CS B657: Computer Vision, Spring 2016

Final Project Final Report

Scott McCaulay

Introduction:

This project is an evaluation of a neural network as a method for performing Optical Music Recognition. The goal is to try to use a neural network alone to parse images of musical scores, with as little help as possible from other computer vision techniques.

A big part of the challenge in this field is the heterogeneity of the data. There is an interest in retrieving data from scanned musical scores from different historical eras, many originally hand written, in various conditions, sizes and using inconsistent notations. This project makes no attempt to address those challenges. Since all the data utilized here will be from the same source, it will be as though the image pre-processing stage is already completed, and done flawlessly. Given this sanitized and consistent data, it may be possible for a machine learning algorithm with no understanding of the domain to produce results at least better than random guessing. That is what the project intends to test.

Background and Related Work:

Background reading has primarily been on the history of the field and the state of the art. There are several good sources for an overview of the field and surveys of recent work. Typical methods of optical music score recognition use a combination of routines for individual tasks such as detection of staff lines, recognition of musical symbols, etc. There has been some work using neural networks as part of a solution, in combination with the aforementioned methods. Neural networks do not seem as widely used today for musical scores as they are for character recognition or speech. One goal of this work is to explore the possibilities and challenges in applying neural networks to this application, specifically where they can help and where they may not be as helpful.

There have been some efforts to create common test beds for this type of research, but given the unique goals of this project, rather than pursue an existing set of test data, I decided to generate my own uniquely tailored data. As mentioned before, it's understood that removes a lot of the difficulty from the problem, but this embarrassingly consistent data meets the purpose of this project.

Data and Methods:

Data:

Test data for this project has been generated programmatically. A set of 200 MIDI files has been generated, along with an "answer key" for each file in CSV format. Score images were generated from the MIDI files using the freely available MIDI Sheet Music application. The images in PNG format are input to the Neural Network application, with the CSV file which contains location, pitch and duration for each note for training and testing.

Limitations were placed on the data to simplify recognition. All notes are in the key of C, so there are no sharp and flat icons to decipher. All files are in 4/4 time signature. Notes are within a range of E2 to E5, only three note durations are used. Each time step can have one or more notes. Given temporal placement, pitch and duration, there are 1,008 unique note events which can be present in a score.

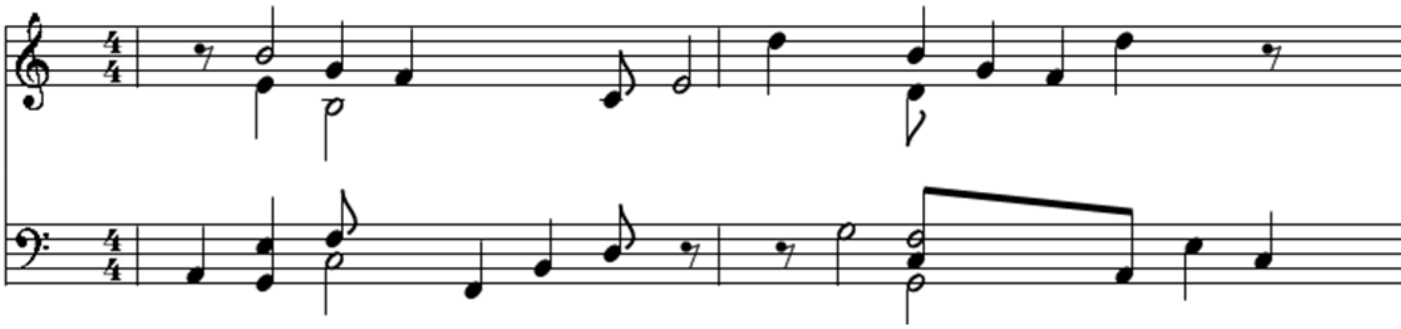

Image 1: An example Score file, a corresponding key is produced as a CSV file

Training the neural network was time consuming, so as a compromise the training set was limited to 50 files. More files would have produced better results. Another 10 files are distributed to use for testing the trained network.

Pre-Processing:

Several steps were taken to reduce wasted effort and improve performance (although more could have been done). The input files are 840 x 312 pixels, making a possible 262,080 nodes in the input layer of the neural network. A preliminary process filters this data to standardize and minimize input to the neural network. First the staff positions are aligned. Second, the value of each pixel across the entire corpus is calculated. Locations whose pixel values never change are eliminated. Also, pixels whose values are split nearly evenly are eliminated. The remaining are the most unique values. This greatly reduces the number of input nodes reducing memory and calculation requirements for our network. Figure 2 shows the results of this process, common values are eliminated, darker pixels in this image indicate more unique occurrences, and lighter grays show areas that may be common to multiple events.

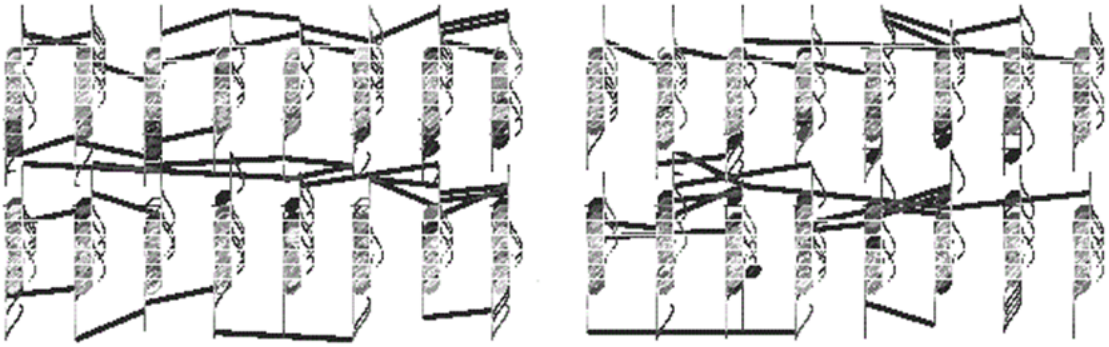

Image 2: Identifying most significant pixel locations in the corpus

The Neural Network:

The network is a simple, traditional feed-forward neural network. It has approximately 32K nodes and over 30 million edges, which presents a moderate challenge for calculation, but is trivial in comparison to networks developed at Google or Stanford with billions or 10's of billions of edges. Without the pre-processing, the number of edges would have been over 250 million.

The input layer of the neural network is fed from the pixels of the score image file, with their values set to 0 for white pixels and 1 for black pixels. The output layer of the neural network is a multidimensional grid representing all the unique note events. As another compromise to resources and performance, no hidden layer was used, weights connected every input node to every output node. The reduction in effectiveness from this decision is unknown.

Given the temporal compartmentalization of both the input and output layers, it was possible to split the network into isolated frames so that inputs from within a frame only feed outputs in the same frame. The divisions of the frames are calculated as the midpoints between the denser clusters of pixels seen in Image 2. Along with the preprocessing of the input data, this reduces the calculation required for the network. A consideration was made to train a single set of weights which could interpret each of these frames. This was rejected due to some uniqueness in the way notes were displayed by frame (bodies, and especially tails of notes could be drawn in either direction around the stem). With more time this may have worked and made a much more compact network.

The goal of the network is to correctly calculate the cells of this grid to match exactly the notes present in the score. This is done by iteratively adjusting the weights of the network edges connecting input pixels to output cells, until the calculation of input

values, adjusted by the excitatory and inhibitory weights of the connecting edges, converge on the correct values in the output layer.

Training was done using a genetic algorithm, with the weights of the network as the genome. Back propagation would have been faster, but the genetic algorithm allowed for some experimentation with the topology of the network (which did not make it into the final project). This method did provide an easy path to parallelization, since each member of a population could be calculated independently.

The Software:

The project contains 5 applications, all written in C# and developed in Microsoft Visual Studio. This allowed for rapid prototyping and testing, and the resultant programs were able to run on large machines at IU (Big Red and Karst) using Mono, which supports cross-platform execution of .NET programs. Two of the applications produce the MIDI input files and test the neural network respectively, the other three combine to train the neural network. Details of the applications are in the deployment section of the report.

Results:

For each generation of evolution, each candidate network was tested against the 50 training sets by applying the pixel values to the input nodes, and applying the weights to calculate the output nodes. The resultant scores in the output nodes were compared to the ground truth files to calculate a fitness score for the candidate.

Using these scores, the network was trained to pursue three goals simultaneously. One goal was simply to segment the absence or presence of note events on either side of an arbitrary score threshold. A second goal was to maximize the count of true positives. This was necessary because negatives outnumber positives by almost 50 to 1, so a good score could be achieved by setting all weights to zero. A third goal as to maximize the gap between the average calculated score of the negative results from the average calculated score of the positive results.

A testing program used the weights calculated from training against a new set of files not part of the training set. Using the pixels from a new file with the trained weights, the output grid is calculated. These calculations are sorted and the highest value up to the number of notes in the test file are returned as guesses. Each guess that represents a note event present in the test file counts as a hit, others are misses.

Results against test files were not spectacular, but were overall much better than random guessing could produce. With 1,008 possible combination of position, note and duration, each guess has less than a 0.1% chance of accuracy. Image 3 shows results of one of the better runs which had 14% of its guesses correct.

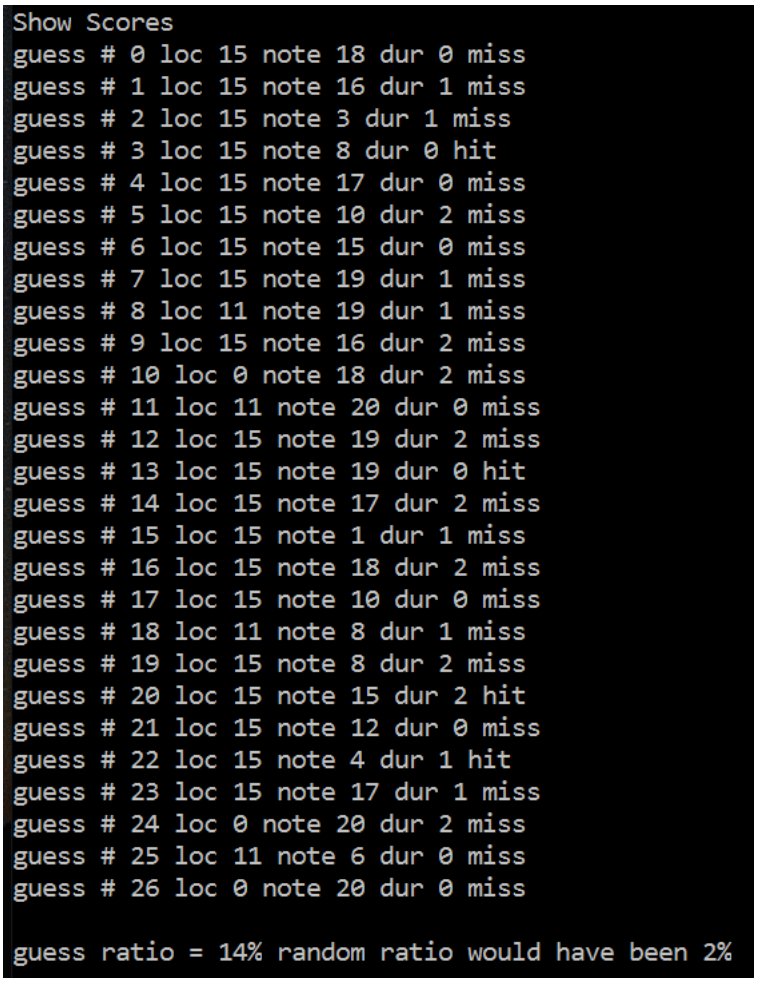

Image 3: Example output from the test program

Analyzing the training data shows that some note locations seemed to be easier to differentiate. The difference in average weights pointing to “on” versus “off” notes in the test data was very high at the first and last note positions and much lower in the middle of the score (12th position is also higher). It’s not known if this is because the data is clearer at the edges, with less interference, or if it’s simply a logic error in the program.

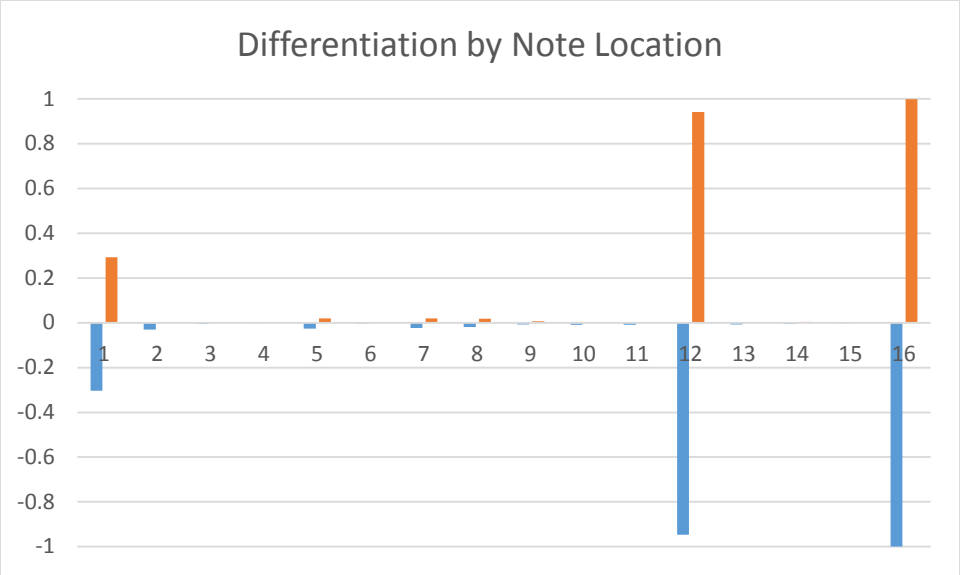

Image 4: Weight training is suspiciously position-specific

How well the training was able to separate positive from negative events by note pitch is also uneven, but more understandably so. The lowest pitch is the easiest to differentiate. It's also the only note whose body area is never occupied by another note's stem or tail. Some of these oddities can probably be attributed to the too small training set and overtraining. The differentiation by duration is not pictured, but predictably the hardest to recognize was the whole note, whose center is hollow.

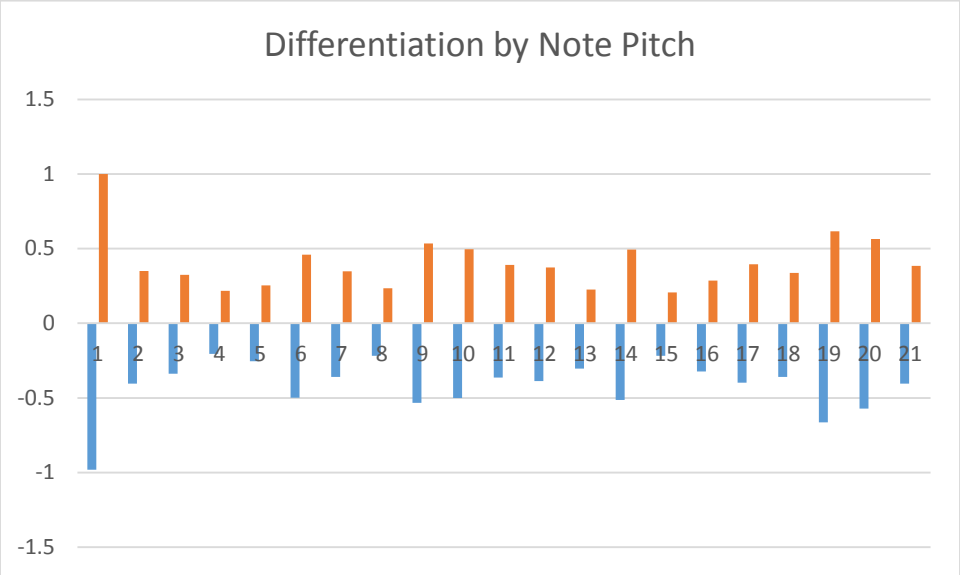

Image 5: Best success is with the lowest note on the staff

Conclusions:

Given the heavily sanitized data, there was an expectation that performance would be much better, and it may have been with different network design choices and training targets. Observing the input data and evaluating the ambiguity of the pixels does highlight the difficulty of the problem, without regard to the methods used to address it.

Preliminary results indicate that a neural network could be a helpful addition to an optical music recognition toolbox, in combination with other tools. Given a uniform set of scores and adequate training data, a neural network could potentially supplement other tools to locate and identify symbols in a score. This work does highlight the ambiguity present even in machine generated audio scores with severely limited types of data. Given complex, inconsistently presented data, considerable preliminary work would be required to get adequate results from a neural network. This network was not very sophisticated and yet did contribute toward solving the problem. A more refined network, supplemented with established methods could perform much better.

Deployment for Testing:

The GitHub repository contains a source directory, and a zip file NNVision.zip.

If it is necessary to build the applications as part of grading, this can be done with Microsoft Visual Studio. Open the project file ./source/NNVision/NNVision.sln and select Build Solution in Visual Studio. Pre-compiled executables are included in the zip file, so building is not necessary for testing.

The source code for the five applications are in subdirectories under ./source/NNVision. A brief descript follows:

./source/NNVision/genMIDI/gen_midi.cs

This generate MIDI files and CVS answer keys

./source/NNVision/testNN/testNN.cs

This selects one test image and uses the NN to guess its notes

./source/NNVision/EvolveRT/launchNN.cs

Kicks off training, takes seconds and threads as parameters

./source/NNVision/RTloop/loopNN.cs

Evolves the weights using a genetic algorithm

./source/NNVision/fixedNN/fixedNN.cs

Runs the weights against the test images and calculates fitness score

Deployment:

To test the programs on Karst (or any linux machine with Mono), unzip the archive NNVision.zip. This contains all the executables, subdirectories and input data required to run. For a full training these should be submitted as a batch job but for quick testing they can run interactively on a login node. The module load commands only need to be run once for a session. The Mono version is critical on Karst.

Prepare the Environment: (run once)

On Karst:

- module load mono/3.2.3

For example, to run for one minute on one thread, mono launchNN.exe 60 1

On Big Red:

- module swap PrgEnv-cray PrgEnv-gnu
- module load mono
- module load ccmnchNN.exe *seconds threads*

Training (a full training runs for hours, but it can be tested for a few minutes)

(Even if a short time is selected, it will take a few minutes to complete)

On Karst:

- mono launchNN.exe *seconds threads*

For example, to run for one minute on one thread, mono launchNN.exe 60 1

On Big Red:

- ccmrun mono launchNN.exe *seconds threads*

Test:

On Karst:

- mono testNN.exe *fileno 0-9*

For example, to test with image 7, mono testNN.exe 7

On Big Red:

- ccmrun mono testNN.exe *fileno 0-9*

Generate Test Files:

On Karst:

- mono gen_midi.exe

On Big Red:

- ccmrun mono gen_midi.exe

References:

Byrd, Donald, and Jakob Grue Simonsen. "Towards a standard testbed for optical music recognition: Definitions, metrics, and page images." *Journal of New Music Research* 44.3 (2015): 169-195.

Fornés, Alicia, et al. "CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal." *International Journal on Document Analysis and Recognition (IJDAR)* 15.3 (2012): 243-251.

Raphael, Christopher, and Rong Jin. "Optical music recognition on the international music score library project." *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013.

Rebelo, Ana, et al. "Optical music recognition: state-of-the-art and open issues." *International Journal of Multimedia Information Retrieval* 1.3 (2012): 173-190.

Vaidyanathan, M. (2007-2013). "Midi Sheet Music." 2.6. 2016, from https://sourceforge.net/projects/midisheetmusic/.

Wen, Cuihong, et al. "A new optical music recognition system based on combined neural network." *Pattern Recognition Letters* 58 (2015): 1-7.