Semantic Segmentation Competition Report

A1751743 Chunyu Zhang
A1742769 Haonan Zhang

1. Baseline mode

After making complex problems, it can be found that taking a simple model as a base version and then adding on top of it makes the final product better.

But when it comes to solving complex problems, simple models often don't solve all problems. Simple models ignore a lot of important supplementary information, for example, simple models ignore the order of statements and sometimes change the relationship between variables. When it comes to making things that require meticulousness, most simple models are not enough, and their too basic properties make them unable to keep up with the latest technology.

Different baseline models are used in different types, and linear regression refers to making predictions from a series of continuous values. Logistic regression, when classifying structured data and language, the logistic regression model will produce results relatively quickly. Boost the decision tree. In the prediction processing related to time or general structured data, it is necessary to improve the accuracy of the decision tree by lifting the gradient. The simple brain gyrus framework, proper adjustment of some variables and then training, has a great effect on image classification and detection. Baseline models tend to take very little time to apply but can do a lot of work.

2. Employed more than three tricks with ablation studies to improve the accuracy

In this assignment, we used data augmentation, Architectural changes and change the learning rate and batch size to improve accuracy.

2.1 data augmentation

We first run with the baseline model (the given code), and then add the data augmentation code. We believe that resizing the image and zooming in on the entire image can bring more pixels for comparison, which can improve the accuracy. But without running the code, it is found that when the image is too large, it will cause extremely slow running speed. At the limit of accuracy and running speed, we choose to improve the accuracy a little bit in exchange for fast running.

2.2 Architectural changes

The second point is Architectural changes, most deep convolutional neural networks consist of a set of key base layers, we tried to optimize this layer a bit. For this part we decided to make the network wider. Because in a deep learning network, the width determines the probability of finding the global optimum. When the network becomes wider, it means that it is easier for the program to find the global optimum, so the accuracy rate is easier to improve. That's why we widen network.

2.3 learning rate and batch size

The third point is the learning rate and batch size. For these, too large or too small will affect the correct rate. If the learning rate is too large, the model will not converge, and if it is too small, the model will converge slowly or fail to learn. Furthermore, although model performance is not so sensitive to the effect of batch size, batch size becomes a very critical parameter when further improving model performance. For the same number of epochs, reduce the number of batches required for large batches, thus reducing training time. On the other hand, gradient computations with large batch sizes are more stable because the model training curve will be smoother. When fine-tuning, you might get better results with larger batches.

Then we get the result:

Trick1	Trick2	Trick3	Accuracy
N	N	N	28.44%
Y	N	N	30.15%
Y	Y	N	31.39%
Y	Y	Y	31.88%

Trick1: Code augmentation: resize.

Trick2: Architectural changes: use wider network.

Trick3: Change of advanced training parameters: Learning Rate, Batch Size.

3. improve the trade-off between accuracy and efficiency

When dealing with the trade-off between improving accuracy and efficiency, the difficulty lies in having to choose which features are most important in each image. This largely relies on human judgment and long-term debugging error handling to decide which feature can distinguish different types of objects. DL lets the machine obtain a dataset of images that have been labeled with object types, and then is trained on the given data to automatically find the most descriptive and obvious features. But DL is sometimes overkill, traditional algorithms can solve a problem with simpler, less code they can be very general and perform the same on various images. In contrast, the features learned by DL can only be based on the training set and cannot perform well in images other than the training set. So, mixing manual tuning methods and DL can achieve better performance, mixing the advantages of both methods, and quickly dealing with problems in systems that require high performance.

4. Explanation of the code implementation

4.1 data augmentation

In the implementation of the data augmentation code, we adopted a series of basic data enhancement operations. When processing the size, we originally changed the resize to increase the size by 1.125 times, so that there will be more pixels for comparison, which can improve the accuracy rate. But too large size will greatly increase the running time, so only increase the size by 1.125 times.

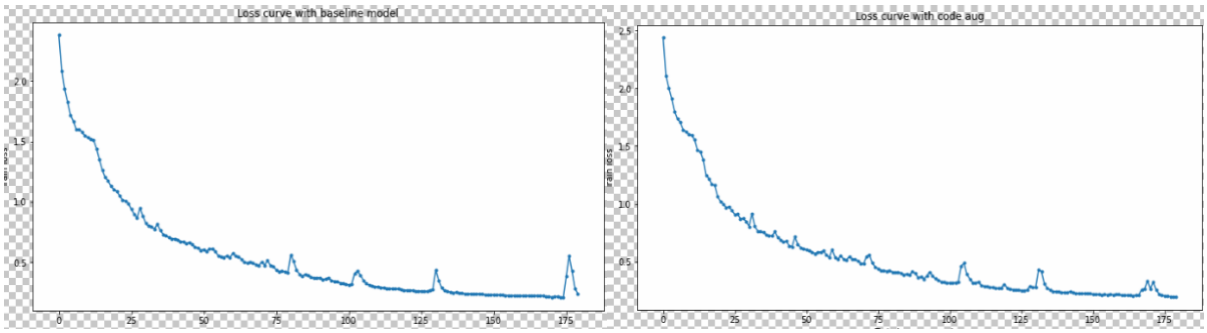
4.2 Architectural changes

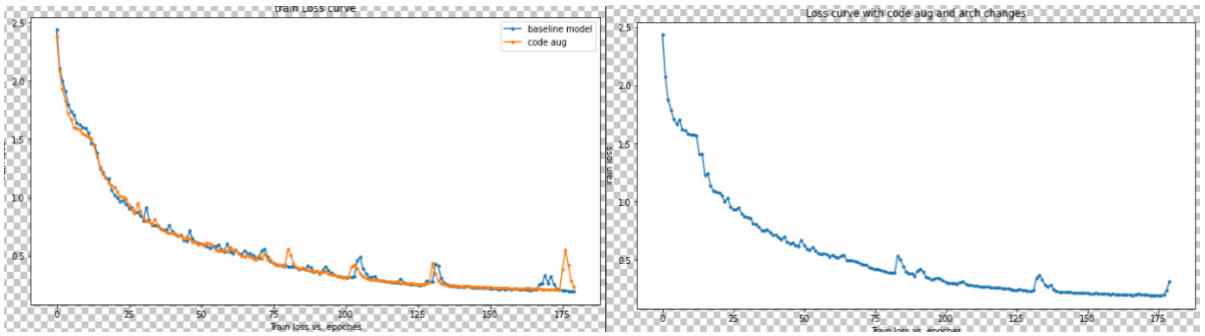
When implementing Architectural changes, we change the pixels in conv2d to a certain extent, 64 becomes 8*8, 256 becomes 16*16, 512 becomes 28*28, which can make the network wider. When the network becomes wider, it means that it is easier for the program to find the global optimum, so that the accuracy rate is easier to improve.

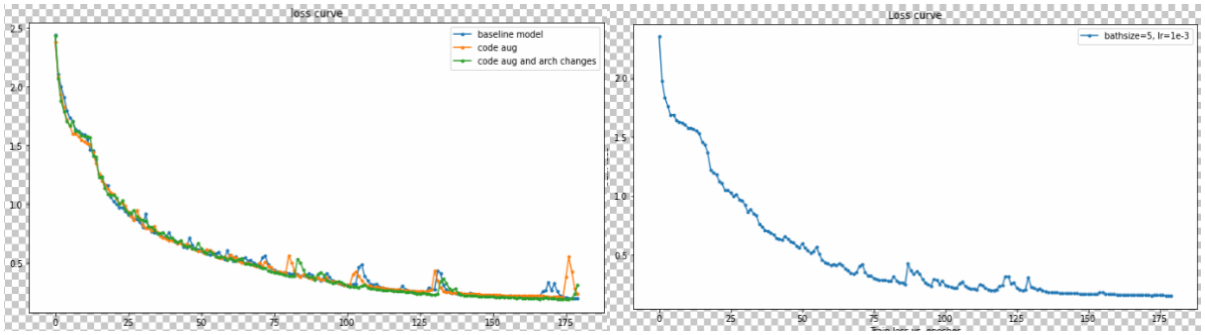
4.3 learning rate and batch size

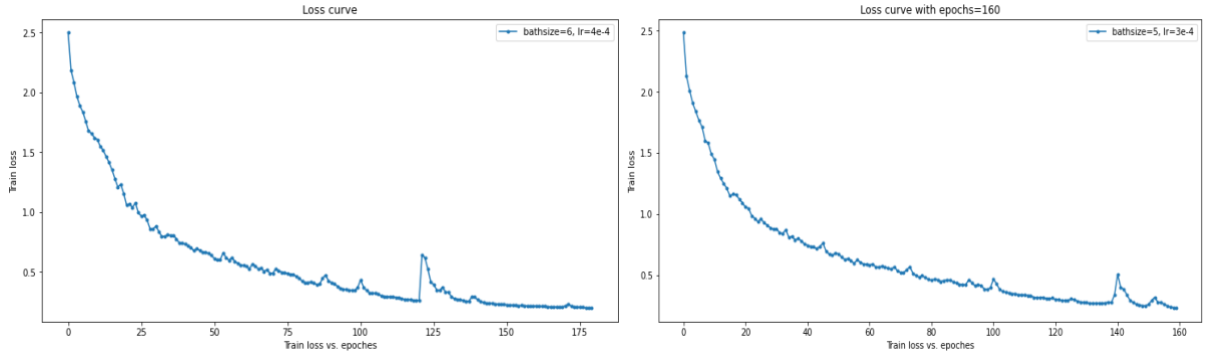
For dealing with learning rate and batch size, since we are not sure about the optimal values, we can only judge the values we need by increasing and decreasing them. So, in the first step, we used batch size = 5, learning rate = 1e-3, and the correct rate was only 28.4. We choose to increase the batch size and reduce the learning rate to near the initial value, so we use the batch size = 6, learning rate = 4e-4 set of data, the same accuracy rate is not ideal, only 29.45%. Finally selected batch size = 5, learning rate=3e-4 and get 160 epochs to train and when epoch bigger than 90 do the evaluation, The accuracy is not affected much, but saves a lot of time.

Loss curve for Part 4:

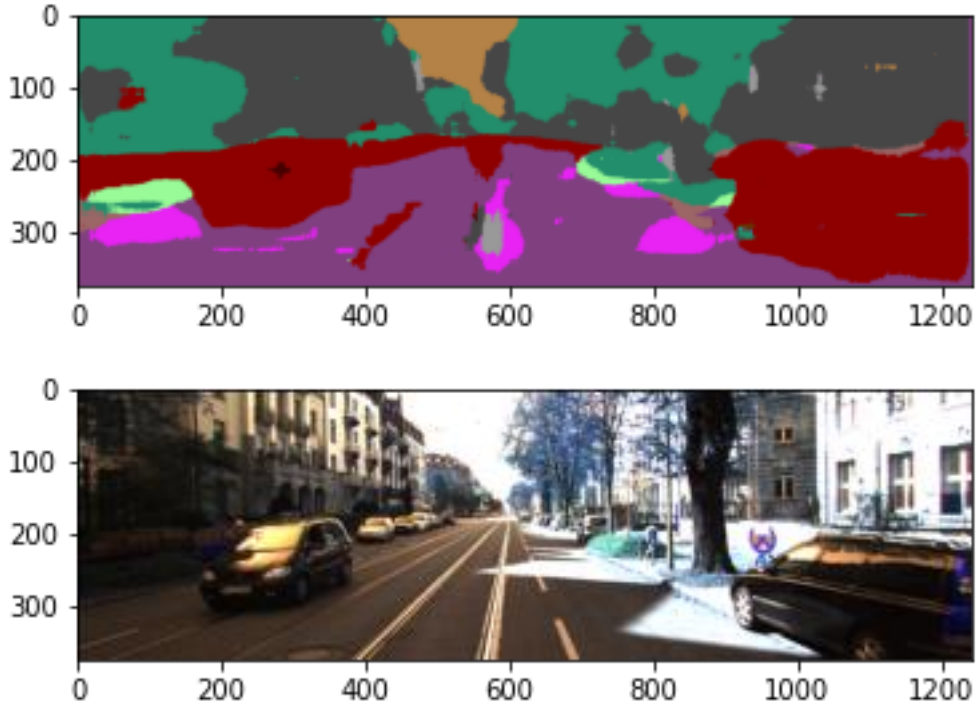








5. Visulization results: segmentation examples (color images)



6. Reflection

Semantic segmentation has always been a very important field in computer vision. Segmentation models are often used in areas such as autonomous driving and medical image diagnosis. Semantic segmentation assigns a category to each pixel in an image, but objects in the same category will not be distinguished. Missing small objects or minor object parts, and mislabeling minor parts of large objects as the wrong class occurs during programming. It should be our underutilized visual context, or it may be a defect of the model itself, and the model features need to be adjusted to retain more details.

In the realization of learning rate and batch size, after we changed the batch size to 6, the accuracy rate is slightly lower, because the batch size is too large, the optimization of deep learning (cannot reduce training loss) and generalization (generalization gap is very large) will cause problems.

The second batch size = 5, learning rate = 1e-3, it is more obvious here, it can be written too large, resulting in a significant decrease in the accuracy rate) Then I found that the batch size is 6 or 5 or 3 will reduce the accuracy rate, so the next training I still use batch size=4 (increase batch size first, because of the benefits of increasing batch size.