Honors Thesis Proposal

For

Musical form reconstruction in printed and handwritten lead sheets via optical recognition of chord symbols

Nashir Janmohamed

[illegible]

Gita Sukthankar, Ph.D.

Thesis Committee Chair

Department of Computer Science

[illegible]

Manoj Chopra, Ph.D., P.E.

HUT Liaison

Department of Computer Science

[illegible]

Chen Chen, Ph.D.

Committee Member

Department of Computer Science

[illegible]

Clarence Penn, Professor

Committee Member

School of Performing Arts

TABLE OF CONTENTS

LIST OF FIGURES iii

CHAPTER 1: INTRODUCTION 1

MAIN IDEA/E2E EXAMPLE 3

CHAPTER 2: RELATED WORK 4

PRIOR WORK IN OPTICAL MUSIC RECOGNITION 4

OTHER RELATED WORK 7

MUSIC 8

JAZZ FORM 9

CHAPTER 3: METHODOLOGY 11

PRE-PROCESSING 11

CHORD SYMBOL CANDIDATE LOCALIZATION 13

CHORD SYMBOL CANDIDATES CLASSIFICATION 13

CHORD SYMBOL PRIMITIVE CLASSIFICATION 15

CHORD SYMBOL RECONSTRUCTION FROM PRIMITIVES 16

A NOTE ON CHALLENGING CASES 17

DATASET OF CHORD SYMBOL PRIMITIVES 18

CHORD PROGRESSION RECONSTRUCTION 20

ENHANCED JAZZ STANDARDS DATASET 20

CHAPTER 4: TENTATIVE TIMETABLE FOR THESIS COMPLETION 21

COMPLETED 21

IN PROGRESS 21

PENDING 21

LIST OF REFERENCES 24

APPENDICES 29

LIST OF FIGURES

1.1 Overview of proposed method applied to a lead sheet. 3

2.1 Example of chord symbol shorthand. 8

3.1 Architecture diagram for our method. 11

3.2 Splitting a chord symbol into primitives: G, h, 7 14

3.3 Various chord symbol representations that all correspond to Cmaj7. 14

3.4 C7#5 = $\{$C, E, G#, B♭$\}$. C7♭13 = $\{$C, E, A♭, B♭$\}$, where G# $\equiv$ A♭. 17

3.5 Dominant chord with extensions listed non-sequentially. 17

3.6 Slash and polychords with a horizontal line separating root and bass note/chord. 18

3.7 Chords with touching primitives. 18

3.8 Pipeline for collecting handwritten chord symbol primitives data. 19

CHAPTER 1: INTRODUCTION

Chord symbols, a musical shorthand that outlines both musical harmony and song structure, are arguably the most important descriptor of musical information in lead sheets. Despite their relevance to the musical document processing task, few works have explored identification of chord symbols using computer vision.

Chapter 2 contains a brief background on music concepts, state of the art in chord symbol classification, and an overview of research areas relevant to this work.

This research aims to identify printed and handwritten chord symbols in musical documents using traditional [21] and deep learning [16] methods and use the identified chord symbols to reconstruct the musical structure. This work generally falls under the umbrella of *Optical Music Recognition* (OMR), the area of computationally reading music notation in documents [7].

Our approach is composed of four stages (with our contributions/innovations in bold font):

1. Document preprocessing/segmentation.
2. Chord symbol candidate **localization**.
3. Chord symbol candidate classification (printed and **handwritten**).
4. **Chord progression reconstruction**.

In addition to the proposed approach for chord symbol identification/form analysis, a key contribution of this work is the introduction of two datasets:

1. Labeled chord symbol primitives, printed and handwritten (see Figure 3.2).

2. Enhanced version of jazz standard dataset [37] that includes song form type, number of bars, and other musical metadata.

We outline the methodology and describe the data sets in more detail in Chapter 3.

To our knowledge, only two works [9, 23] have addressed the problem of identifying chord symbols in musical documents. **We build on their work in chord symbol identification by 1) identifying handwritten symbols and 2) localizing chord symbols on the page rather than simply classifying them**.

Furthermore, while there has been research in chord symbol identification, **we believe we will be the first to use localized chord symbols to reconstruct a representation of musical structure**.

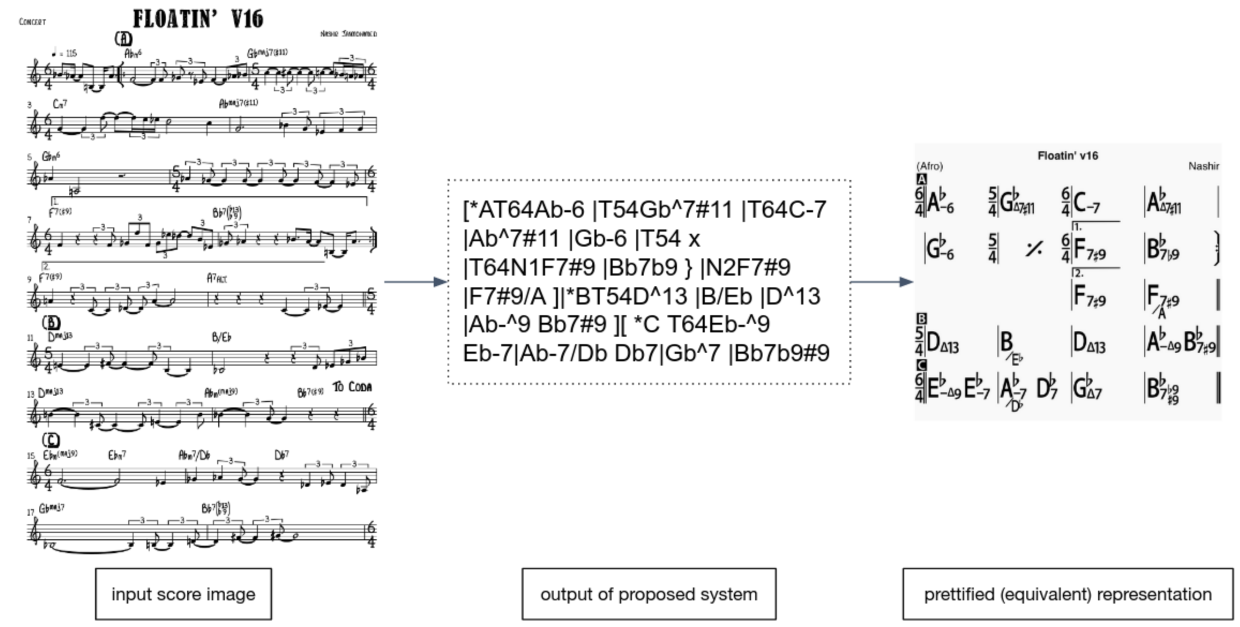
A variety of applications can benefit from the proposed OMR pipeline:

- Use images of lead sheets as context for real-time music generation (e.g., chordal, melodic, rhythmic) within the musical structure defined by sheet music.
- Perform large-scale automated harmonic analysis of handwritten musical documents.
- Convert OMR output into representation compatible with third-party software that generates backing tracks (e.g., iReal Pro) - musicians could then use the backing tracks to make their practicing more efficient/comprehensive, or share their compositions with others.

Lastly, at the time of this writing, Audiveris (a leading open source OMR tool) cannot recognize chord symbols; there is a clear need for a solution to this problem.

MAIN IDEA/E2E EXAMPLE

We show an example of our proposed method's end-to-end input and output below. The left side of the figure shows the input, a lead sheet with chord symbols. The middle of the figure shows our output representation, which uses the iReal format [4]. Using third-party open-source tools, we can easily translate this format to other music representations (e.g., MIDI [14], MusicXML [19]). The right side of the figure shows the iReal string from the middle of the figure rendered using the iReal pro app.



```
[*AT64Ab-6 |T54Gb^7#11 |T64C-7
|Ab^7#11 |Gb-6 |T54 x
|T64N1F7#9 |Bb7b9 } |N2F7#9
|F7#9/A ]|*BT54D^13 |B/Eb |D^13
|Ab-^9 Bb7#9 ][ *C T64Eb-^9
Eb-7|Ab-7/Db Db7|Gb^7 |Bb7b9#9
```

FLOATIN' V16
Concert
Floatin' v16
(Afro)
Nashir
input score image
output of proposed system
prettified (equivalent) representation

Figure 1.1: Overview of proposed method applied to a lead sheet.

CHAPTER 2: RELATED WORK

First, we discuss prior work in Optical Music Recognition (OMR) that identifies chord symbols in musical documents and their shortcomings. Then, we give a targeted overview of techniques used in our work. Finally, we discuss germane musical concepts.

PRIOR WORK IN OPTICAL MUSIC RECOGNITION

Many works focus on music object detection (e.g., notes, clefs, staff lines) [1, 7, 29], which is of evident importance; automatic digitization enables automated search, analysis, and preservation of musical scores. However, chord symbols are common in post-19th-century popular music lead sheets, and recognizing these is an essential task often neglected in the literature.

Kodirov et al. [23] were the first to address the challenge of identifying chord symbols in musical documents. Their approach consists of three stages: 1) preprocessing, 2) separation, and 3) recognition.

1. Preprocessing (input - image, output - chord blobs):
   (a) Binarize input image [28].
   (b) Detect and remove staff lines [8].
   (c) Identify and blob connected components.
   (d) Extract only blobs that include chords.
   (e) Rebinarize each chord individually; reduces distortion.
2. Separation (input - chord blobs; output - individual primitives):

    (a) Use rule-based approach that assumes Times New Roman or Arial font to extract each primitive.

3. Recognition (input - individual primitives, output - labels):

    (a) Compute the smallest area matrix that encloses the primitive.

    (b) Compute feature vector from pixels using zoning: chord sample divided into $m \times n$ zones; each zone takes the percentage of black pixels (foreground pixels) as the feature; compute a single feature vector per primitive by concatenating the value of each zone.

    (c) Compute label using k-nearest neighbor classifier using Euclidean distance between feature vectors.

Kodirov et al.'s approach is limited to recognizing chords of the form $\{$(root) (alteration) (m $\cup$ 7)$\}$.

Dinh et al. [9] extended Kodirov's work by 1) considering special characters such as '(', ')', and '/', 2) rejecting outliers (such as lyric text or musical symbols), and 3) recognizing chord symbols with a greater number of symbols (e.g., DmM^7/C$\sharp$). Overall, their approach is similar - they describe two stages: 1) chord symbol region detection and 2) chord symbol recognition.

1. Chord symbol region detection:

    (a) Preprocessing: binarization, un-distortion [35], stave removal[33], lyric removal [6, 11]. After preprocessing, only chord symbols and some outliers remain above the stave area.

    (b) Chord symbol candidate detection: identify each chord symbol candidate by scanning each staff from left to right and grouping candidates based on their horizontal and vertical proximity within a line.

2. Chord symbol recognition:

(a) For each chord symbol candidate, extract the candidate to a new sub-image and iterate over the components in the candidate.

   i. Vertical slicing: find valleys in the histogram of each connected component, splitting symbols at the valleys of the histogram.

   ii. Group validation: validates each component by counting the number of sub-components in each segmented component; if there is more than one component after vertical slicing, filter out the component from the next steps.

   iii. Up-scaling small groups: Scale components with an area smaller than 9×7.

   iv. Sub-lexicon extraction: uses prefix matching to create a sub-lexicon that excludes chord symbols whose prefix does not match (e.g., for symbol B7 and lexicon $\{$A♭, B7, B♯$\}$, after identifying B, the sub-lexicon consists of $\{$7, ♯$\}$).

   v. Outlier detection: compute a dissimilarity score based on comparing each component to a stored template with the same label; if the score is above a threshold, reject the chord symbol.

Dinh et al.'s approach is more robust than Kodirov et al.'s. However, it is still limited to symbols with a certain font. Also, Dinh et al.'s vertical slicing method can break valid components into invalid sub-components, leading to false negative rejection of chord symbols.

For the task of chord identification, there are no suitable open-source datasets. Kodirov et al. released their dataset [22], but it is small (12-20 examples per class in the training set), has a limited alphabet (A, B, C, D, E, F, G, ♯, ♭, 7), and uses a single printed font. Dinh et al. (ref 2) describe using two datasets of printed Korean sheet music that contain more complex chord symbols (e.g., F♯m7♭5, DmM^7/C♯), but did not release either dataset.

There is a need for a dataset of handwritten chord symbols with a more extensive set of chord classes (e.g., 7alt., 7♭9♯5, Δ7, polychords) and for a dataset of both printed and handwritten chord symbols. Such a dataset would enable OMR in a broader variety of lead sheets (printed and handwritten). Not only does this have practical benefits for musicians and composers, but it also enables future work in Music Information Retrieval [10] and contextual music generation from lead sheets (i.e. jazz melody/chordal improvisation).

Introducing such a dataset is a part of the work proposed for this thesis.

Our approach builds on [9, 23] by

1. Handling a larger lexicon of possible chord symbols by using a grammar-based approach.
2. Classifying printed and handwritten symbols.
3. Using chord location information to reconstruct the musical form.

OTHER RELATED WORK

Harte et al. [20] propose a method for the symbolic representation of musical chords and syntax of chord notation in Backus-Naur form. This method requires conversion from the typical format found on lead sheets from shorthand to a set of intervals above the root and does not allow for poly-chord constructs.

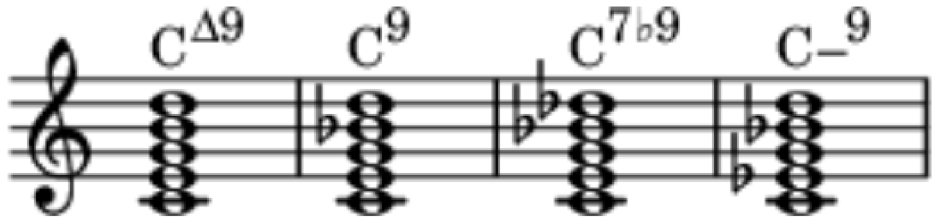
Tiehuis [34] introduces a Parsing Expression Grammar (PEG) [12] for recognizing music chord symbols which we modify and use in our chord classification algorithm.

In addition to collecting and labeling samples of chord symbol primitives, we plan to explore data augmentation techniques [26, 31]. We also intend to use GANs [17] and possibly diffusion [30] for synthetic data generation.

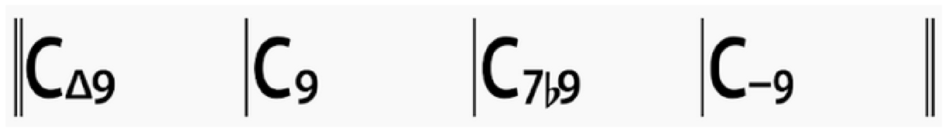
We will also explore various techniques for deep learning with small datasets [2, 5, 27], as well as the use of curriculum learning [3, 18, 32, 36] to improve the generalizability of our chord symbol primitive classifier to out-of-distribution symbols.

MUSIC

A *chord* is a set of pitches/frequencies consisting of multiple notes heard as if sounding simultaneously (though they may not all have the same onset).



(a) Sample chord progression with all notes specified.



(b) Sample chord progression with chord symbols specified.

Figure 2.1: Example of chord symbol shorthand.

A chord progression or harmonic progression is a succession of chords. The sample progression shown in 2.1 is an example of a chord progression.

The notation in 2.1b implies the same choice of notes as displayed explicitly in 2.1a. This shorthand is a prominent feature of lead sheets, a music document often used in jazz and other popular music forms that gives a bare-bones outline of the harmony and melody of a song. See the leftmost image in Figure 1.1 for an example.

JAZZ FORM

Jazz standards are a pre-set, agreed-upon progression of chords with an associated melody superimposed on the chord progression.

When performed, jazz standards often have a head-solo-head structure:

- Head: fully specified melody, harmony, and rhythm (one at the beginning and one at the end).
- Solo section: no specified melody, mostly-specified harmony (the soloist or accompaniment may choose to alter/modify certain aspects of the harmony), no specified rhythm - up to the discretion of the soloist and accompaniment.

The head and the solo section occur over a repeated chord progression called the *form*. The form of the melody dictates the structure for the rest of the song (with some exceptions, e.g., vamps, tags, intros, and outros). There are exceptions where the solo section is over a different chord progression than the melody, but this is uncommon.

Four recurring forms occur frequently in jazz: 1) 32 bar AABA, 2) 32 bar AB (aka ABAC), 3) 12 bar (usually a blues form), and 16 bar. However, many “jazz standards” do not fit neatly into these categories, and compositions from post-1950 often avoid this convention. One thing to note is that, even for two charts with the same class of song form (e.g., 32 bar AABA), they may have different visual layouts in their respective lead sheets, which poses a challenge in the document processing task.

CHAPTER 3: METHODOLOGY

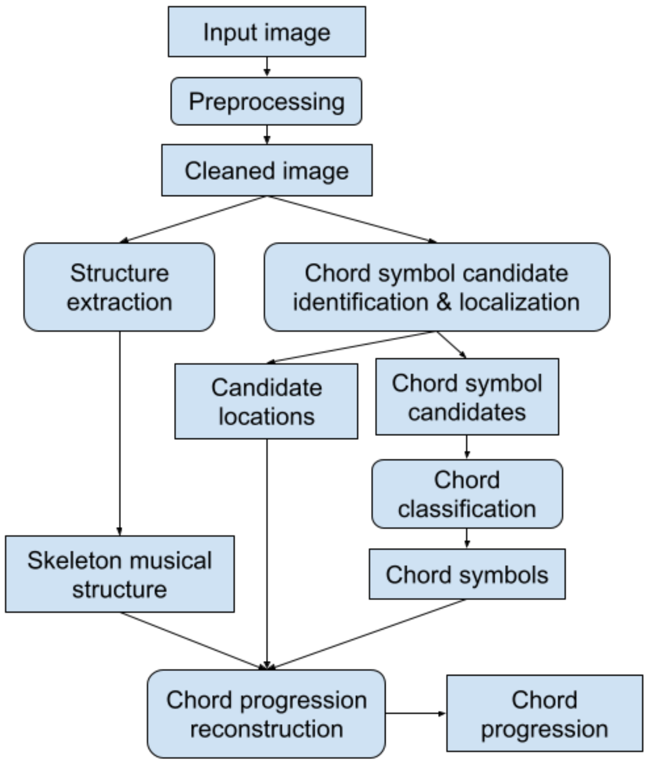


Figure 3.1: Architecture diagram for our method.

PRE-PROCESSING

We plan to follow Dinh et al.'s [9] pre-processing method with one extra step:

1. Image binarization.

2. Un-distortion.

3. Staff removal.

4. **Bar detection**.

5. Lyric removal.

In 4), we extract the location of each bar and use this to build a sequential list of bars as they appear in the musical document. Each `Bar` contains metadata that we use in the chord progression reconstruction step:

```
class Bar:
  chords: ChordSymbol[]
  ts: TimeSignature
  has_start_repeat: bool
  has_end_repeat: bool
  has_start_double_bar: bool
  has_end_double_bar: bool
  has_coda_symbol: bool
  has_segno_symbol: bool
  has_dc_symbol: bool

class ChordSymbol:
  img: 2DImage
  text_label: str
  beat_number: int
  x: int
  y: int
  h: int
  w: int

class TimeSignature:
  numerator: int
  denominator: int # must be of the form 2^i, i >= 0
```

We obtain a list of `Bar` objects, `bars`, s.t. `bars[i]` corresponds to measure `i`. Initially, each bar has an empty list of chords. This modification allows us to localize each chord symbol to a bar (and the corresponding beat within the bar) in the document.

CHORD SYMBOL CANDIDATE LOCALIZATION

We then identify all chord symbol candidates and anchor each candidate to the nearest bar such that the bar is under the chord symbol in the document.

```
class ChordSymbolCandidate:
  img: 2DImage
  measure_num: int
  beat_number: int
```

At this point, candidates may include outliers such as lyrics or other musical symbols, which we handle during the classification stage.

CHORD SYMBOL CANDIDATES CLASSIFICATION

Our chord symbol classification algorithm consists of the following steps:

1. Split a chord symbol candidate into a set of *primitives* (i.e. connected components).
2. Classify/reject each primitive.
3. Use a grammar [20, 34] to reconstruct the identified primitives into a chord symbol, rejecting invalid chord symbols.

[1]This symbol indicates a half-diminished seventh chord, which we label as h.

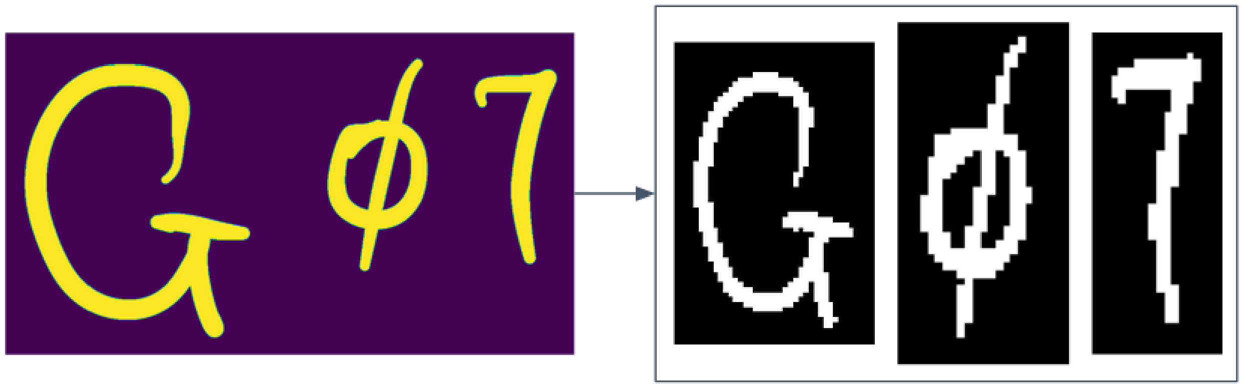


Figure 3.2: Splitting a chord symbol into primitives: G, h^1, 7.

This piecewise approach seeks to handle the challenging cases of non-sequential symbols, touching components, and multiple distinct combinations of components corresponding to the same chord symbol label (see Figure 3.3).

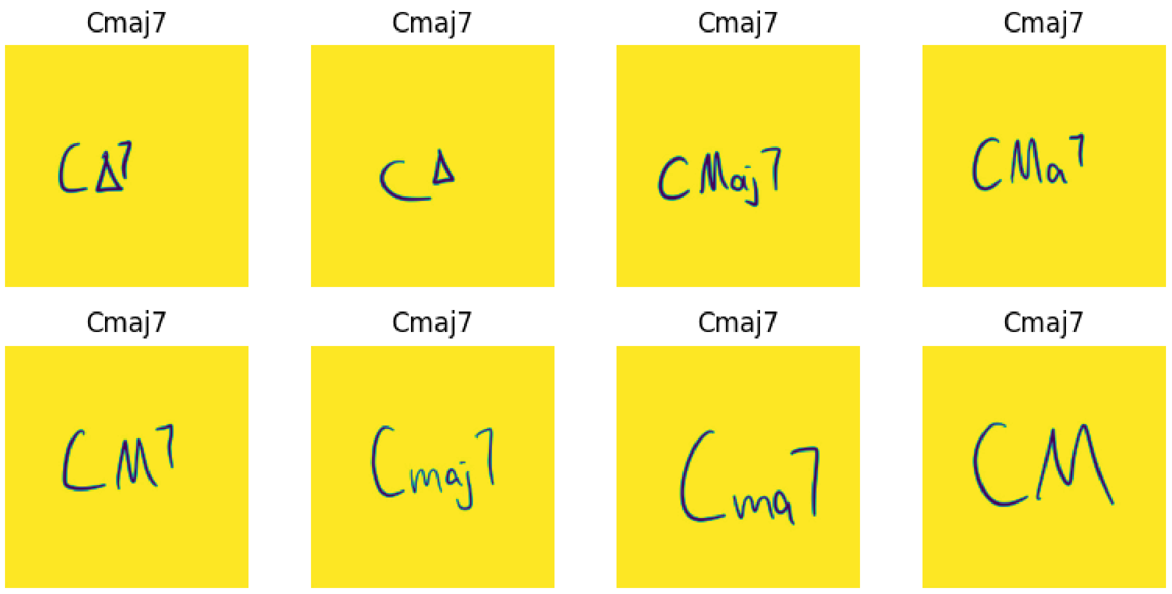


Figure 3.3: Various chord symbol representations that all correspond to Cmaj7.

After the chord symbol is classified (and only if it is not rejected, as in the case of an invalid candidate, e.g., lyric text, coda symbol), we use the stored `measure_num` and `beat_num` to populate `bars[i]` with the identified chord symbol.

CHORD SYMBOL PRIMITIVE CLASSIFICATION

It may be hard for a model to learn many rules implicitly for an entire chord symbol candidate (see Figure 3.3). Instead, we classify each character in the candidate, build words from these characters, then recognize each word as a chord quality according to a grammar-based approach. This approach is tractable since the number of possible chord symbols we can encounter on a lead sheet is finite.

Note that if we reject a chord symbol candidate, it might be due to touching components. To address touching, we will first try using erosion (a morphological operator) and then explore approaches for touching character segmentation (e.g., [15]) if needed.

Ideally, we would have a single classifier that performs well on both printed and handwritten sheet music; we will first focus on printed chord symbols and see if the method can produce good accuracy. Only after validating the method do we transition to handwritten chord symbols and evaluate the performance. The classifier will be trained and evaluated on a data set of primitives[2].

There may be a domain gap for a classifier trained on printed primitives when evaluating it on handwritten primitives. We can try using approaches to minimize domain gap [24] to address this. Also, due to limited dataset size, we will try pretraining/finetuning an existing model on our data set of primitives.

We will also evaluate different training approaches:

[2]As mentioned above, part of this work involves curating (and releasing) a data set of primitives

1. Pretrain on printed, then finetune on handwritten.
2. Train on both printed and handwritten primitives simultaneously.
3. Progressive training, i.e., curriculum learning:
   (a) Printed primitives.
   (b) Style transfer.
   (c) Real handwritten.

and evaluate how these strategies affect the model's performance on out-of-distribution data.

CHORD SYMBOL RECONSTRUCTION FROM PRIMITIVES

We plan to use a modified version of Tiehuis' parsing expression grammar [34] (see Appendix 4) to recognize chord symbols. The current version has some limitations - it cannot recognize 7sus4 chord (among others). Furthermore, we have to flatten non-sequential chord extensions (see Figure 3.5) before feeding them into this grammar, which expects a sequential list of characters.

If this grammar does not recognize a sequence, we reject that sequence.

We may fail to recognize touching primitives in the previous step, in which case we would reject many valid chords at this point. In that case, we will consider a more robust way to separate chord candidates into primitives.

A NOTE ON CHALLENGING CASES

- Many possible chord qualities for a given root note (see Appendix 4 for an extensive list of chord qualities commonly found on jazz lead sheets).
- Equivalent chord symbols but distinct notation (see Figure 3.4), where the symbols imply the same choice of notes but are notated differently. The difference in notation should be preserved.

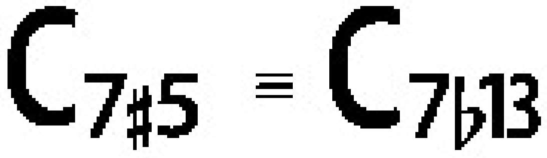


Figure 3.4: C7#5 = {C, E, G#, B♭}. C7♭13 = {C, E, A♭, B♭}, where G# ≡ A♭.

- Chord symbol with non-sequential extension format.

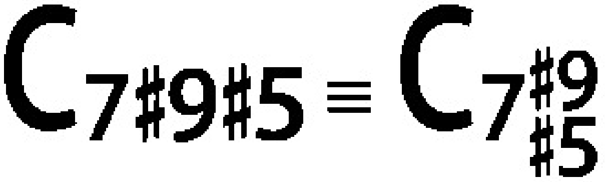


Figure 3.5: Dominant chord with extensions listed non-sequentially.

- Vertically separated slash/polychords (see Figure 3.6).
- Touching primitives in chord symbol (see Figure 3.7) - chords with touching primitives require additional preprocessing before they can be separated into disjoint components and classified.

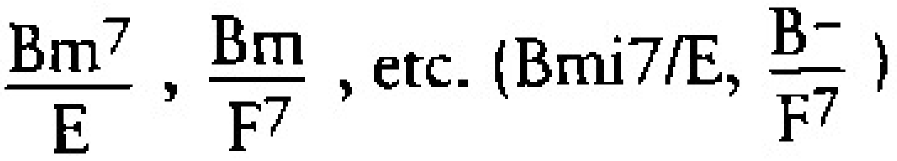


Figure 3.6: Slash and polychords with a horizontal line separating root and bass note/chord.

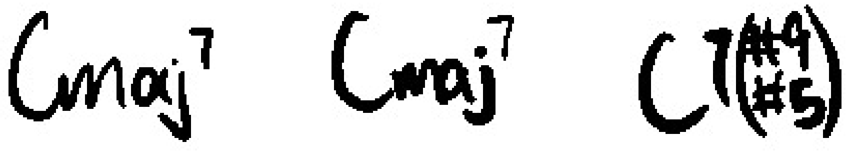


Figure 3.7: Chords with touching primitives.

It is not straightforward to directly compare our method to those proposed in [23] and [9], especially since ours is a superset of theirs. However, we can compare results on chord classification to Kodirov - since they release their dataset, we can evaluate our classifier on their test set; we should be able to identify those with high accuracy. Since Dinh does not publish their method or data, we cannot make a direct comparison.

DATASET OF CHORD SYMBOL PRIMITIVES

After curating an exhaustive list of primitives used for notating chord symbols (e.g., {A, ♯, -}, we will collect a large number of labeled images of each primitive. We plan to curate a dataset of handwritten symbols from multiple people for a diversity of samples, then apply additional data augmentation techniques to this set of handwritten symbols to increase the size of the dataset.

The handwritten training data will be collected as shown in Figure 3.8. Each page will consist of a single symbol type; by labeling just one page, we label 80 examples, which significantly reduces the labeling cost.

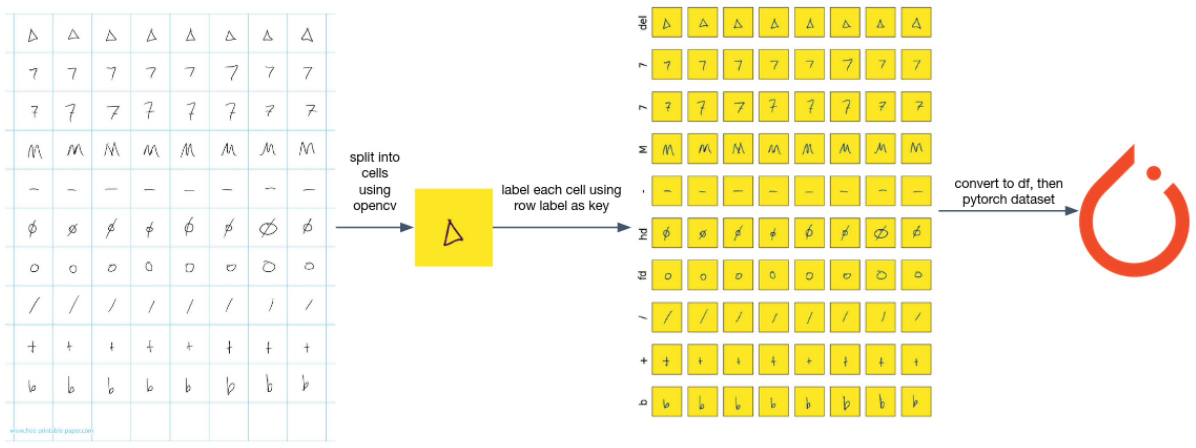

split into cells using opencv
label each cell using row label as key
convert to df, then pytorch dataset
del
7
7
M
-
hd
fd
/
+
b

Figure 3.8: Pipeline for collecting handwritten chord symbol primitives data.

We will then use data augmentation techniques [13, 26, 31]) to increase the size of the dataset, hopefully allowing for a deep learning approach to build a robust and generalizable classifier.

We also plan to explore data transformation and style transfer using GANs [17] or diffusion [30], creating models that generate synthetic symbols to augment the data.

We also plan to curate data from a selection of lead sheets in order to have a large set of type-faced evaluation examples with the following approach:

1. Use chord candidate identification algorithm to identify candidates.

2. Manually reject all non-chord symbol candidates.

3. Split candidates into primitives.

4. Cluster primitives (using k = number of primitives we choose to identify).

5. Hand-label some valid candidates in each cluster, propagating labels to others (e.g., pseudo/semi-supervised labeling).

CHORD PROGRESSION RECONSTRUCTION

After processing all chord symbol candidates, we convert the constructed `Bar`s to an iReal format [4] representation of the form.

We can initially evaluate the results of chord progression reconstruction using a string similarity metric, such as Levenshtein distance [25], comparing the output string of the processing pipeline to the ground truth iReal string. We can then construct other evaluation metrics as needed.

ENHANCED JAZZ STANDARDS DATASET

We plan to modify the existing Jazz Standards Dataset [37], adding music metadata information by parsing iReal song strings. This dataset will serve as our ground truth for form reconstruction.

We will create iReal strings (as needed, some lead sheets may already have corresponding iReal strings) for a curated set of lead sheets. Then, we will run our processing pipeline on the lead sheet, comparing the pipeline's output to the iReal ground truth.

Alternatively, we can convert the iReal strings to MusicXML format (.mxl), generate a dataset of scores from these .mxl files, then test the method on these scores.

CHAPTER 4: TENTATIVE TIMETABLE FOR THESIS COMPLETION

COMPLETED

- Acceptance to HUT program
- Select an advisor/get their buy-in
- Form committee
- Participate in HUT orientation
- Pick a research topic
- Initial literature review
- Write proposal

IN PROGRESS

- Get advisor/committee approval on the proposal
- Participate in at least one workshop
- Submit the proposal to HUT program
- Implement methodology

PENDING

- Apply to UCF Student Scholar Symposium

- Apply to Florida undergrad conference in Miami
- Prepare poster for UCF Student Scholar Symposium/Florida undergrad conference
- Participate in UCF Student Scholar Symposium poster session
- Establish a date for the thesis defense with the Thesis Chair and committee (at least six weeks prior to the last day of classes)
- Complete first draft of the thesis (at least six weeks prior to the defense)
- Submit a draft of the thesis for a format review to UCF's Showcase of Text, Archives, Research & Scholarship (STARS) system
- Submit a signed notice of defense through email to the Office of Honors Research at least one week prior to the date of defense
- Complete the scheduled thesis oral defense
- Incorporate the committee's recommendations into the thesis
- Make all formatting corrections suggested by the Thesis Editor from the format review
- Submit a copy of the thesis to Turnitin.com/iThenticate
- Complete research
- Finalize the thesis (at least two weeks prior to defense)
- Apply to conferences (e.g., ISMIR), submit a paper describing results
- Submit the signed Thesis Approval forms to the Office of Honors Research in person or via email by the published deadline
- Upload the final thesis to STARS (after submitting signed Thesis Approval forms)

- Complete the Honors Undergraduate Thesis exit survey, which will be available online toward the end of each semester
- Complete the Honors Undergraduate Thesis Intent to Graduate Form and submit it to the Office of Honors Research (must submit by February 15, 2023)

LIST OF REFERENCES

[1] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[2] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[4] Massimo Biolcati. ireal file format. Retrieved November 27, 2022 from `https://www.irealpro.com/ireal-pro-file-format`.

[5] Lorenzo Brigato and Luca Iocchi. A close look at deep learning with small data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 2490–2497. IEEE, 2021.

[6] John Ashley Burgoyne, Yue Ouyang, Tristan Himmelman, Johanna Devaney, Laurent Pugin, and Ichiro Fujinaga. Lyric extraction and recognition on digital images of early music sources. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, volume 10, pages 723–727. International Society for Music Information Retrieval Canada, 2009.

[7] Jorge Calvo-Zaragoza, Jan Hajič Jr, and Alexander Pacha. Understanding optical music recognition. *ACM Computing Surveys (CSUR)*, 53(4):1–35, 2020.

[8] Artur Capela, Ana Rebelo, Jaime S Cardoso, and Carlos Guedes. Staff line detection and removal with stable paths. 2008.

[9] Cong Minh Dinh, Luu Ngoc Do, Hyung-Jeong Yang, Soo-Hyung Kim, and Guee-Sang Lee. Improved lexicon-driven based chord symbol recognition in musical images. *International Journal of Contents*, 12(4):53–61, 2016.

[10] J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

[11] Markus Feldbach and Klaus D Tonnies. Line detection and segmentation in historical church registers. In *Proceedings of sixth international conference on document analysis and recognition*, pages 743–747. IEEE, 2001.

[12] Bryan Ford. Parsing expression grammars: a recognition-based syntactic foundation. In *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 111–122, 2004.

[13] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós. 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):243–251, 2012.

[14] Ichiro Fujinaga. Standard midi-file format spec. 1.1, updated, N/A. Retrieved November 27, 2022 from `http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html`.

[15] Utpal Garain and BB Chaudhuri. Segmentation of touching symbols for ocr of printed mathematical expressions: an approach based on multifactorial analysis. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 177–181. IEEE, 2005.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[18] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[19] W3C Music Notation Community Group. Musicxml 4.0, 2021. Retrieved November 27, 2022 from `https://www.w3.org/2021/06/musicxml40/`.

[20] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, volume 5, pages 66–71, 2005.

[21] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[22] Elyor Kodirov. Omr chord separation and recognition 6306 database, version 1, 2014. Retrieved November 27, 2022 from `https://sites.google.com/site/elyorkodirovresearch/omr-chsr6306`.

[23] Elyor Kodirov, Sejin Han, Guee-Sang Lee, and YoungChul Kim. Music with harmony: chord separation and recognition in printed music score images. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, pages 1–8, 2014.

[24] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10285–10295, 2019.

[25] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[26] Norman Mu and Justin Gilmer. Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

[27] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, pages 443–449, 2015.

[28] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[29] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes, and Jaime S Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[31] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[32] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, pages 1–40, 2022.

[33] Bolan Su, Shijian Lu, Umapada Pal, and Chew Lim Tan. An effective staff detection and removal technique for musical documents. In *2012 10th IAPR International Workshop on Document Analysis Systems*, pages 160–164. IEEE, 2012.

[34] Marc Tiehuis. Peg grammar for chord notation, 2017. Retrieved November 28, 2022 from `https://tiehu.is/blog/peg1`.

[35] Quang Nhat Vo, Tam Nguyen, Soo-Hyung Kim, Hyung-Jeong Yang, and Guee-Sang Lee. Distorted music score recognition without staffline removal. In *2014 22nd International Conference on Pattern Recognition*, pages 2956–2960. IEEE, 2014.

[36] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[37] Melih Can Yardı. Jazz standards dataset, version 1, 2021. Retrieved November 27, 2022 from `https://www.kaggle.com/datasets/melihcanyardi/jazz-standards/versions/1`.

APPENDICES

ACKNOWLEDGEMENTS

MY FAMILY

For supporting my pursuit of a second degree in computer science while encouraging me to continue exploring my passion for music.

MY COMMITTEE

Dr. Sukthankar for overseeing my thesis and providing invaluable feedback. Dr. Chen for his mentorship since I started at UCF and for encouraging me to find a way to combine my passion for music and computer science. Professor Penn for continually giving me new insights on how to approach music, life, and work. Dr. Chopra for facilitating my thesis.

BURNETT HONORS COLLEGE; COLLEGE OF ENGINEERING AND CS

For 1) the opportunity to work on this thesis and 2) for awarding a scholarship to support my work.

CHORD SYMBOL PEG GRAMMAR

Table 1: Parsing Expression Grammar for recognizing chord symbols

PolyChord	=	Chord (s Chord)?
Chord	=	Chord1 / Chord2 / Chord3 / Chord4
Chord1	=	Note Special ChordUpper
Chord2	=	Note ThirdSeventh Extended? ChordUpper
Chord3	=	Note Third? Sixth ChordUpper
Chord4	=	Note Third? Extended? ChordUpper
ChordUpper	=	Addition? Alterations? Slash?
ThirdSeventh	=	Augmented / Diminished
Augmented	=	’aug’ / ’+’
Diminished	=	’dim’ / ’o’ / ’ø’
Special	=	’5’ / ’sus2’ / ’sus4’
Sixth	=	’6’ / ’6/9’
Third	=	(MajorThird !Extended) / MinorThird
MajorThird	=	’Maj’ / ’M’ / ’Δ’
MinorThird	=	’min’ / ’m’ / ’-’
Extended	=	ExtendedQuality? ExtendedInterval
ExtendedQuality	=	MajorThird / ’dom’
ExtendedInterval	=	’7’ / ’9’ / ’11’ / ’13’
Alterations	=	Alteration / AlterationList
AlterationList	=	’(’ Alteration (’,’ Alteration)* ’)’
Alteration	=	Accidental AlterationInterval
AlterationInterval	=	’4’ / ’5’ / ’9’ / ’11’ / ’13’
Addition	=	’add’ (’2’ / ’4’ / ’6’)
Slash	=	’/’ Note Accidental?
Note	=	[A-G] Accidental*
Accidental	=	[b#]

Above copied from [34] with permission - this parsing expression grammar categorizes the most common cases for standard Jazz notation.

Rules for PEG grammar

Note that s corresponds to / or —, the separator between the primary and secondary chord, or between the chord and its bass note.

Examples: Cmaj7/B, $\frac{\text{Bdim7}}{\text{Adim7}}$

Note also that this grammar allows for polychords such as E♭m / A^7, a shortcoming of [20].

NON-COMPREHENSIVE LIST OF CHORD SYMBOLS

A non-comprehensive list of possible chord qualities with C as the root note is as follows (note that any of these can have an arbitrary bass note or an arbitrary bass chord to construct a polychord such as Cm7/F♯7):

Triads

- C
  - Cmaj
  - CMa
- C-
  - Cm
  - Cmi
  - Cminor
- Cdim
  - Co
- Caug
  - C+
- Csus

Major 7th/9th/11th/13th chords

- CM7
  - Cmaj7
  - CMa7
  - C∆
  - C∆7
- Cmaj7#5
  - Cmaj7b6
- Cmaj#4
- Cmaj9
- Cmaj13
- Cmaj9#11

Seventh chords (minor)

- Cm/ma7
  - C-maj7
- Cm6
  - C-6
- CmM9
  - -Maj

- Cm9
- Cm13
- Cm11

Seventh chords (diminished)

- Cm7b5
- Cdim7

Seventh chords (dominant-esque)

- C7
- C7sus4
- C7#11
- C7#9
- C9sus4
- C7sus2
- C9sus2
- C9
- C13
- Calt7

- C7b9
- C7b5
- C7#5
- C9b5
- C9#5
- C#9b5
- Cb9#5
- C7b13
- C7#5

Others

- C5
- C6
  - Cmaj6
  - CM6
- Csus2
- C6/9