Data Processing Final Project

Pytorch: Handwritten digits & letter classification

Matus Halak

For my final project, I wanted to finally get my hands dirty with machine learning and AI. I had learned about some of the theory behind neural networks in some previous courses and had watched quite some videos about AI and neural networks out of interest, but never tried to make a neural network on my own. Since we were supposed to learn a new skill and library, this seemed like a perfect opportunity to try to finally learn some of the ML basics which I could keep building on later.

After some googling, it seemed like Pytorch was the most widely used AI/ML framework, which is why I decided to learn some pytorch basics. With this in mind, I looked for beginner pytorch projects and I kept coming across the handwritten digit classification task using the MNIST database. Upon consultation with Amir, I decided to pursue this for my final project, with the addition of 1) making an interface for drawing my own handwritten digits to be classified, and 2) also classifying handwritten letters on top of just digits. My goal was to learn how to make and train a neural network that could recognize and classify my own drawings of digits or letters. My research question was whether I could make and train a model that would perform this task well (>80 % accuracy).

I was fortunate enough that both MNIST (digits dataset) and EMNIST (digits + letters dataset) datasets are built into pytorch and can be downloaded and accessed through pytorch. Therefore, obtaining and working with the training and validation data was very straightforward. After some initial research, I realized that while pytorch is quite user friendly, there was still a lot of new syntax and steps I had to learn. Thus, I started the whole project by making the interface for drawings, which seemed like a simpler task. I used pygame and with the help of a few video tutorials, I made a nice little script that did exactly what I wanted, so I was quite happy about that (I kept tweaking the draw.py script a little bit, as I progressed through the project, but the bulk of the work was done in the beginning).

Once I started working with pytorch, I grasped the general workflow and the main steps in defining, training and testing a model quite quickly. What I kept struggling with throughout the whole project (and still have little insight on) is how to decide what model architecture and training parameters to use for a given task. Answers I have found online claim it’s mostly trial and error but there are effectively infinitely many possibilities and with the limited horsepower of my computer, I often felt really lost trying to “experiment” with the different parameters trying to optimize performance. Nevertheless, I managed to achieve good performance on both the MNIST and EMNIST training and validation datasets with fully connected (and for EMNIST also convolutional) neural networks.

My biggest nemesis in this project turned out to be classification of my own hand-drawn images. I struggled endlessly to figure out why the models performed so well on unseen validation data (>90%) but would perform awfully on my hand-drawn images. Some preprocessing steps like inverting the colors of my images, resizing them and turning them to greyscale helped somewhat, but not nearly enough. At some point I realized that I would have to do a lot more preprocessing to make my images look more like the training and validation images and improve the performance on my images (crop, change line thickness, center only the drawn character, add padding, etc.). I knew this was probably the right way to go but would take a lot more time, and I will try to do this in my spare time over the summer to see if it can address some of the issues I encountered. When working with EMNIST, at some point I achieved >90% accuracy on my hand-drawn letters, but once I started adding more examples of other letters and especially once I included my own hand-drawn numbers the accuracy went down. Based on this I reached the conclusion that probably there were some letters which were really distinct from everything else and easy for the model to learn, and some letters which looked very similar and were hard to distinguish (eg. round letters like u,v,w etc.) and that in general, digit characters look quite similar and might be harder to distinguish. Still I believe that with more preprocessing, this issue would most likely go away, but currently, my models can very reliably learn some digits and some letters and classify them correctly (> 90%) while almost always misclassifying many other digits and letters.

While I didn’t manage to get my models to work as well as I had hoped, I got good results on unseen validation data for both MNIST and EMNIST and I think I learned quite a bit about what I could do next to improve my work and get the results I wanted. Overall I think this project has taught me a lot and I am really excited to keep learning more about pytorch and starting other personal ML projects.