CS 6700 Project Final Report

Professor Bart Selman

Matthew Luebbers (mbl228), Ryan Meredith (rcm269), & Kyler Ruvane (kgr32)

An Exploration of Multi-Purpose Neural Networks

Table of Contents:

Abstract:

In common practice, neural networks are trained to complete a singular specific task, optimizing its performance with respect to that task. For this project, we explored the concept of training a single neural network to complete multiple unrelated tasks, and comparing the performance to that of separate neural networks learning each task separately. This concept is known as a multi-purpose neural network, or MPNN. To test

the abilities of MPNNs to learn varying tasks, we picked two tasks that can be quantitatively verified: handwritten digit classification and binary XOR evaluation. We also picked a third task that is more qualitative and creative to test neural network performance in a domain that is traditionally viewed as human-only: music composition. We ran a number of experiments, gathering data both from the single-task baseline nets and a number of MPNNs learning multiple tasks. While we were able to achieve similar performance to the baselines on the two quantitative tasks using MPNNs, the time taken to train the combined network was significantly greater. The experiments demonstrated, however, that compact neural networks can be made to work on multiple unrelated tasks, a fact that may provide insight into how human brains complete tasks.

Design & Goals:

This project aims to evaluate the complexity and effectiveness of learning multiple independent functions in a single neural network. Although the process that a human brain goes through to solve problems is very similar to the process a neural network uses to solve problem, there is still one very big difference: the human brain can learn hundreds of thousands of problems and does not need to specialize to the extent that a neural network does to achieve good results. The goal of this project was to explore how well a single neural network can imitate the human brain's capabilities.

Specifically we wanted to measure the effectiveness of learning multiple functions with a single neural network as well as learning a task that is generally thought to be exclusively human in nature. The task we settled on to achieve this last goal is music composition. It is a common view that music and art are two categories that will always be exclusively created by humans and that artificial intelligence will always seem artificial and inferior in these creative fields. The other two functions we chose to learn were handwritten digit classification and the XOR function for two 16 bit numbers. These functions were chosen because they allow us to quantitatively measure how successfully the neural network learned. Another bonus is that the handwritten digit classification problem is widely benchmarked and the XOR function is learnable to 100% accuracy.

Implementation:

This section will be broken up into two major sections. In the first section we will describe the neural network framework and guidelines that we used throughout all of the neural networks that were trained. In the next section we will discuss the implementation of the five different neural networks: three individual neural networks that solved music composition, handwritten digit classification, and the XOR function respectively as well as the MPNNs - these being both a neural network that combined handwritten digit classification and the XOR function and finally a neural network that combined all three tasks.

Neural Network Framework:

The entire project was written in Python and made use of a Python neural network framework called Keras. Keras is an API that is built on top of TensorFlow. It allowed us to quickly define and customize our neural networks and train them on our data. The table at the end the implementation section gives the specific attributes of each neural network that we created. To combine multiple functions into a single neural network we simply concatenated inputs and outputs together to form a larger zero-padding input array and output array. For example to send digit classification (784 inputs) and the xor function (32 inputs) through a neural network you would send a 816 input array that is 0 padded for the function that is not being represented in the input.

Music Composition (**A, E**):

We decided to write a program to create novel music modeled off of classical piano pieces. To go about this, we used a neural network format where we would predict the next note based on a number of previous notes, which we later set to 20. The process we used to create songs once we had trained the model was to select a seed input of

20 notes from a random song in our training set, predict the next note, and then append that to the current list of notes. We then used that note and the previous 19 to predict the following note. We repeated this process until the song was of a predetermined length. To train the neural network, we assembled a dataset of 96 classical piano MIDI files, all in duple meter (time signatures of 4/4, 2/4, and 2/2) to add some regularity to the music. An important Python library that we used to parse the music and convert the output back to a MIDI file was Music21. Music21 provides functions to convert MIDI files to streams of sounds consisting of notes, chords, and rests. From here we were able to iterate over the stream, converting each sound into a representative vector. We represented sounds using an array that was 89 elements long. This corresponds to the 88 keys on a keyboard and one spot for rests. Every element was zero except for keys that are being played. Chords were represented with multiple values set to one rather than just one note. For a rest, the final element in the array was the only value that contained a one.

XOR (**B, D, E**):

The XOR function was learned by producing 60,000 examples of random 16 bit integer inputs and their corresponding XOR value. This was done with a simple Python script that chooses two random numbers, XOR's them together, and then writes the input and output to a file in a binary array format that can be parsed by a neural network. The network used to learn XOR is described in column B, D, and E of the table below.

Digit Classification (**C, D, E**):

Digit classification was learned by using the MNIST dataset which contains 60,000 labeled examples of handwritten images. These images are 28x28 grayscale images. To send them through a neural network, we flattened the array to 784x1 and scaled each number between 0 and 1. The network used to learn digit classification is described in column C, D, and E of the table below.

Neural Network Attributes:

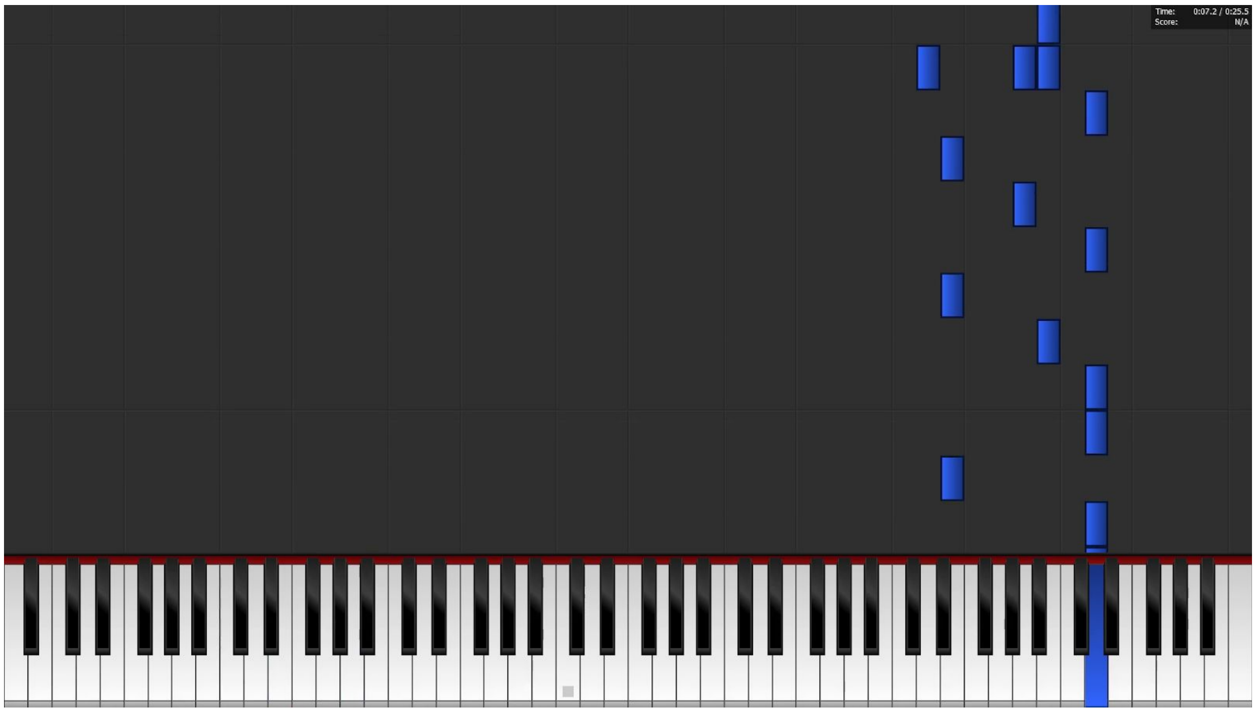
The following table describes the various neural nets used for experimentation in this project.

Name	**A**	**B**	**C**	**D**	**E**
Task(s)	Music Composition	XOR	Digit Classif.	XOR + Digit Classif.	XOR + Digit Classif. + Music Composition
Training Samples	100,000	60,000	60,000	60,000 (~30,000 each)	60,000 (~20,000 each)
Test Samples	N/A	10,000	10,000	10,000 (~5,000 each)	10,000 (~5,000 each)
Epochs	25	100	15	200	100
Batch Size	64	32	128	128	128
Neural Net Shape	In - 1780 H1 - 500 H2 - 500 H3 - 200 H4 - 100 Out - 89	In - 32 H1 - 32 H2 - 200 H3 - 100 Out - 16	In - 784 H1 - 64 H2 - 64 H3 - 64 Out - 10	In - 816 H1 - 32 H2 - 200 H3 - 100 Out - 26	In - 2596 H1 - 32 H2 - 200 H3 - 100 Out - 115
Dropout	20%	5%	5%	5%	5%
Activation Functions	ReLU for hidden, softmax for output	ReLU for hidden, tanh for output	ReLU for hidden, softmax for output	ReLU for hidden, tanh for output	ReLU for hidden, tanh for output
Loss Function	Categorical Cross-Entropy	Mean Squared Error	Categorical Cross-Entropy	Mean Squared Error	Mean Squared Error
Optimizer	RMSProp	Adam	Adam	Adam	Adam
Metric	“Categorical Accuracy”	“Accuracy”	“Accuracy”	“Accuracy”	“Accuracy”

Results & Evaluation:

Music Composition (**A**):

This neural network went through a couple iterations. The first iteration was trained on a a set of 100,000 examples randomly chosen from 96 different classical piano songs. Although the songs were intentionally varied, all songs had duple meter (common time and its variants) to ensure some measure of consistency. Below is a snapshot of one of the songs it produced.



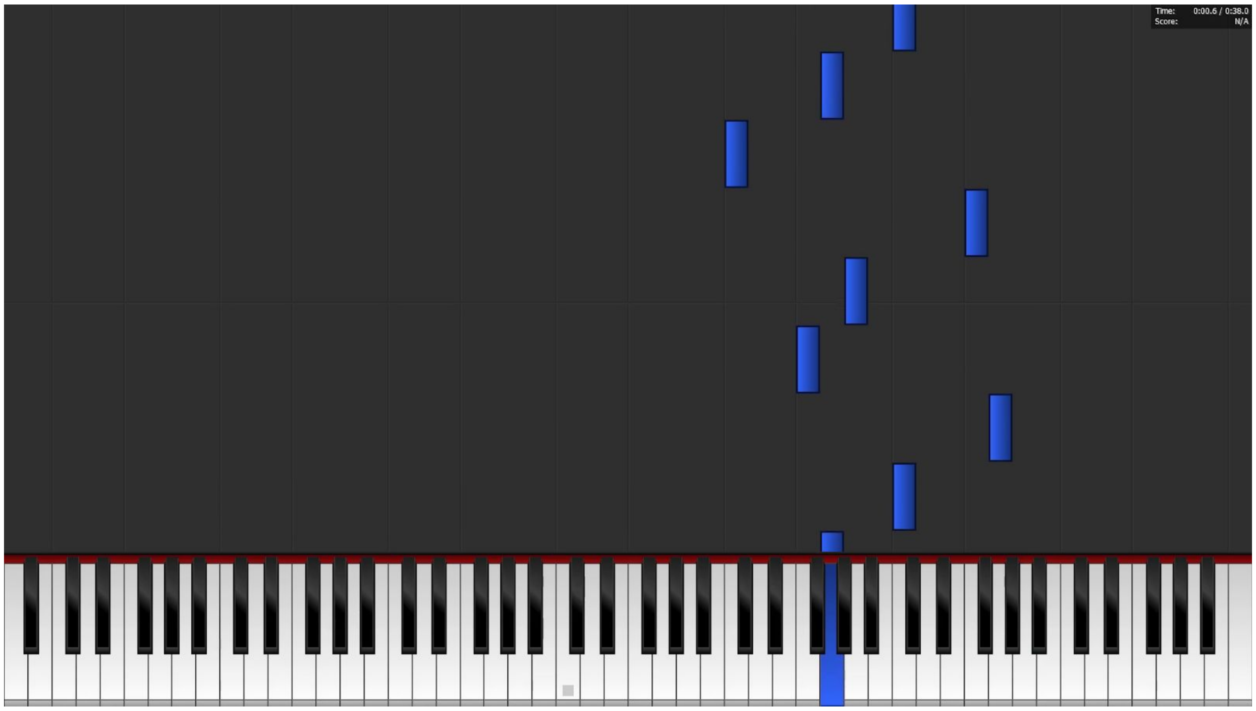
See video: https://youtu.be/Sct0uVnfteE

This neural network produced middle quality songs that were often hard to define the structure of song but regardless somewhat pleasing to listen to. An unfortunate problem was that each song contained repetition near end of song due to convergence of the output.

Another example from the same NN: https://youtu.be/HWPv5xkX0TA

Although we successfully produced music with the first iteration of this neural network, we were not completely satisfied with the very minimal structure of the songs produced.

The reason we believe the first iteration encountered this issue was the extremely varied nature our training data. We believe the songs were simply too different from one another in order to use to generalize a neural network to learn all of them. Our second iteration used a set of 20 famous songs from the video game series *Final Fantasy*. These songs, in addition to being much simpler, are much more uniform. Below is a snapshot of one of the songs it produced.



See video: https://youtu.be/P7TfZkAOQzE

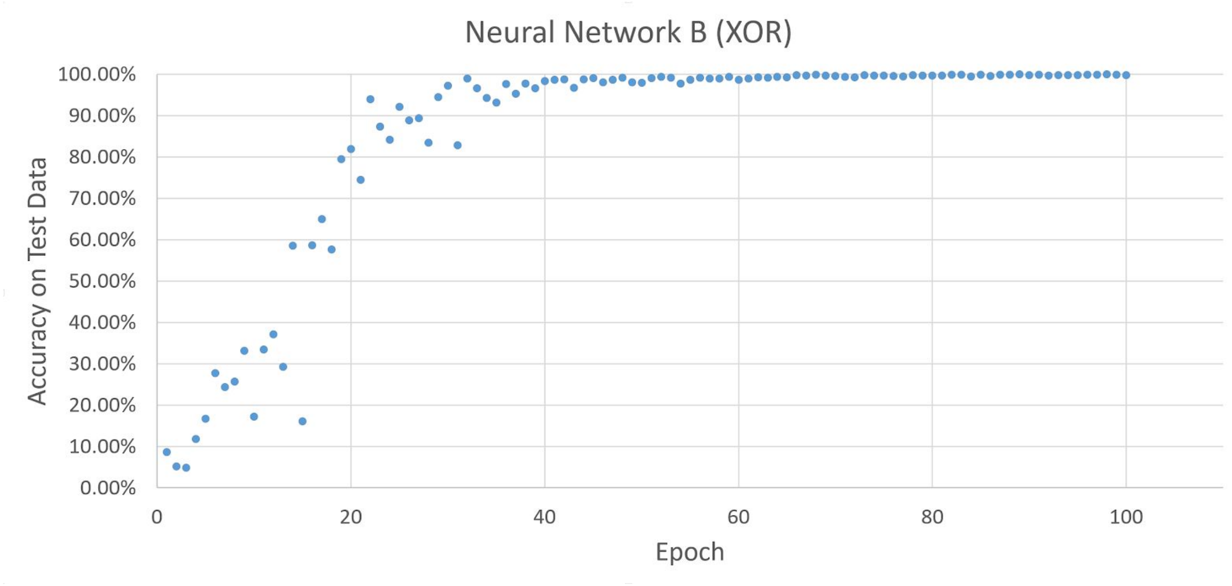
The songs produced by this version of the neural network were much more structured. In the above image you can see a down scale of G-C-G, F-A-F, C-G-C. This scale makes a lot of sense musically and is pleasing to listen to.
Another example from the same NN: https://youtu.be/pwApVz1pLpw

The last iteration of this neural network was a test to see if the neural network was being overfit to the training data. This network was only run for 5 epochs instead of 25 epochs. Overall the result was very similar to the network that ran for 25 epochs. This leads us to believe that we were not overfitting the data but also that the neural network effectively plateaued in the early epochs.

Example for NN with only 5 epochs: https://youtu.be/DpC_Pu_l2Ck

XOR (**B**):

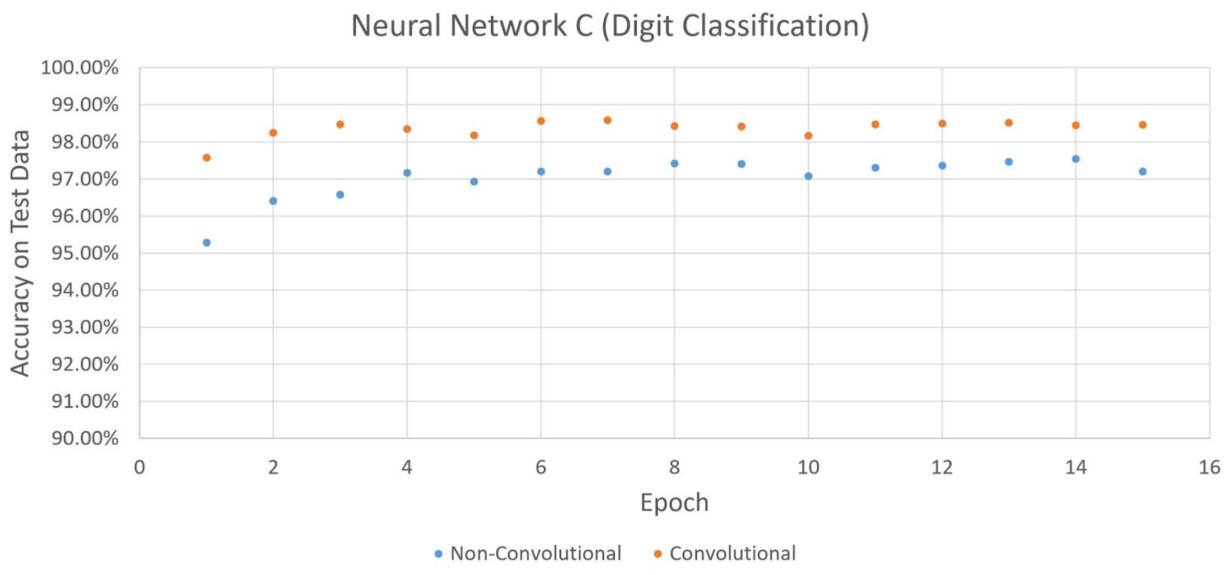
As described previously, this neural network was implemented to provide a baseline for the XOR function. Although the XOR function is often learned in neural networks, it is usually constrained to a single bit XOR. To make the function more interesting for a neural network to learn, we attempted to learn 16 bit XOR in a single neural network. This means that it had 32 input nodes and 16 output nodes instead of a single XOR bit with 2 input nodes and 1 output node.



This graph plots the number of epochs vs the accuracy on the test data. As you can see it approaches 100% around epoch 50 and levels off around epoch 75. The final accuracy percentage achieved was 99.87% after 16 minutes, 57 seconds of training. To achieve even better results we added a layer of post-processing after the neural network makes its prediction. Because the XOR function expects only 0 or 1 output we can force this constraint by setting all values less than 0.5 to 0 and all values greater than or equal to 0.5 to 1. This post processing resulted in 100.00% accuracy on the test data set.

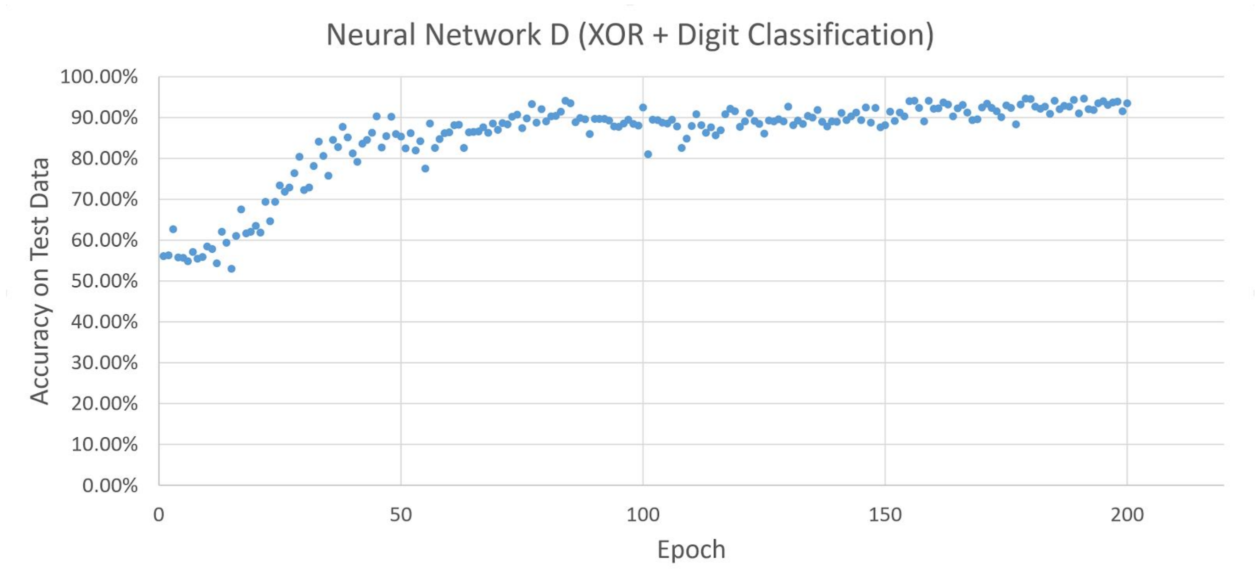
Digit Classification (**C**):

This neural net provides a baseline for performance on classifying digits from the MNIST handwritten digit database. It takes as its input a 28x28 grayscale image and returns a digit 0-9. We trained two similar neural nets on this data - one as described in the Neural Network Attributes table on pages 4-5 (Neural Network C), and another with a 3x3 convolutional layer added into the first hidden layer of the network.



This graph shows both the non-convolutional and convolutional nets' performances on the test dataset. Both started off with a high accuracy - over 95% for the non-convolutional and over 97% for the convolutional. Over 15 epochs' worth of training, the accuracies both improved marginally, ending on 97.20% for the non-convolutional net after 23 minutes, 16 seconds of training, and ending on 98.46% for the convolutional net after 20 minutes, 25 seconds of training. The post-processing method used on Neural Network B failed to improve the results for this particular task. State of the art systems in MNIST digit classification typically perform somewhere in the 99%-99.7% range, meaning our accuracy is roughly 1% removed from the best purpose-trained systems.

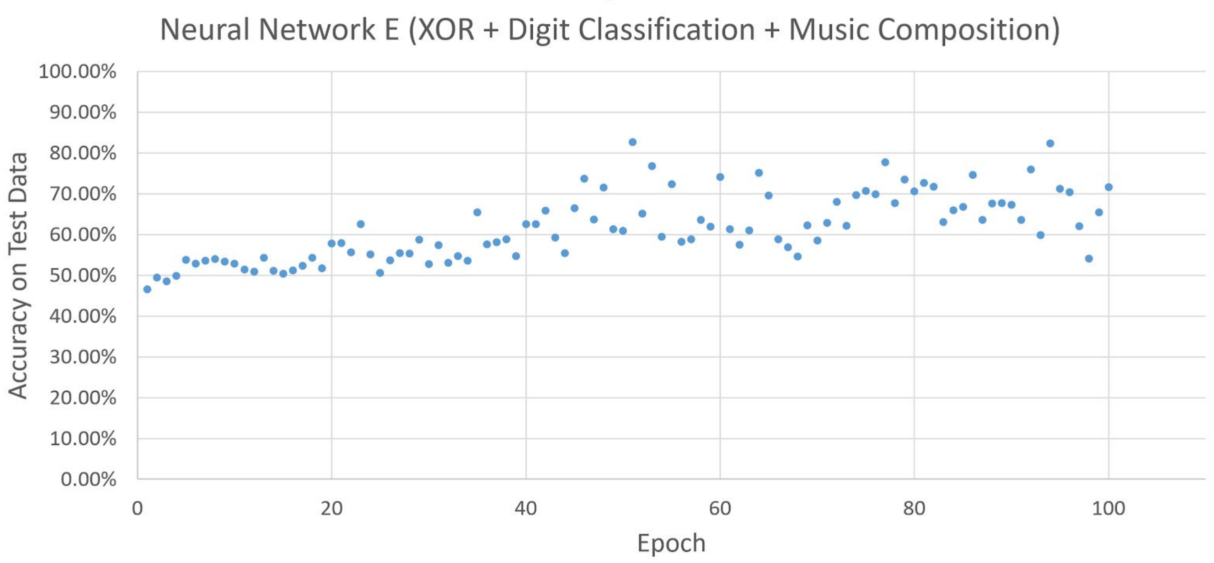
XOR + Digit Classification (**D**):



This graph plots the accuracy of the neural network on a test set that contained an equal mixture of digit classification problems and XOR problems vs the epoch number. As you can see it took many more epochs to reach a plateau when combined than either graph B or C did individually. However the end result was very positive. After 4 hours, 33 minutes, 33 seconds of training 200 epochs the neural network achieved 93.54% accuracy the mixed validation set. On a validation set that only included digit classification problems, the neural network achieved 95.36% accuracy. On a validation set that only included XOR problems, the neural network achieved 90.63% accuracy.

Using the same post processing that drastically helped the XOR neural network we increased the neural network accuracy of XOR to 100% and the combined accuracy on both digit classification and XOR to 97.82%. These results are very positive and indicate that a single neural network can effectively learn multiple independent functions.

XOR + Digit Classification + Music Composition (**E**):

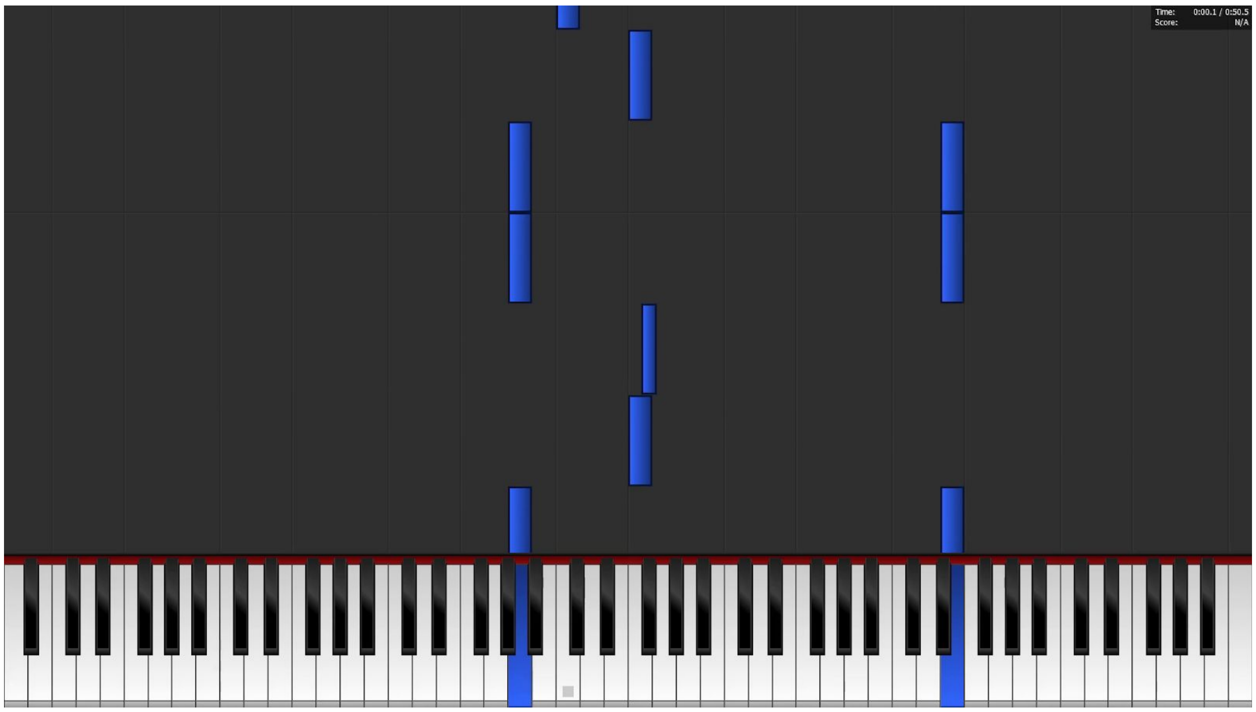


This neural network was trained is very similar to **D** except that it was also trained to generate music. The neural net shape increased from 816 inputs to 2596 inputs to accommodate the capability to train music creation. This change made the neural network much slower to train and less accurate on the two quantitative functions. The graph above shows the accuracy on a validation set that was equally composed of XOR problems and digit classification problems. It does not make sense to validate music creation since it is extremely difficult to quantify.

It took to 7 hours, 16 minutes, 41 seconds train 100 epochs. This is roughly 4 minutes 20 seconds to train a single epoch as compared to roughly 2 minutes 45 seconds of training on neural network **D**. This is 1.6 times slower to train when music composition is added. In addition to slower training, the accuracy was also appeared worse in neural network **E** than in neural network **D.** Even comparing the accuracy of **D** at 100 epochs to **E** at 100 epoch you can see that **E** only reaches 71.67% whereas **D** reaches 92.55%. Additionally **D** was much consistent between epochs than **E** was. However we can see what's causing the discrepancy when we look at the success rate of the XOR function and digit classification separately. On a validation set that only included digit classification, the neural network achieved 95.60% accuracy. On a validation set that

only included XOR problems, the neural network achieved 47.70% accuracy.
As you can see above, the majority of the error is coming from the XOR function. When we applied the same post processing that was applied in **B** and **D** the XOR problems were solved with 100% accuracy. When applied to the entire validation set (including XOR and digit classification problems) the neural network achieved 97.46% accuracy. This extremely similar to the 97.82% accuracy that was achieved by neural network **D** after its own post processing.

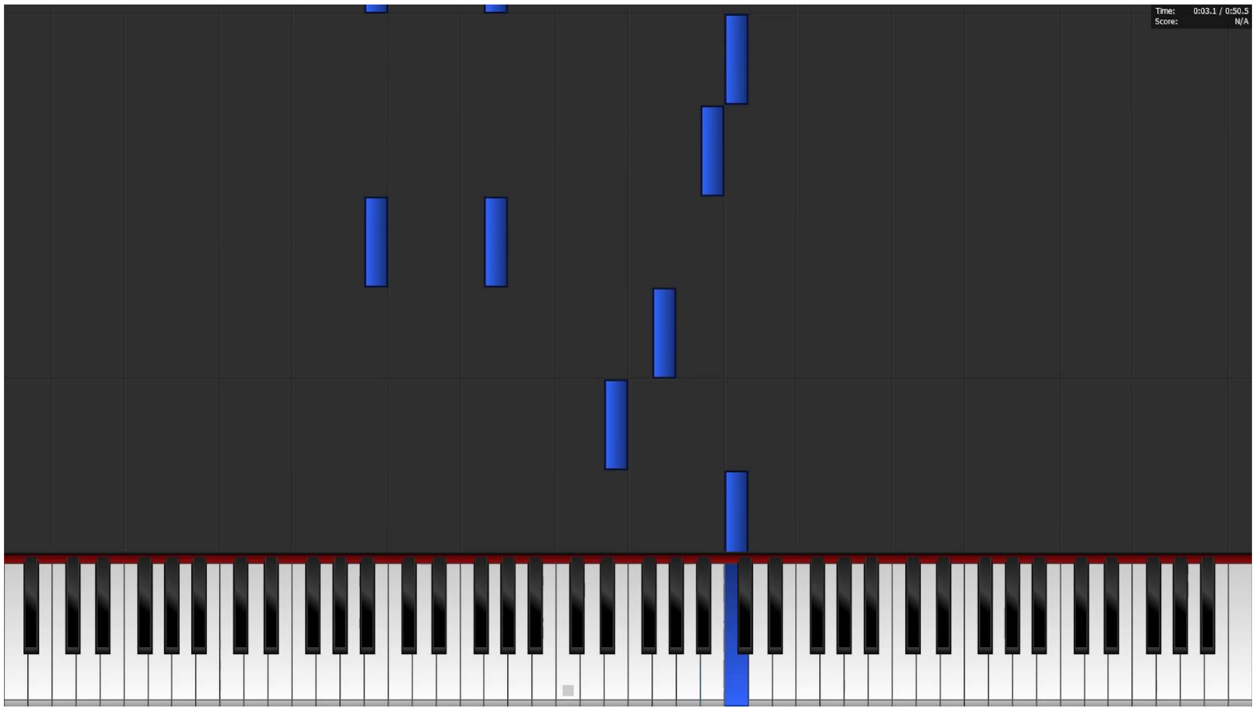
The music created by combined neural network was good but possibly subpar when compared to the baseline neural network **A**. Like in **A**, this neural network went through a few different iterations. Below is a snapshot of a song produced after training 100 epochs using classical music.



See video: https://youtu.be/lXl6zjEPeKA

This song suffered from the same problems that the first iteration in **A** did. It converged rapidly to a particular note or chord and looped until the end of the song. It also lacked many structural elements that are present in most songs.

Below is a snapshot of the second iteration of neural network **E**. This iteration was the same as the second iteration of neural network **A**: it used 20 final fantasy songs to train instead of classical music.



See video: https://youtu.be/BghOhYG_kDQ

This neural network produced songs of middle to low quality. However it did not converge as much as the first version and had slightly better musical structure.
Another example from same neural network: https://youtu.be/N605pn0iV1M

Lastly, the neural network was trained using much less epochs. This obviously affected the accuracy of the two quantitative functions since it had less time to train those functions. This was done to compare the music created after 5 epochs to the music created after 100 epochs. Generally the music sounded better on this version than the one that trained for 100 epochs. This is likely because we were overfitting the data on the songs and this made it difficult for the neural network to generalize.
Example for 5 epochs combined: https://youtu.be/5_y4GhngBfQ
Another example for 5 epochs combined: https://youtu.be/jnbN0NSCu5M

Discussion:

In terms of the initial goals of the project, we definitely were successful. We demonstrated that Multipurpose Neural Networks (MPNNs) are in fact viable. They were able to achieve similarly high accuracies to the single-task neural network baselines on both XOR and digit classification. Especially after our simple post-processing step, the accuracies jumped up to near 100% for each task.

We have shown definitively that training a single neural net to perform completely distinct tasks well is possible. The question still remains, however, as to the usefulness of these MPNNs. While MPNNs can perform nearly as well as their single-task counterparts, they take significantly longer to train than simply training multiple single-task networks. For example, our XOR and digit classification combined neural net **D** took roughly 7.5x longer to train than the XOR **B** and digit classification **C** nets combined to achieve similar performance. Thus, the actual benefit of MPNNs working with completely distinct and independent tasks is questionable, and their use cases would likely be rather constrained. Nevertheless, we have demonstrated some interesting capabilities of neural networks.

With regards to the music composition phase of our experiments, we also got some interesting results - the neural net appeared to learn the duple meter present in its training data, often grouping phrases in two-beat or four-beat sets. It also followed scales and appeared to have an underlying, often pleasant melody in places, despite the random notes thrown in making many phrases seem atonal. As expected, the music created by our neural nets came nowhere near the quality of human-written music, but some of the music produced demonstrated some rudimentary music theory concepts.