­

Jazz Piano Lead Sheet Arrangement via a Conditional Generative Adversarial Network

Elliot Hogg

200817750

David Herbert

Mark Turner

Jim Mcgrath

DD/mm/yyyy

15000 Words

Declaration

I hereby declare that this dissertation represents my own work except where otherwise stated.

Abstract

Over the past 10 years, deep learning has been extensively researched and used in the task of music generation. However, one area that remains mostly untouched, is the task of lead sheet arrangement. In the genre of Jazz, most music is represented in the lead sheet format – a single notated melody and chord symbols. The interpretation and arrangement of the music must be decided and undertaken by the performer. To tackle this problem, a novel conditional generative adversarial network (C-GAN) was designed to generate jazz chords which could be used to create full arrangements. Participant lead evaluations showed that resulting arrangements were found to be more effective than template chord selection.

Acknowledgments

I would firstly like to thank my supervisor David Hebert, whose support and advice was invaluable to this project.

I would also like to thank Mark Turner for his continued help and support throughout the project.

Contents

1. Introduction (1700 words)

2. Background Research (1700 words)

(6600 words):

3. Data

4. Implementation

4.1. XML Chord Scraper

4.2. Deep Learning Models

4.3. Lead Sheet Arranger

4.4. Complete System ??

(3300 words):

5. Results and Evaluation

5.1 Scraper Accuracy

5.2. Deep Learning

5.3. Arrangements

5.3.1. Participant survey

5.3.2. Technical evaluation

(1700 words):

6. Conclusion

6.1. Meeting aims and objectives

6.2. Effectiveness of approach and systems

6.3. Contribution to the field

6.4. Future Considerations

1. Introduction

Lead sheet arrangement is a task whereby a musician takes an abbreviated representation of a song and adds harmonic and rhymical elements to transform it into a compelling piece of music. In the area of jazz music, the lead sheet is the traditional and most common way for songs and pieces of music to be notated, mainly due to the fact that they allow for creative freedom and leave much to the interpretation and direction of the musician, an idea which is essential to jazz music. Figure 1 shows the first 4 bars of a piano lead sheet.

Lead sheet arrangement can be divided into two subtasks: (1) Harmonic – choosing a set of notes to be played for each chord symbol. This task is also referred to as *chord voicing*. (2) Rhythmic – temporally arranging chosen chord notes to form a rhymical pattern. Figure 2 below highlights these subtasks.

A black and white photo of a piano

Description automatically generated with low confidence

Figure 1. The first 4 bars of the lead sheet for the song Over the Rainbow by Errol Garner

Diagram, engineering drawing

Description automatically generated

Figure 2. Lead sheet arrangement of the first bar from figure 1. From left to right, a C chord symbol is first represented harmonically, and then arranged rhythmically.

Computationally, lead sheet arrangement can be defined as a process that takes a lead sheet as input, and outputs a harmonically and rhythmically complete digital representation of that song.

There are a few commercial applications, such as *Band in a box* and *iRealPro,* that perform the harmonic element of lead sheet arrangement, however they are reliant on simple selection algorithms and a limited dictionary of chord voicings. The chord voicings themselves are generic and lack musicality. There are however no applications that attempt to perform the rhythmic subtask of lead sheet arrangement, arguably due to this being a much more complex task (citation).

There has been some academic research of traditional programming approaches to the harmonic task of lead sheet arrangement that have shown to outperform the aforementioned applications, however the results of traditional programming approaches lack variation, as they struggle to output multiple different sets of notes for the same chord symbol (citation). They also rely on being preprogramed with a set of musical theory rules.

The use of deep learning in lead sheet arrangement has seen very little academic research. There exists only two publications on the topic, both of which have the same authors. There has also been some recent publications using deep learning in the task of chord voicing generation, which is essentially the same task as the harmonic arrangement task in lead sheet generation. These papers are critically evaluated in section 2.x.

Although there is limited research on the task of music arrangement, the task of musical generation has seen a large amount of attention (citation). This literature is valuable to this project, as arrangement can be defined as a subtask of generation. Surveying an evaluating recent music generation research can greatly contribute to the direction of this project. Especially in regard to gaining an insight into which deep learning algorithms have been successful in the task of music generation, and thus arrangement.

And the focus on music composition within the field of deep learning has

has regained a large amount of academic and public attention within the past 10 years (citation). This is primarily due to the development of deep neural networks (Ji et al., 2020: 5). One great example of such generation is *DeepBach*, a deep learning model that can compose novel, convincing chorales in the style of composer Bach (Hadjeres et al., 2017). Deep learning has also been used to generate monophonic and polyphonic music of many different genres (citations needed).

Research and implementation of deep learning in the task of lead sheet arrangement is however very limited. There exist only two publications on the topic, both from the same authors. In their research, they employed deep learning to arrange pop lead sheets into a full arrangement for a 4 piece band (reference).

Music generation tasks are usually achieved using either recurrent neural networks (RNN) or generative adversarial networks (GAN). The former type of network is highly effective at generating temporal sequences, as the networks outputs are dependent on previous outputs (citation). The latter network is effective at generating completely new instances of data that are undistinguishable from a given set of data (citation).

The inherent nature of the lead sheet is that, once a musician has composed an arrangement, they tend to memorise it rather than notate it (citation). This is a particular issue in jazz, in which there exist very few arrangements of jazz standards in relation to the size of the total corpus (citation). For intermediate or amateur jazz musicians, or for advanced classically trained musicians, the task of lead sheet arrangement can be challenging, and can create an effective barrier to entry for the genre as a whole (reference).

Harnessing the power of deep learning in order to create a system capable of generating compelling arrangements would be invaluable to the jazz community, as it would make jazz music much more accessible.

The focus of this project is to create a lead sheet arrangement system that will take a lead sheet as input and output an arrangement. The system will also be intended to provide a framework for others to employ and use in order to advance the field of research. The system will be limited in that it will only address the harmonic element of the process, ie. It will provide chord voicings for the chord symbols.

The system will consist of three key elements. (1) A chord scraper that is capable of extracting chord voicings from fully arranged songs with chord symbols. This scraper will be limited to songs in MusicXML format, which is a popular XML based music encoding language (reference). (2) A chord generator that will use a conditional generative adversarial network to generate musically pleasing and convincing chord voicings. The network will be trained using a novel dataset of labelled chords gathered by using the chord scraper on existing jazz piano arrangements. (3) A lead sheet arranger, which will take the melody from the lead sheet and the chords from the generator, combine them, and output an arrangement in MusicXML, MIDI, and Audio formats.

The project will also present the novel dataset of labelled chord voicings for use in further deep learning tasks.

**1.2 Aim**

To create a deep learning led system that can generate arrangements of any given jazz lead sheet.

**1.3 Objectives**

* To develop a program that can scrape chord voicings from fully arranged pieces of piano music with chord symbols in MusicXML format
* To present a novel dataset of labelled jazz piano chord voicings
* To create a novel conditional generative adversarial network that can output compelling colourings and voicings for chord symbols from a given lead sheet
* To develop a program that can combine chord voicings with the melody of a lead sheet in a way that is both convincing and playable.
* To perform some empirical and participant led testing on the outcomes.

**1.4 Outline**

The structure of the report follows both the order it was developed in as well as the order in which the system as a whole operates.

* **Chapter 2** provides a review of the recent literature within the field of deep learning as it pertains to music. There will also be a critical evaluation of existing lead sheet arrangement models.
* **Chapter 3** provides information about the data used in the project.
* **Chapter 4** provides a deep dive into how the three key components of the system were conceptualised and developed. Starting with the chord scraper, then the C-GAN deep learning model, and finally the lead sheet arranger.
* **Chapter 5** presents an evaluation of each component, as well as an evaluation of the results of the system as a whole. Evaluations of the system are both technical and participant led.
* **Chapter 6** presents a conclusion of the project, with commentary on the how initial aims and objectives were met, as well as some future considerations and recommendations for further research.

2. Background Research

2.1 Motivations

The initial motivations for this project were to make a meaningful and novel contribution to the jazz piano community within the context of a computer science project. As a keen jazz pianist with a classically trained background, the task of lead sheet arrangement has always been a challenge. In order to make an arrangement sound good, a large amount of knowledge and experience in voicing chords is required. A system that could generate chord voicings that were indistinguishable from the voicings of renowned pianists such as Bud Powell, Bill Evans, or Herbie Hancock would remove the barrier for entry to a large number of aspiring jazz pianists.

This is the motivation that led me to pursuing a research project in the field of deep learning and lead sheet arrangement. I hope that this research can at the very least, serve as a springboard for future research to be carried out in this area.

2.2. Generative Adversarial Networks (GAN’s)

Generative adversarial networks are a type of neural network developed by Ian Goodfellow and his colleagues in 2014 (Goodfellow et al., 2014). For the reader to understand this paper, a high-level explanation of GAN’s will be provided. The purpose of a GAN is to be able to generative new instances of data from a given data distribution. For example, if a GAN was trained on a dataset of images of faces, it would then be able to generative completely new images that would exist in a similar distribution space to the training dataset.

A GAN is made up of 2 independently functioning neural networks, a discriminator, and a generator. (Goodfellow et al., 2014: 3).

The generator is a neural network that takes a noise distribution as input, and outputs a fake instance of data. To use the previous example, the generator would output fake images of faces. Initially, the fake images would be very poor, however through training the generator, it would eventually be able to output fake images that were convincing enough to be indistinguishable from images in the dataset.

The discriminator is a neural network that operates as a classification network. It takes as input both real instances of data, and generated, fake instances of data, and outputs a number indicating whether the input was real or fake.

When initially training the GAN model, the discriminator easily classifies the real instances as real, and the generated instances as fake. However, throughout training, the generator improves to produce more and more convincing instances, to the point where the discriminator is unable to effectively classify real instances as real, and fake instances as fake. At this point, the generator can produce completely novel instances of data from the given dataset distribution.

2.3. Conditional Generative Adversarial Networks (C-GAN’s)

Conditional generative adversarial networks allow the output of GAN’s to be conditioned on a given input (citation). Extending the example from 2.2 to having a labelled dataset of faces made up of faces of men, woman, girls, and boys. When training the generator, instead of passing in a noise distribution, the labels are passed in. In doing so, the generator is conditioned to effectively learn the distributions of each of the four labels. Thus, when generating faces using the trained network, one of the four labels could be passed into the generator, resulting in a generated face of that labels distribution.

2.3. Critical evaluation of lead sheet arrangement literature

2.3.1 Lead Sheet Generation and Arrangement by Conditional Generative Adversarial Network

2.3.2 Lead Sheet Generation and Arrangement via a Hybrid Generative Model

2.3.3 Chord Jazzification: Learning Jazz Interpretations of Chord Symbols

2.4. Critical evaluation of relevant conditional generative adversarial network literature

2.4.1 Image-to-Image Translation with Conditional Adversarial Networks

2.4.2 Conditional Generative Adversarial Nets

2.4.3 1D conditional generative adversarial network for spectrum-to-spectrum translation of simulated chemical reflectance signatures

2.5. Survey of available datasets

Table

Description automatically generated with medium confidence

Classification model notes

Represented the data in 3 different ways and used RNN and CNN to run classification tasks

Report results…

Look at where label classifications were different that expected -> are they in any way more accurate? Could this potentially be used to classify chords without chord symbols in XML files?? To gather more data??

GAN notes

**Embedding:** transform indexes into a vector of fixed size. E.g., representing words as numbers

**One-hot embedding:** create a “sparse” binary vector that is the length of the dictionary (100 words = 100 length vector). Then each word is represented by a 1 instead of a 0 in the vector.

\_\_\_\_\_\_\_\_\_\_\_

https://www.youtube.com/watch?v=BUNl0To1IVw&t=3018s:

Generative models:

Autoencoders and Variational Autoencoders (VAEs):

Learn lower-dimentional **latent space** and **sample** to generate input reconstructions

GANs:

Don’t explicitly model density, and instead just sample to generate new instances

When generator is trained – it is learning to transform from distribution of noise to target data disttibution

Gaussian noise 🡪 target data manifold

Z 🡪 Y

Take on point from latent noise distribution – will result in particular output in target data space.

The points in latent noise distribution represent different outputs

If we transverse in the Gaussian noise space we get interpellation of output (2 outputs mixed together)

55:10 – conditional GAN

Introduce pairs to G and D

Paired translation

12 vector integer notation -> 88 vector full representation

Effective distribution transformers

\_\_\_\_\_\_\_\_\_\_\_

**How c\_gan works exactly:**

The generator takes in a label and outputs a prediction

Label -> **Generator**  - > Fake Chord

Label + Fake Chord || Real Chord -> **Discriminator**  - > Real or Fake

The discriminator takes in pairs of labels + fake / real occurences and outputs a value on whether or not they are real

After the model is trained we give the generator labels and it gives us predictions

Adapted pix2pix model:

Transformed chord notes and label data into 7x12 matrices (omitting bottom 3 notes and top note)

Adapted pix2pix model to contain 1D layers as oppose to 2D layers (see paper)\

Initial results:

The loss of the discriminator on both real and generated data rapidly went to near zero, why? Probably the generator

**Novel model**

“Both G and D could be a non-linear mapping function, such as a multi-layer

perceptron.”

**Discriminator**

Discriminator that takes in concatenation of labels and either fake/real chords

How should this data be represented/concatenated?

Look at results of classification tasks

When concatenating label and chord, should an extra dimension be added?

Discriminator should have a single output with sigmoid activation (value between 0 and 1 indicating whether or not it is fake/real)

If using 1d representation, should concatenation be on a third axis, i.e. (88,) + (88,) 🡪 (1, 176) **OR** on the x axis, ie. (88,) + (88,) 🡪 (176,)

How can this decision be made? Other than just testing it, is there some reasons for why one would perform better than the other?

Probably the third axis, as it represents the relationship between label and chord in a more meaningful way – EXPAND ON THIS

If using 2d representation of labels/chords -> what is the use of convolutions? Can they actually find patterns -> research this

**Generator**

Takes in a label and transforms it into a fake chord candidate

Again, do we need to use convolutional layers? Or will deep/fully connected layers do the job

Real examples = (label + real chord) + [1] (label for discriminator)

Fake examples = (label + fake chord) + [0] (label for discriminator)

**Training**

**GAN Loss Function:**

the generator tries to minimize the following function while the discriminator tries to maximize it:

* D(x) is the discriminator's estimate of the probability that real data instance x is real.
* Ex is the expected value over all real data instances.
* G(z) is the generator's output when given noise z.
* D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
* Ez is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
* The formula derives from the [cross-entropy](https://developers.google.com/machine-learning/glossary#cross-entropy) between the real and generated distributions.

(<https://developers.google.com/machine-learning/gan/loss>)

**Conditional GAN Loss Function**

**Backpropagation**

**Generator**

**Results:**

**Data Embedding**

I experimented with the data embedding:

**1)** 2 different 1D representations:

**1.** Source chord, real chord = (12,)(88,)

**2.** Source chord, real chord = (88,)(88,) [source chord(label) embedded to full 88 note vector in the middle C octave

**2)** a 2D representation:

(88,)(88,) 1D embeddings reduced to (84,)(84,) by removing bottom 3 notes and top note. Then transformed to (7,12)(7,12)

**Classification Models**

In order to determine the best embedding for the GAN, as well as the best type of network for the discriminator and generator, performed some classification tasks using all the embeddings and a combination of RNN and CNN.

1D RNN classifier using both **1.1 and 1.2** (above)

2D CNN classifier using **2**

Results:

Both performed incredibly well:

Give functions, data, graphs, etc

**Model 1: 2d pix2pix model (adapted for my dataset):**

Results showed that the discriminators loss quickly went to near zero – meaning that the generator was not generating convincing chords from source chords.

Show graph of loss over time and any other graphs/chards that were in 1D paper or other papers

Show some example generated chords

Why?

Unclear, using convolutional layers on simple binary chord matrices potentially didn’t yield any meaningful patterns

patchGAN uses a segment of the real/fake images (chords) passed in. ie. If matrices is 128x128, it looks at a 16x16 segment or multiple segments

this approach wont be as effective for chord data, as there would be notes spread around the vector, particularly on the x-axis (0 axis??)

Also mention loss functions /learning rate etc

**Model 2: 1D pix2pix model (adapted for my 1D dataset):**

Results?

Same as above

**Model 3: 1D Novel Model**

This model was born out of some initial testing using novel classifier networks

**V1**

The discriminator had a single output neuron with sigmoid activation with binary-crossentropy loss function

Results…. Loss of the discriminator on fake chords quickly reduced to 0, meaning generator was not improving

\*mention the loss function of the GAN here\* - explain it etc

\*more results here with graphs etc and chords shown in images etc with stats\*

**V2**

Created an 88 neuron output layer on the discriminator

\* mention how this changes the way the loss function works in the c\_gan model

Discriminator loss on real chords went from 1.5 – 1.2 over 11 thousand iterations of training

Loss on fake chords quickly went to zero

**Changed way source chord and real/fake chord concatenated**

Initially it was (88,)(88,) 🡪 (2, 88)

Maybe dense layers don’t perform as well on matrices?? Reference/research

Changed to (88,)(88,)

**Leadsheet XML Arranger**

**INPUT:** leadsheet (musicxml)

Step 1. Extract chord symbols

Step 2. Embed chord symbols as 88 “source-chord” note vectors (see preprocessing chapter)

Step 3. Pass source chord vectors into c\_gan model

Step 4. Combine leadsheet melody with generated chords into full arrangement

**OUTPUT:** Chords + Melody Arrangement

Data parsed from lead sheets

**Meta information:**

-Key

-Time signature

**Chords:**

-Chord symbol

-Bar number

-Position in bar

-Note at chord position (won’t account for 2 voicings / not sure which note it will get if chord present)

Data Issues

The number of different types of chords in the data was skewed :

Insert pandas table

1000 dominant 7 wheras much less of other chords

Data skewness

Data should be bell curve

But currently it is right skewed

Something to do with mode median

SOLUTION?

Get more data + remove some label occurences

WHEN SPLITTING TEST vs TRAIN:

INITIAL DATA COUNTS WITHOUT WIDENING CHORD SELECTIONS:

dominant 1177

minor-seventh 1053

major-seventh 674

major 453

dominant-ninth 255

suspended-fourth 249

dominant-13th 159

minor-ninth 133

half-diminished 129

minor 117

major-sixth 96

major-ninth 68

minor-11th 63

diminished-seventh 59

diminished 44

minor-sixth 38

Name: label, dtype: int64

Performing data analysis using pandas and matlabplot

Removed all chord labels that have less than 600 occurrences

Created a stacked bar chart representation of each label to analyse data and note occurrences

\*\*insert bar charts\*\*

For the large part the data was good - \*\*insert\*\* margin of error numbers

However a small number of labels had wrong notes associated with them

\*\*insert\*\* some examples with numbers of wrong notes

Therefore further data cleaning was performed. Created array of unwanted notes for each label, and iteratively removed those notes from each chord vector.

\*\* insert cleaned bar charts\*\*

We now have

References

Ji, S., Luo, J. and Yang, X., 2020. A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions. arXiv preprint arXiv:2011.06801.

Hadjeres, G., Pachet, F. and Nielsen, F., 2017, July. Deepbach: a steerable model for bach chorales generation. In International Conference on Machine Learning (pp. 1362-1371). PMLR.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. Advances in neural information processing systems, 27.