Machine Learning for Statistical NLP: advanced

Chinese character detection

Berenice Le Glouanec

University of Gothenburg

December 31, 2021

1 Introduction

This assignment has been made with the aim to detect Chinese character in images.

2 Data

It is a jupyter notebook that anyone can run without any requirements excepted having access to a dataset of images with bounding boxes. Here the dataset has been provided by the University in the MLTGPU server at `/scratch/lt2326-h21/a1/images/` location.

There is 845 images of size 2048 * 2048. The dataset has been divided into three parts : 507 images in the training set and 169 images in each testing and validation sets.

3 Process

First, we extract the bounding boxes of each images in each data set to get our ground values. Then we create a grid with the size of an image from the dataset (2048 * 2048). We process each dataset in our get truth function to transform each polygon points into a numpy array of 0 and 1. When a pixel corresponds to a Chinese character, its position in the numpy array is given 1, otherwise 0. Then each image (that we resize to 200 * 200 for less memory consumption) obtain a numpy array with the representation of its bounding boxes.

Due to a very time consuming process when getting these results, I decided to process my values in parallel, a function that I heard about thanks to a classmate.

4 Models

With Amelie's suggestion in Discord, I decided to implement one of the most known convolutional neural network which is Le Net model. I have been helped by various websites to understand which values should I modify and how. Instead of an average pooling I decided to use only max pooling because I already knew this layer. In this model there is three set of layers (composed by a Conv2d, a ReLu and a MaxPool2d), then the result goes through Linear, ReLu, Linear, Sigmoid and Upsample functions. Here we use Sigmoid as output because it is more suitable for binary classification.

The second model is based on a modern convolutional neural network suggested by `d2l.ai` website. The CNN uses a batch normalization. The structure is based on the previous model, but there is no ReLu anymore but BatchNorm functions instead. Also, I added a Sigmoid layer for each set, and changed the max pooling to an average pooling for each set as well.

5 Training and testing

In the training part, I used the Binary Cross Entropy loss function because it is more common to use this function for binary classification than the Cross Entropy Loss, even if it is doable with this last one. There is only 5 epochs because there is no significant differences between the second epoch and the following ones. In the testing part, I used the Mean Squared Error loss function as suggested by Asad in assignment's task.

There is a show model function where it is possible to chose a model and a dataset, it will then print a random image from the dataset, its ground values and its predictions.

6 Results

The mean squared error is slighly the same for each model and dataset (around 0.25), also the total loss is more or less the same for each model (around 10.9). I guess my models were not so

different from one to another. However, the total loss in each epoch in the training part seems pretty good for each of them, even if it is surprising, I wasn't expecting these results and I am not sure I can explain why it seems so good, maybe I didn't implement the loss correctly.

7 Conclusion

From my understanding, the first model (Le Net) seems to be more active in the predictions, it moves a lot everywhere in the image compared to the second model (Batch normalization) where it is more focused in the middle of the image.

References

https://stackoverflow.
com/questions/42220458/
what-does-the-delayed-function-do-when-used-with-joblib-in-python
https://www.tutorialdocs.com/
tutorial/joblib/examples.html
https://towardsdatascience.com/
build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f
https://d2l.ai/chapter_computer-vision/
object-detection-dataset.
html https://thegradient.pub/
semantic-segmentation/ https:
//www.pyimagesearch.com/2021/07/19/
pytorch-training-your-first-convolutional-neural-network-cnn/
https://d2l.ai/chapter_
convolutional-modern/alexnet.
html https://d2l.ai/chapter_
convolutional-neural-networks/lenet.
html https://medium.com/dejunhuang/
learning-day-57-practical-5-loss-function-crossentropyloss-vs-bceloss-in-pytorch-softmax-vs-bd866
https://d2l.ai/chapter_
convolutional-modern/batch-norm.
html https://www.guru99.com/
convnet-tensorflow-image-classification.
html https://towardsdatascience.com/
using-tensorflow-object-detection-to-do-pixel-wise-classification-702bf2605182
https://scikit-learn.org/stable/
modules/classes.html#module-sklearn.
metrics