Musically: An Optical Music Recognition App

Yiwei Han, Annie Zhang, Angus Fletcher

1 Introduction

In recent years, with the advent of high-speed high-resolution scanners, there has been a drive in the digitisation of documents. Optical Character Recognition (OCR) is a well-known field with a plethora of readily available non-commercial and commercial products, however Optical Music Recognition (OMR) is generally understood to be a less established area of study and a more difficult problem that is still in active development.

Seven in ten children play an instrument, according to ABSRM [1], with 21% who learn through informal routes such as by accessing digital tools, peer-to-peer networks, or other self-teaching methods. OMR systems can be applied to early learning to assist in understanding and interacting with musical mediums.

2 Project Goals

Musically, the proposed OMR system, aims to provide a friendly interface that can take in sheet music and allow users to be able to listen to and interact with the visualisation in meaningful ways. Specifically, users will be able to (1) hear what the sheet music sounds like, in its entirety or at particular sections, (2) interact with the virtual sheet music visualisation, such as modifying notes' pitch or duration, and (3) visually see a marker indicating where the music is up to on the virtual sheet.

Musically aims to be able to identify the rhythm (duration) pitch (frequency) and dynamics (volume) of each note in the input sheet music. Although the parameters can be determined by identifying the symbols and their position on the staff lines, if modifiers are present in the local region around the notes, their meaning may be changed. This is a multifaceted and multidimensional problem as not only is it important to identify the symbols but also their context and positioning relative to other symbols.

3 Relevant Terminology

It is important to be briefly familiar with the terminology of musical symbols to understand this report. The different note types can be seen in figure 1 and the way different pitches (frequencies) are represented are shown in figure 2

4 Literature Survey

Almost since the dawn of computing computer-based music recognition has been a studied field. The first paper was [2] published in 1966, another early work was [3]. The review [4] gives an excellent summary for those looking for a deeper understanding of OMR's roots.

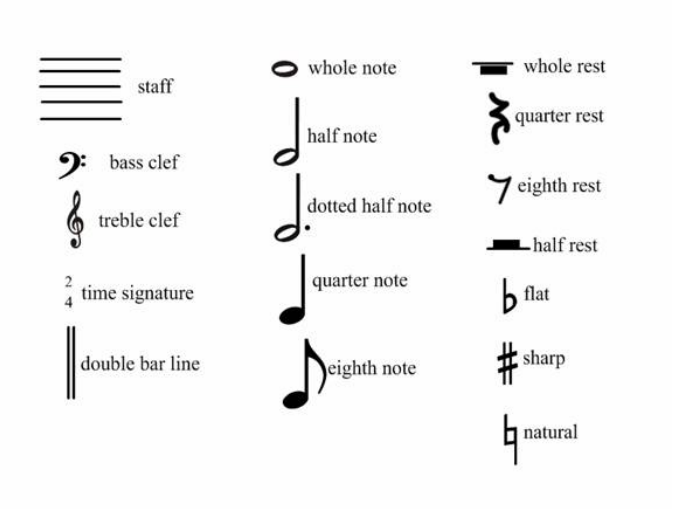

Fig. 1. Names of each note, which have different lengths of time

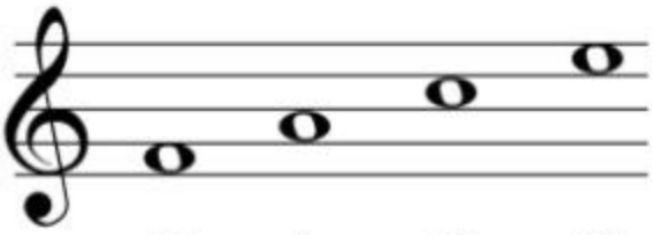

Fig. 2. Vertical offset changes the pitch of a note

Today the seminal paper is [5]. It provides an excellent summary of the research up to 2012. While there are many techniques and systems in use this brief summary will outline the key features of these systems, both those in literature and commercial use, as well as their shortcomings.

4.1 Difficulties of OMR

While OMR would appear to be a very similar problem to that of more traditional OCR it is actually both a much more difficult and less studied problem. The primary reason for this increase in difficulty is that many musical styles exist in which note modifiers - such as sharps, flats, naturals and staccatos - are not always presented in a standard way.

Furthermore, musical notes are often located in close proximity to each other on the staff and thus it can be hard to determine a symbols meaning without quite a high level of context. Previously this was not possible and care was

taken to segment images carefully, however, this limited the effectiveness of such systems [6]. Some recent work has tried to use a combination of CNN's and RNN's to solve this issue by attempting to provide more context [7]. However, even with modern computers, the increased context comes at a high computation price. The need for pooling layers in the CNN causes a degree of translational invariance, reducing the quality of results.

Furthermore, musical notes, even when correctly identified and attributed, must then be combined into a full piece which can be a challenging task for even a seasoned musician, given the variety of styles and fonts and musical rules. Several works have focused on the creation of musical grammars as further outlined in section 4.2.5.

4.2 OMR architecture

4.2.1 Preprocessing

Before any work must be done it is important to do some amount of preprocessing and binarisation. A range of papers have used many techniques including enhancement, binarisation, noise removal, blurring, de-skewing and perspective correction, and morphological operations. The list of papers utilising these techniques is extensive so readers are referred to [5] for a longer list of resources.

Of these techniques, binarisation is the most useful. [8] and [9] analysed the effectiveness of several techniques and concluded that the method in [10] was most effective, however, most papers use a standard Otsu threshold [5] and some papers claim it provided the best results [11].

[12] uses specific knowledge about musical structure to increase its effectiveness of thresholding, using the staff width and staff height and tried to maximise the number of thin black, long white runs of the same length. However, this is the only such paper that uses this approach.

4.2.2 Staffs processing

To effectively segment the image it is necessary to remove the staff lines for almost all systems bar some ([13]. [14]) The classic trade-off is that this filtering allows for better segmentation at the risk of fragmenting symbols[15].

There are indeed several approaches to this but most often morphological operators[15] or Hough transforms[16] are used. These methods become less effective in the presence of deformation of the staff lines.

4.2.3 Symbol processing

Once removed a simple connected components algorithm can be used to identify each object [15]. However, using a Marker-controlled watershed algorithm [17] or a P-algorithm may be more effective and preventing fragmentation [17]. For lower quality scans a series of rules must also be used to combine disjoint components [15].

This step is often combined with classification. Another key distinction at this step is ether or not to use raw pixel values (either with or without the staff) or to generate high-level features.

4.2.4 Classification

Classification is obviously the key step to such a system. Many methods are available for use here. Some more traditional systems use a linear classifier and the k-nearest neighbours [15]. [18] uses an interesting combination of high-level features such as location and number of vertical and horizontal lines to feed into several different SVMs optimised for each different subsection of musical symbols.

[7] uses a CNN to creature a 256 element feature vector that is fed into an RNN. [16] also uses neural nets.

Some methods like [13] use Hidden Markov models to combine all of the last 3 steps into a single step, but this hasn't been generalised beyond simple scores.

Mathematical frameworks are sometimes used and template matching is also a common technique[5].

4.2.5 Music reconstruction and grammar

One issue is associating symbols together, especially modifiers. Interestingly, the same shape symbol can mean different things based on how it is related to its base note, for example, a small dot below or above a note means to shorten, but to the right or above of a note it means to lengthen. This complicated nature has lead to the creation of several formal grammars to help relate this music together[2].

4.2.6 Final representation and structure

Several papers such as [7] develop their own final representations but several standards exist. As discussed, to fully capture a note there are 3 key properties to record: pitch, duration and volume. MIDI files can be used to encode this information into an audio form. MusicXML (mxl) is also often a common output format [5].

4.3 Document Identification strategies for Neural network design

When designing a neural network for image detection like this the key is generating a large amount of labeled data to train on. [19] provide both a suitable structure for a neural network to train on the MNST dataset (which is similar to our own) and also techniques to artificially increase the amount of samples using elastic deformation which they used to much increase the quality of results on the MNST dataset.

4.4 Perceived shortcomings of current research

Overall the greatest failings of the outlined papers is they do not (with the exception of [12] and [13]) use the fact that the processed image is in fact sheet music. If knowledge of this structure could be used it may be possible to create new methods that require less computing power, but with the same level of accuracy. In a way this is what a neural network like [7] is trying to do but it is not possible to take in enough context without being too computationally expensive. We can use our domain knowledge of music structure to include relevant context, while maintaining low overhead.

4.5 Commercial products

Commercial products which conduct OMR exist, such as PhotoScore, SharpEye and SmartScore. Overall the accuracies of such products are still fairly low [7].

5 Problem Decomposition

The proposed application will execute the following tasks to convert an image of a sheet of music into the sounds it represents.

1) The sheet music image is preprocessed. This includes operations such as filtering, noise removal, binarisation, and transforming of the image's perspective in case it is not upright.
2) Key parameters of the image, such as staff height and staff width, are identified, which will parametrise the extent of processing in the subsequent steps.
3) The staff lines must be removed in order to prevent them from interfering with the identification of musical symbols.
4) Once the staff lines are removed, the remaining objects are the musical symbols. The symbols are segmented, and each individual symbol extracted. Each individual symbol is then closely studied to identify its local meaning.
5) All symbols are collated and musical syntax and grammar applied to them, such that the entire set piece has been reconstructed in the application. A MIDI file is generated which correctly plays back the piece of music displayed in the original sheet music image.
6) A user interface allows the user to interact with the application, such as opening images of the sheet music, starting or stopping the music playback, and selecting specific notes to play.

6 Scope

The scope of this project is to create a fast and simple interface for Optical music recognition. Given the limited time frame the target users of this program are beginner musicians who only know a subset of the musical language, reducing the amount of complex grammar rules we need to encode into the solution.

The second goal of this project is to attempt to find an application of domain knowledge to the OMR problem, something that is currently not commonly done. This is a technique often used in other fields and we would like to test the feasibility of using similar ideas on this problem.

We will test our results both visually on scanned pieces of music and algorithmically on a large amount of auto-generated music. Results will be in 3 categories: fully correct, correct pitch only, correct duration only.

7 Overview

An overview of the general use case is as follows. A user uploads an image via the front-end user interface, which is sent to the server and cached. The image is then processed by the Segmenter, which returns Symbols pertaining to the image. The symbols are predicted by the Classifier using its trained keras model, then finally a musical representation is constructed and cached. MusicXML is used in this system as it is a highly available standard for representing all music symbols and constructs in a serialisable format. The process is shown in Figure 3.

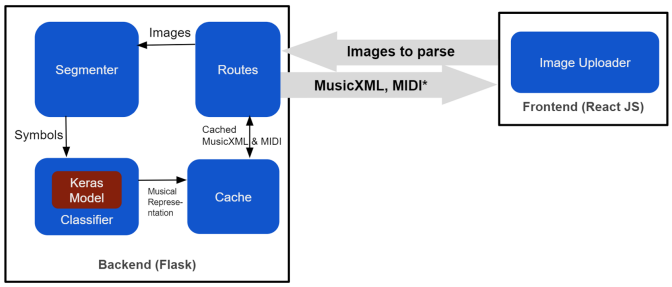

Fig. 3. Overview of the system.

7.1 Staff line removal

Staff line removal is a critical step and one that is often the first to fail on deformed pieces as outlined earlier. This project developed a line following algorithm that provided far Superior results to traditional methods outlined earlier that use Hough transforms or morphological operators. There are several steps involved:

1) Identify the most common run length of black and white pixels, this will the the width of the staff lines and the space between them.
2) identify seeds for the line follower. These are discovered by a simple finite state machine that looks for 5 consecutive runs of black then white pixels with the lengths found in step 1 ($\pm$ a small amount of error).
3) Lines are the grown out from these seeds, they seek to maintain the minimal sum of all the pixels that are part of the staff.
4) If the algorithm detects that it is surrounded by too much black (as there is a note head) it will instead move forward rather than follow the line. This stops the lines merging by following note heads down to the line below or above.
5) Linearly interpolate and remove pixels on the staff line, places with lots of black however are note removed (meaning notes aren't cut in half).

The novelty of this approach is the decision to not always follow the line, as the merging that happens when you do this is why the literature have not used this method in many projects. Our approach allowed us to have 95% more vertical deformation that the techniques used in [7]. An example of a very extreme example of this is shown in figure 5 and 4.

7.2 Creation of training data

The initial goal was to have access to a large database of labeled music, as used by previous works, however all requests for access and API keys were ignored. Instead it was decided to create an artificial data source. The random music generator outline in section 7.3 was used to generate 110 pages of music (around 32,000 symbols). The XML could then be parsed in association with the image segmenter to create labeled images. This was then augmented using the *imgaug* library. Each class was augmented until it had 10,000 images. Resulting in a total of 110,000 training images. The following transformations were done:

1) Scale $\pm 10\%$

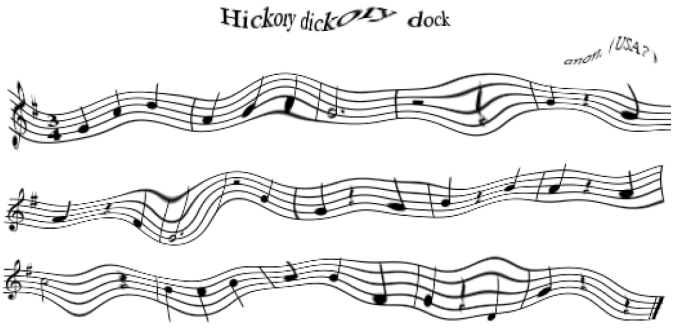

Fig. 4. An extremely deformed piece of music

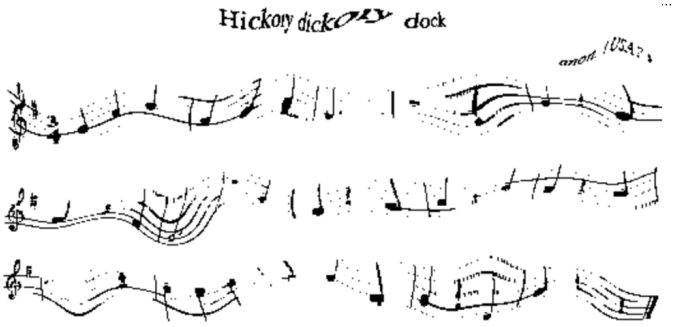

Fig. 5. The lines mostly removed

2) Translation $\pm 5\%$
3) Rotation $\pm 2\%$
4) Shear $\pm 4\%$
5) 80% of the time elastic transformation[19] amount from a normal distribution with mean 2% and std dev 0.5
6) 85% of the time Gaussian blur of sigma 0 to 1.2
7) Additive gaussian noise of sigma 5% of max intensity in image.
8) 20% salt and pepper noise over $5-10\%$ of the image pixels

7.3 Random music generation

A random music generator was created. It would select random notes and duration's to create large pieces of MusicXML. Duration's were selected such that all musical rules were followed (each bar must sum to a specific value). Notes were selected from a Gaussian distribution centered around the previous note, this gave music that had the appearance of realism, although it did not follow many musical rules like chord progressions and repeating Melody. It did however include a large variety of the of common patterns found in music.

7.4 Segmentation

After the staff lines have been removed from the music sheet, the remaining musical symbols is extracted. Without the staff lines in the sheet image, these symbols appear as unconnected black objects on a white background.

The connected components algorithm was used to to create a pixel-wise label for each object. The image was first binarised and inverted so that the image represents white objects on a black background. Then, the special connected components with statistics function from OpenCV was called to create a numerical label for the objects. The statistics of each object include properties such as pixel area, centroids, and bounding box coordinates.

Because some note types had pixels which were overlapping with the staff line pixels, these notes' structures have been separated due to the staff line removal, and therefore their components now have different labels. To recombine the separated components, the statistics of each component was compared against each other to find components which seem to belong to one single symbol. First, the distances of the centroids were compared to see which components are close to each other, then a more thorough nearest-pixel-distance metric was computed, and finally, bounding box and area sizes were compared (depending on the symbol type). Components that were computed to belong together merges their labels into a single numeric value.

At the conclusion of this stage, the sheet of music has been converted into a list of musical symbols each containing the sub-image that contains the symbol, and its coordinates within the original image.

7.5 Neural network classifier

A neural network (NN) was chosen as the method to detect a symbol's type. Keras, a high-level neural networks API, was used with a Tensorflow backend, as it is easily adaptable and requires less configuration, suited to our needs for a simple NN model. Also, Google Colaboratory, a hosted Jupyter notebook environment, allowed training to be done quickly using Google readily available Tesla K80 GPU. Colaboratory also allowed numerous training sessions to be ongoing simultaneously, enabling different models to be tested at the same time.

The full specification of the network chosen is as follows. The network consists of two convolution layers of size 32 with L2 regularisation and a relu activation function. Max pooling is applied before a fully connected layer of size 128 with a relu activation, followed by a final softmax layer.

The test and training data was split 1:4, and had undergone some data transformations, such as rotation, flipping, scaling in the independent axes, Affine warping, elastic deformation, salt and pepper noise and additive Gaussian noise. This data augmentation was done using the imgaug python library which extends keras's default data augmentation methods. Data augmentation was used to artificially increase the amount of data applied to the network, therefore with small amounts of data, good performance could still be achieved.

7.6 Pitch detection

Once symbols had been classified as notes by the neural network it was necessary to determine their pitch (vertical offset). This could not be done by the neural network itself as it is intrinsic the the nature of the neural network to be translation invariant, one of the reasons for the weaknesses in [7]. The note head was instead found using template

matching, this could then be used to work out the vertical offset from the staff lines (at that x offset) whose locations were known from their earlier removal. This was another advantage by using the line following staff line method over filtering as it allowed accurate pitch detection even when staff lines were warped or sloped and thus find it hard as staff line locations at specific x offsets are not known.

7.7 Song reconstruction

Once all notes have been identified and their duration's and pitches calculated it is necessary to construct them into a song. There are several steps to this process:

1) Add either Bass or treble clef
2) Add time signature
3) Add individual notes with duration based on the neural network and pitch based on the template matching of the note head. Depending on the clef (Bass or Treble a different tone is used).
4) Convert into musicXML to send to the client
5) Convert into midi file to send to the client

7.8 Novel domain knowledge note correction

One of the key weakness we identified in the literature was a lack of domain knowledge being used in the reconstruction. This is something that has been critical in increasing results quality in other related fields, as was highlighted in the lectures.

To this end we decided to try and apply our knowledge of musical structure to increase accuracy.

In music a piece is divided up into a number bars, indicated by the vertical lines on the staff. Each bar must contain notes whose duration add to that specified by the time signature. When analysing the results of the output of our system it can bee seen that the time signature and bar lines are detected with what appears to be 100% accuracy. Notes however are sometimes (rarely) mistaken. When this happens it is often clearly recognisable as the sum of a bars duration's will not match that of the time signature.

This is the domain knowledge that we will exploit this this technique. Once we identify the bar has a mistake we know it must be with one of the notes within that bar (most likely) since the previous bar added correctly.

We now need a way to re identify any notes that must be wrong. Fortunately we already have a second method since we used template matching during the pitch detection step. Instead of using just a generic note head we used the note head of each type of symbol (since they are mostly different between notes. The best match for the note head is stored when this pitch is calculated.

If a bar does not sum correctly the algorithm will search for a note within the bar where the template matcher gave a different note type (and hence duration) than the neural network. If so it recalculates the note and checks if this better matches the bar.

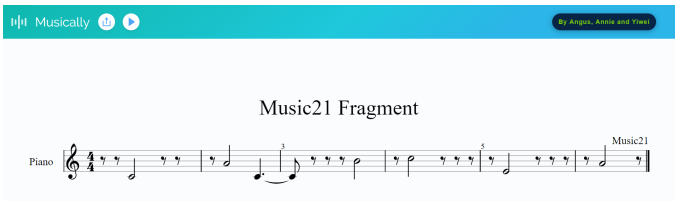

Fig. 6. The user interface.

If after this process the bar is still not correct it will insert rests until the bar is the correct length. While this seems like we are increasing the error (and we are) this will help with the visual appearance, when a bar is too short the music visualation software used will make the entire song look completely different as it tries to account for this mistake. Since our target market is beginners this seemed like an important decision to make.

7.9 User Interface

The user interface comprised of an uploader, music controls and a visualisation of the sheet music played. The uploader allowed the user to drag and drop an image to be uploaded, and after processing and sending back a MusicXML and MIDI, the XML was shown on the screen, with music following along. This can be seen in Figure 6.

The Flask microframework is used for the server, and React is used as the library for front-end development, as both are highly available and modern technologies.

For data transfer, images are transferred to the server through a serialised byte stream, while for the output, MusicXML is sent through a string format with MIDI transferred using a Base64 data format. All these methods are technically efficient for their purposes, which meets our need for a responsive user experience.

7.10 Implementation Issues

7.10.1 Music XML to image

The program used to do this was MuseScore, a major issue encounted is that Musescore had a seemingly random process for deciding to combine eighth notes into tied eighth notes

8 Testing and Results

For the ground truth in our testing framework, 200 measures of music was randomly generated and a Music XML and a corresponding set of images of the music sheets were generated. The framework used these images and our recognition program to generate new Music XML file.

The framework compares, measure by measure and symbol by symbol, the durations and pitches of the symbols present in both XML files. All symbols have duration, but a large subset are notes and have a pitch value.

The accuracy metrics used to evaluate our program is as follows:

- Both pitch and duration for a particular symbol was recognised correctly.
- Pitch was recognised correctly (duration is ignored).

- Duration was recognised correctly (pitch is ignored).

For the 2000 measures generated, we achieved the following results for the above metrics:

- Accuracy of both pitch and duration = 93.56%
- Accuracy of Pitch only = 96.75%
- Accuracy of duration only = 97.48%

These results are comparable with those obtained in the literature. Lastly the neural network itself was trained until it achieved a 99.3% accuracy on the validation data. The fact this is higher than the numbers above show we most likely had some overfitting of our model.

9 Conclusions and Future Work

Overall we have successfully developed a OMR system that has very high accuracy. We successfully trained a neural network to identify symbols. Furthermore we developed a novel and highly robust staff removal technique. The application of domain knowledge was a novel approach in improving accuracy. Lastly we created a novel and easy-to-use user interface so that users can interact with the system.

Future work and extension features include:

- Extending the recognition program to be able to recongnise more advanced symbols including accidentals, sharps, flats, etc. This can easily be achieved by extending the neural network model to allow these new symbols to be trained and predicted.
- The neural network was only trained on artificial data. We could allow the neural network model to be trained by users. This is where users of our program can correct any mistakes made by it, and the program can automatically retrain itself.
- The domain knowledge reconstruction method was only applied on a per bar basis. It could be extended to frame the entire piece as a global optimization problem. Further more classifiers could be added and they 'vote' on the correct output such that the piece is has the fewest number of no matched bars overall.

References

[1] ABSRM, "The statistics," 2018. [Online]. Available: https://gb.abrsm.org/en/making-music/4-the-statistics/

[2] D. Pruslin, "Automatic recognition of sheet music," Ph.D. dissertation, Massachusetts Institute of Technology, 1966.

[3] D. PREARU, "Computer pattern recognition of standard engraved music notation," *Ph. D thesis, Massachusetts Institute of Technology*, 1970.

[4] M. Kassler, "Optical character-recognition of printed music: A review of two dissertations," *Perspectives of New Music*, vol. 11, no. 1, pp. 250–254, 1972. [Online]. Available: http://www.jstor.org/stable/832471

[5] A. Marcal, J. Cardoso, C. Guedes, A. M. Rebelo, I. Fujinaga, and F. Paszkiewicz, "Optical music recognition-state-of-the-art and open issues," 2012.

[6] P. Pecina *et al.*, "In search of a dataset for handwritten optical music recognition: Introducing muscima++," *arXiv preprint arXiv:1703.04824*, 2017.

[7] E. van der Wel and K. Ullrich, "Optical music recognition with convolutional sequence-to-sequence models," *arXiv preprint arXiv:1707.04877*, 2017.

[8] L. Pugin, J. A. Burgoyne, and I. Fujinaga, "Goal-directed evaluation for the improvement of optical music recognition on early music prints," in *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*. ACM, 2007, pp. 303–304.

[9] J. A. B. L. P. Greg and E. I. Fujinaga, "A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources," 2007.

[10] A. Brink and N. Pendock, "Minimum cross-entropy threshold selection," *Pattern recognition*, vol. 29, no. 1, pp. 179–188, 1996.

[11] O. D. Trier and A. K. Jain, "Goal-directed evaluation of binarization methods," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 17, no. 12, pp. 1191–1201, 1995.

[12] T. Pinto, A. Rebelo, G. Giraldi, and J. S. Cardoso, "Music score binarization based on domain knowledge," in *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 2011, pp. 700–708.

[13] L. Pugin, "Optical music recognitoin of early typographic prints using hidden markov models." in *ISMIR*, 2006, pp. 53–56.

[14] R. Göcke, "Building a system for writer identification on handwritten music scores," in *Proceedings of the IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA)*, 2003, pp. 250–255.

[15] I. Fujinaga, "Staff detection and removal," in *Visual Perception of Music Notation: On-Line and Off Line Recognition*. IGI Global, 2004, pp. 1–39.

[16] H. MIYAO and Y. NAKANO, "Note symbol extraction for printed piano scores using neural networks," *IEICE TRANSACTIONS on Information and Systems*, vol. 79, no. 5, pp. 548–554, 1996.

[17] S. Beucher, "Image segmentation and mathematical morphology," 2010. [Online]. Available: http://cmm.ensmp.fr/ beucher/wtshed.html

[18] T. Nguyen and G. Lee, "A lightweight and effective music score recognition on mobile phones." *JIPS*, vol. 11, no. 3, pp. 438–449, 2015.

[19] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, vol. 3, 2003, pp. 958–962.

Appendix

Contributions of Group Members

Annie **Neural Network — 3 days.** Cleaned and wrangled labelled datasets. Created a CNN using Keras, executed training of the model, and saved the trained model from Google Colaboratory. Various parameters were explored, such as parameters pertaining to the model; data augmentation and data transformation variables determined the extent of the training data.

User interface — 6 days. Created a the user interface, allowing users to upload music images and play their MIDI representation. Created the back-end server. Created an API, including a cache which utilises various components of the prediction system to optimise for speed.

Staff line removal — 2 days. Worked on preliminary exploratory methods for staff line removal.

Angus **Neural network training data – 3 days** Generated and parsed music to create labeled data for training.

Neural network training – 2 days Modified the neural network with increased dropout and L2 regularization to reduce over fitting.

Staff line removal – 2 days Invented a novel method for staff line removal that is more robust than most of the current leading research as it is robust to affine and elastic deformation unlike traditional methods which require straight lines.

Pitch detection – 1 day Used template matching to find note heads, used removed staff lines to find the pitch.

Song reconstruction – 1 days reconstructed the labeled symbols into a music21 stream so it can be used by the server for generating XML and Midi files.

Domain knowledge note correction – 3 days Used knowledge of music structure to choose best result from either the neural network or the template matcher.

Random music generator – 1/2 a day Improved the music generator by making it a Markov chain, where each note was based on the previous note, making it sound like more realistic music.

Yiwei **Random music generator — 2 days.** Created a random music generation program which can generate a series of random musical symbols on a sheet for training and testing purposes, and export the sheets as images. Ensured the music obeyed musical rules and note and bar lengths.

Segmenter — 5 days. Implemented the segmentation algorithm to segment and extract individual musical symbols from a sheet with stave lines removed.

Testing framework — 3 days. Implemented a testing framework to test the accuracy of recognised music against the ground truth.