Project 3: OCR

Due: 26 April 2022

v20220421-1000

One of the tasks often used to demonstrate and describe neural networks is that of visual recognition: how to take an image and process it to identify what it contains? In this project, you'll build networks to perform optical character recognition on digits in images.

The MNIST corpus

In the shared directory (`/home/shared/389/`) I have put a copy of the MNIST corpus of handwritten digits. This data contains one digit per 28x28 greyscale image, and the images have had some preprocessing done to resize and reposition them into a standard layout. The directory has a link to further information about the data, including its source and some of the high-performance OCR models that have been published to classify it, but the most important part of that link is the one that tells you the layout of the data files.

I will not fully recap it here, but note that each data file contains thousands of images, in binary format. You will need to do a little work to read in the data, before you can really start applying any neural networks.

The other thing to note is that the data is already split for us into training and testing. We will follow the standard split: use the 60k training images to build your model, and then the 10k test images to evaluate its performance.

TensorFlow

Nobody is really programming their own neural networks from scratch these days, and while there is learning value in doing so[1] I'd rather have you use one of the tools in common use out there: TensorFlow.

The vast bulk of documentation and use of TF is in Python, so we'll be using that. Remember that on our systems `python` is still installed as the older Python 2, so you'll need to consistently use the `python3` executable.

[1]and doing so is how I've run this project in the past, and is the backup plan for the last 2/3 of this one

My path through this is still tentative. Give me lots of live feedback on how it's going.

Checkpoint

The checkpoint is to build a neural network to process the MNIST handwritten-digit corpus, a classic but simple classification problem. Much (most?) of what you type in will probably be from tutorials, which is fine.

This will be the main one:

- https://www.tensorflow.org/tutorials/keras/classification

Build, in Python, each of the following models, and evaluate how each one does on the test set:

- a one-layer neural network (no hidden layer, just the 28x28=784 input layer and the 10-possible-label output layer), to classify the MNIST digits;
- same but with a single hidden layer of 100 nodes;
- same but the hidden layer has 400 nodes.

Give the results in your readme as well has how to re-generate each one.

Then, generate a handwritten digit *not* in the MNIST set that you think is ambiguous between two interpretations (perhaps the ones we used in class, perhaps others). Include it in your directory as a .png or .jpg file (or .bmp), figure out how to read it into your program and use your trained model to make a prediction about that single image. (This piece is the place where the prep work most diverges from the direct contents of the tutorial... remember that you can look at other documentation too.)

Documentation is more than usually important here, even for the checkpoint—you must tell me how to run your stuff.

The checkpoint is due Tuesday, 12 April at 4pm.

The goal

As discussed in class, the *goal* is to be able to read in an image file that has multiple handwritten digits in it, and feed that through the mnist layer you built for the prep work to identify all the written digits in the source image.

Final version

When you hand in your work, make sure to *include notes and drafts*, whether that's smaller components that work, or slightly work, even if you haven't pieced them into a larger whole; or URLs of pages you've found helpful with notes of what you learned from them and how you think that thing would be used in a final product.

Then, because there's a lot of files floating around, make sure the readme tells me what it all is. The readme should definitely not be a brain-dump even if some of the other files are.

Rubric

Handing in

The checkpoint and the final version are due at 4pm on their respective due dates. Hand them in as `proj3` using the handin script.

Don't forget documentation! And appropriate output that will help me understand how you think you're meeting the rubric requirements.