Judit Casademont Moner
LT2316

Assignment 1: Chinese character "detection"

Report

Introduction

The objective of this assignment is to write two models with different architectures to be used to detect Chinese characters present in images. To do so, the CTW Dataset is being used. Said dataset contains 32,285 images and their respective json files with the bounding boxes information (which contain the coordinates of the corners of the bounding boxes that signal where Chinese characters can be found, among other details about the images), but for this work a much smaller dataset of 845 will be used. The two models that are explored in this work are the convolutional neural network LeNet, by Yann LeCun et al. (1998) and VGG16, another convolutional neural network architecture proposed by K. Simonyan and A. Zisserman (2014).

The assignment is written in the form of a Jupyter Notebook. Each cell is run one after the other in the order of appearance.

If someone were to apply the two models in the notebook, LeNet and VGG-16, on different datasets, they would need to, 1. Change the paths to the images accordingly, and 2. Check how the information about the bounding boxes is present in their dataset and adjust the appropriate function to be able to extract the needed information.

Data preparation

The available data, found in a folder containing the images of size 2048 x 2048 pixels and a file called train.json, had to be pre-processed before it could be handled by the models. To do so, some functions had to be made.

The pre-processing part is formed by a series of functions that 1. Obtain the usable images from their path, 2. Collect the information about the bounding boxes and store them in a dictionary, 3. Convert the images into Torch tensors, 4. Divide them into train (80%), test (10%) and validation (10%) sets, 5. Create Torch tensors representing the "gold truth" for the position of the polygons in the image, and 6. Process the data in parallel (a function suggested by a classmate) in order to reduce a processing time of over half an hour to about 10 minutes.

Additionally, a few choices were made on how to process the data, regarding its size and what type of information to be kept. Firstly, it was decided that, while collecting the information about the bounding boxes, all non-Chinese characters (is_chinese: False) would be ignored, since they would not be used in the models. Secondly, the size of the images (2048 x 2048) was drastically reduced to 300 x 300 for speed and memory purposes. If the images were processed while on their full size, they would raise a 'CUDA out of memory' error, hence the need for resizing.

The models

Both models of the assignment are convolutional neural networks with different architectures. CNNs are commonly used to process and analyse visual input.

Model number one is LeNet, first introduced by Y. LeCunn et al. in 1998, and it was originally used to recognize hand-written input. I decided to start by building a LeNet mainly because of its (apparent) simplicity. The architecture of my LeNet consists of the following layers: Convolution, ReLU, Max pooling, Convolution, ReLU, Maxpooling, fully connected Linear layer, ReLU, Linear and a Sigmoid function. I chose to keep a kernel size of 5 by 5 throughout the whole network since, if I chose higher values such as 30, by the latest layers after Maxpooling, the size of the input would become too small. I also kept my in and out channels at the initial three for the three colour values. The purpose of the final Sigmoid function is for the model to output values between 0 and 1.

Model number two is VGG16, designed by K. Simonyan and A. Zisserman in 2014, is a more modern convolutional neural network than LeNet. As its name suggests, it is formed by 16 layers. In the model for this assignment, I have five blocks of layers and a classifier. Each of the five blocks is formed by the following layers: Convolution, Batch normalization, ReLU, Convolution, Batch normalization, ReLU and Maxpooling. After these, there is a block called 'classifier' that uses the Sequential Pytorch module and which is formed by the following layers: Linear, ReLU, Dropout, Linear, ReLU, Dropout and Linear. The whole model, like with the previous one, ends with a Sigmoid function to output values between 0 and 1.

Since all training images were downsized to 300 x 300 pixels during pre-processing to train the model efficiently, after outputting them it was necessary to upsize them back to 2048 x 2048. This was done by using nn.Upsample and by reshaping and permuting.

For the first model, Cross Entropy loss was used, whereas for the second one, I chose to make use of Binary Cross Entropy loss. The optimizer was Adam for both models.

Testing and evaluation

As it was expected from the little amount of available data (845 images), the results of the tests showed that the performance of the models is not satisfactory.

Model number one produced a loss of 0.318122 in its final epoch, and its mean squared error was of 21.504563, which is too high. If we examine the effects of applying the model to images individually, we can see that, in most of the cases, the model believes that the whole superior half of all images is filled with Chinese characters. This might be due to the fact that, in many images, the bounding boxes signalling Chinese characters are found in this area, and my model simply makes a generalisation and assumes that the whole area contains the objects we are looking for.

Model number two's loss in its final epoch was of 0.497387, which is higher than that of the LeNet model. However, its mean squared error is notably lower, 1.643059. This would make us believe that this second model is much more accurate than the first one. Nevertheless, if we take a look at the effects of applying the model to an individual image, we realize that my VGG16 outputs no polygons whatsoever, at least in the images that were checked. Since the bounding boxes are small relative to the size of the picture (so, Chinese characters do not take a lot of image space), it is logical that, by outputting a total of zero bounding boxes for each image, the model will have a low mean squared error although it is not doing its job.

Conclusion

My conclusion is that it is likely that my models offer a bad performance due to the lack of data and to the fact that the images had to be downsized for training in order to get over the memory errors from CUDA. There is also the possibility that running many tests, changing and tuning the parameters and adding or modifying some of the layers the models' performance could be improved. The models presented in this assignment are merely experimental and they could be improved in future work.

References

For LeNet:

- http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf
- https://www.pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/

For VGG16:

- https://arxiv.org/pdf/1409.1556.pdf
- https://neurohive.io/en/popular-networks/vgg16/