\rho = \frac{\max_i |C_i|}{\min_i |C_i|}$$

where $|C_i|$ is the number of samples in class i.

In task 2 we have had some experience on a data-level method *random over sampling* (ROS) and concluded that adopting ROS requires the model to have a moderate capacity, and a suitable generalization technique to be deployed. In this task we will cover an algorithm-level method, namely weighted loss, too.

Meanwhile, suppose A is the collection of majority classes while I is the collection of minority classes; $\ell(\boldsymbol{x})$ and $m(\boldsymbol{x})$ are the label and the model prediction for sample $\boldsymbol{x}$; $|C_A|$ is the number of samples in majority classes while $|C_I|$ is that in minority classes. Then we define four metrics:

$$p_{\alpha \to \beta} = \frac{1}{|C_\alpha|} \sum_{\boldsymbol{x} | \ell(\boldsymbol{x}) \neq m(\boldsymbol{x})} \mathbf{1}\{\ell(\boldsymbol{x}) \in \alpha\} \times \mathbf{1}\{m(\boldsymbol{x}) \in \beta\}$$

where $(\alpha, \beta) \in \{A, I\}^2$. (e.g. $p_{A\to A}$ represents the probability of classifying a majority class sample to a wrong majority class given the label belongs to a majority class.) Notice that such definition is nothing else than providing a summary of the confusion matrix. We define these four metrics because the confusion matrix provide too much detailed information which is far beyond what we need.

3.1. Effect on Baseline Models

As is shown in task two, it is expected that images belonging to the minority classes are less concerned by the baseline models, and therefore the performance of our model on minority classes will drop.

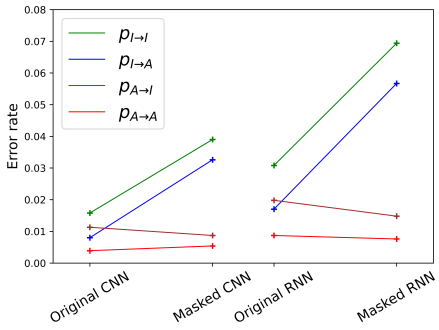

Figure 4. Effect of class imbalance on baseline models

From Fig.4 we see that both CNN and RNN demonstrated similar performance degradation: $p_{I\to I}$ and $p_{I\to A}$ increased a lot while $p_{A\to I}$ and $p_{A\to A}$ remained almost the same, which meets our expectation. What's more, we have an empirical conclusion for imbalanced datasets:

Conclusion 1: It's almost sure that $p_{I\to I} > p_{I\to A} > p_{A\to I} > p_{A\to A}$.

Reason: For imbalanced datasets, we're always having less knowledge about the distribution of I than that in A, and more knowledge leads to lower error rate.

It's expected that all the four probabilities are almost the same on an ideal dataset, while class imbalance introduce gaps between the four values. Therefore, to solve class imbalance problem as much as possible, we should try our best to reduce the gaps.

3.2. Random Over Sampling

My first trial ended in failure: ROS brought about no improvement on the original dataset. But after I introduced data augmentation, ROS worked, improving the overall accuracy to 98.61%.

Having confirmed the effectiveness of ROS, I took a more detailed look at ROS. ROS was conducted until $1/\rho = 0.2, 0.4, 0.6, 0.8, 1$ respectively and the four metrics were plotted. From Fig.5 and Fig.6 we know that if we are not so particular about accuracy, we could simply stop at

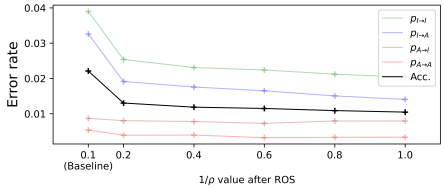

Figure 5. Effect of ROS on CNN given different ρ values

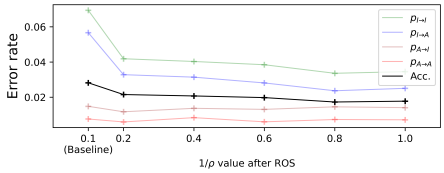

Figure 6. Effect of ROS on RNN given different ρ values

$1/\rho \approx 0.2$, a value that would ensure us a relatively good performance over the whole test set. However, the defined four metrics tell us something more. ROS helps us bridge the discrepancy between $p_{I\to A}$ and $p_{A\to I}$ with increasing $1/\rho$, which is exactly a part of our goal. Therefore we have such a conclusion about the usage of ROS:

Conclusion 2: ROS should be carried out until all the classes were evenly distributed, i.e. $\rho \approx 1$.

However, we see from the figures that $p_{I\to I} - p_{I\to A}$ and $p_{A\to I} - p_{A\to A}$ increased. When $1/\rho$ finally reaches 1.0, the four metrics demonstrates almost equal differences. Such an observation might offer an upper bound of the performance of ROS, indicating that we can't expect ROS to recover complete information about the original data set.

3.3. Weighted Loss

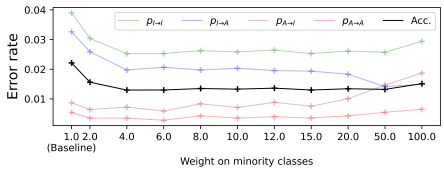

Figure 7. Effect of weighted loss on CNN

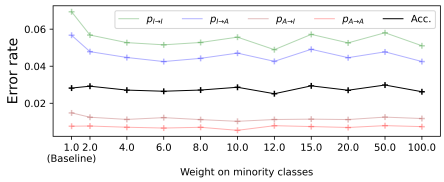

Figure 8. Effect of weighted loss on RNN

Among the algorithm-level methods, I adopted weighted loss, the most simple and straightforward one. Similarly, I

found that simply adding weights to the final losses won't help, and weighted loss also requires augmentation to take effect. Then as it is the case with ROS, I set different weight values on minority classes and plotted the four metrics corresponding to different weights.

As is shown in Fig.7, weighted loss helps CNN reduce the gap $p_{I\to A}-p_{A\to I}$ just as ROS does. With a high enough weight we can even reverse the order of these two metrics. However, it seems that the reduction on $p_{I\to I}$ is not as large as what ROS does, and the overall accuracy is lower than the accuracy of ROS models. Furthermore, the most serious problem of weighted loss is demonstrated in Fig.8, it simply failed to take effect on our RNN model, and I couldn't even figure out why.

Comparing these results with that obtained by ROS, I have another conclusion as follows:

Conclusion 3: Random over sampling is likely to be a safer solution compared with weighted loss.

4. CNN and RNN

I had always held the belief that CNN was used on images while RNN was used for sequential tasks, but in this task I witnessed the power of RNN in image classification. Therefore, in this section we turn to the CNN and RNN architectures and discuss their differences as well as their common points.

4.1. Parameter Sharing (common)

There is one thing in common about CNN and RNN: both architectures feature parameter sharing.

In convolutional neural networks, each convolution channel represents only one filter applied to the whole image. It somehow resembles humans' vision: detecting local information (e.g. sharp edges), texture information and global information (e.g. semantics) through layered filters.

In recurrent neural networks, each input $\boldsymbol{x}_n, (\boldsymbol{h}_n, \boldsymbol{c}_n)$ goes through the same calculation in the cells to obtain $(\boldsymbol{h}_{n+1}, \boldsymbol{c}_{n+1})$. The capacity of the RNN architecture exists in the complexity of the mapping within each cell.

Both these architectures are sharing parameter to execute certain tasks (image feature extraction and memorization).

4.2. Parameter Efficiency (difference)

When we refer to parameter efficiency, we are actually talking about the number of parameters these two architectures need to obtain a certain level of performance. In this task, RNN is undoubtedly the architecture with better parameter efficiency.

From my perspective, the reason is that when we train an RNN model, we are demanding our model to compress information about input and remember it with a single vector (100 dimensional in our architecture), and the exact way to do compression is left to the model. However, only the pooling part in convolution layers in CNN is doing the compression job. The way of compression is not-learnable and obviously not so efficient.

As a result, we must have strong dense layers to handle extracted features ($32 \times 14 \times 14$ dimensional vector in our architecture) and transform it into values related to probabilities of the input belonging to each class. It is exactly the dense layers that requires the largest number of parameters.

Therefore, we can conclude that RNN is more likely to be an architecture with higher parameter efficiency than CNN. Despite the error rate of our CNN baseline is about $2/3$ that of RNN baseline, a single CNN model takes $6,322$KB storage while an RNN model takes only 524KB.

4.3. Network Depth (difference)

Different from CNN whose depth is determined by humans and won't change, RNN architecture has a variant depth dependent on the length of input sequence. For example, in our task, RNN has a depth of 28, which is very very deep compared to our CNN model.

The depth of RNN led to differences in training. My experience was that a learning rate of at least 0.01 is required to make sure the model obtain an accuracy of $> 95\%$ within 10 epoches. In contrast, CNN accomplishes this target with only one epoch if the learning rate is 0.01.

5. Summary

In this task, we explored class imbalance on MNIST and compared CNN with RNN. We defined four metrics to quantify our results and draw conclusions. Although these metrics are only meaningful in the context where we can divide class into majority ones and minority ones, the intuitions they provide make sense even in the broadest sense of class imbalance. My conclusions are listed below:

- Generally speaking, $p_{I\to I} > p_{I\to A} > p_{A\to I} > p_{A\to A}$.
- Solutions to class imbalance provides a reduction in $p_{I\to A} - p_{A\to I}$ while increases $p_{I\to I} - p_{I\to A}$ and $p_{A\to I} - p_{A\to A}$.
- Random over sampling is often a safer solution than weighted loss.
- CNN and RNN both have shared parameters, but RNN is more parameter-efficient. Meanwhile, RNN is very deep, which might cause slow training.