COMPOSER CLASSIFICATION WITH CROSS-MODAL TRANSFER LEARNING AND MUSICALLY-INFORMED AUGMENTATION

Daniel Yang   **TJ Tsai**

Harvey Mudd College

`dhyang,ttsai@hmc.edu`

ABSTRACT

This paper studies composer style classification of piano sheet music, MIDI, and audio data. We expand upon previous work in three ways. First, we explore several musically motivated data augmentation schemes based on pitch-shifting and random removal of individual notes or groups of notes. We show that these augmentation schemes lead to dramatic improvements in model performance, of a magnitude that exceeds the benefit of pretraining on all solo piano sheet music images in IMSLP. Second, we describe a way to modify previous models in order to enable cross-model transfer learning, in which a model trained entirely on sheet music can be used to perform composer classification of audio or MIDI data. Third, we explore the performance of trained models in a 1-shot learning context, in which the model performs classification among a set of composers that are unseen in training. Our results indicate that models learn a representation of style that generalizes beyond the set of composers used in training.

1. INTRODUCTION

This paper studies composer style classification based on sheet music, audio, and MIDI data. Given a previously unseen page of sheet music or fragment of audio/MIDI, the goal is to predict which one of a fixed set of C composers composed it based on its compositional style.

Previous works on composer classification generally fall into one of three categories. The first category of approaches extract manually designed features from the data and feed them to a classifier. Some features that have been explored include chroma [1, 2], expert musicological features [3–5], musical intervals or counterpoint characteristics [6, 7], piece-level statistics or features describing piece structure [2, 8], and pre-defined feature sets like the jSymbolic toolbox [9, 10]. Many standard classification algorithms have been used, such as decision trees (e.g. [7, 9]), KNN (e.g. [11]), logistic regression (e.g. [5]), SVMs (e.g. [3, 9]), and neural networks (e.g. [2, 12]). The second category of approaches train one sequence-based model for each composer and then select the model that has

the highest likelihood for a given query sequence. These sequence-based models include n-gram language models [8, 13–15] and several variants of Markov models [16, 17]. These models are typically fed with a very low-level representation of the data, such as sequences of note values or intervals between consecutive notes. The third category of approaches train a neural network classifier in an end-to-end fashion. Rather than relying on manually designed features, this approach tries to automatically learn a suitable feature representation from the raw data that is effective for classification. This paradigm was explored early on in [18], and many recent works have focused on convolutional neural networks trained on piano roll-like data [19–21].

The above works generally assume that the data is available in a symbolic format such as MIDI, MusicXML, or **kern. Recent works [22, 23] have explored the composer classification task based on raw sheet music images. While this makes the problem more challenging due to the high-dimensional nature of images, it also provides a very distinct advantage: there is an enormous amount of sheet music data available through the International Music Score Library Project (IMSLP). [1] These works first convert each sheet music image into a sequence of musical words based on the bootleg score feature representation [24], and then treat the problem as one of text classification. They incorporate best practices from natural language processing, such as pretraining a language model on a large set of unlabeled data, and then finetuning a classification model based on a small set of labeled data.

This paper expands upon [22] in three different directions. [2] First, we explore several different forms of data augmentation for the bootleg score representation. These include pitch shifting of noteheads relative to the staff lines and several different forms of dropout that are musically motivated. Second, we explore cross-modal transfer learning, in which a model is first trained entirely on sheet music, and then used for composer classification of audio and MIDI data. Third, we explore the performance of trained models in a 1-shot learning context, in which the goal is to perform classification among a set of unseen composers given only one representative piece from each composer.

This paper has three main contributions, which correspond directly with the three directions above.

---

[1] `https://imslp.org`

[2] Code can be found at `https://github.com/HMC-MIR/ComposerID`.

- We propose several forms of data augmentation for the bootleg score representation and evaluate their impact on the composer classification task. We demonstrate enormous performance improvements (37.3% to 63.9% accuracy) which are even larger than the benefit of pretraining on all IMSLP piano sheet music (37.3% to 57.5%). Both can be combined to achieve even greater gains (89.0%). We share our optimal settings and selection of augmentation methods, which may be useful for tasks using symbolic data or any piano roll-like representation.

- We successfully demonstrate cross-modal transfer learning. By making a minor modification to the bootleg score representation, we show that it is possible to train a model on sheet music and use it for composer classification of audio and MIDI data. This approach may benefit tasks where the amount of data in one modality is limited or restricted (e.g. due to copyright issues).

- We evaluate the performance of trained models in a 1-shot learning context, in which we use the model's penultimate layer activations as a feature embedding. Our results strongly indicate that the models are learning a more generalizable notion of compositional style that extends beyond the composers in training.

We describe each of the three new directions in detail in the next three sections.

2. DATA AUGMENTATION METHODS

This section explores several data augmentation schemes for our task. The next five subsections describe the feature representation (Section 2.1), proposed augmentation strategies (Section 2.2), experimental setup (Section 2.3), experimental results (Section 2.4), and analyses (Section 2.5).

2.1 Feature Representation

We first describe how sheet music is converted into a sequence of words as proposed in [22]. This forms the backdrop for our proposed augmentation strategies. This process consists of two steps.

The first step is to compute a bootleg score representation of the sheet music image. The bootleg score is a mid-level feature representation that encodes the position of filled noteheads relative to the staff lines in piano sheet music [24]. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of distinct staff line positions in both the left and right hand staves and where N indicates the number of grouped note events (e.g. a chord containing four notes played simultaneously would constitute a single grouped note event). This representation discards a significant amount of information in the sheet music, such as non-filled noteheads, time signature, key signature, accidentals, note duration, measure boundaries, octave markings, and clef changes.

Nonetheless, it has been shown to be useful for a variety of tasks involving sheet music such as sheet-MIDI passage retrieval [24], audio-sheet music synchronization [25], sheet music identification [26], and finding matches between the Lakh MIDI dataset and IMSLP [27].

The second step is to tokenize the bootleg score. This is done in two different ways, depending on if the language model architecture is word-based or subword-based. For word-based models (e.g. AWD-LSTM [28]), each column of the bootleg score (62 bits) is represented as a single 64-bit integer and interpreted as a word. For subword-based models (e.g. GPT-2 [29], RoBERTa [30]), each column is represented as a sequence of four 8-bit characters so that the bootleg score can be expressed as a length $4N$ sequence of 8-bit characters. Based on a training set of character sequences, a byte pair encoder (BPE) [31] can be trained in an unsupervised fashion to learn a vocabulary of common subwords. The trained BPE can then be used to tokenize the length $4N$ sequence of characters into a sequence of subwords. The result of the tokenization step is a sequence of words or subwords that are fed to the language model.

2.2 Proposed Augmentation Strategies

We explore two different types of data augmentation for the bootleg score representation. These strategies could be applied to symbolic data or piano roll representations as well.

The first type of data augmentation is based on pitch shifting noteheads in the bootleg score. Similar to other musically informed representations like chroma and CQT, shifts in the bootleg score correspond to transpositions in key. We consider pitch shifts up to $\pm K$ positions, where K is a hyperparameter. When a pitch-shifted notehead falls off the edge of the bootleg score (i.e. the top or bottom of the left hand or right hand staff), the notehead is simply removed. Note that a value of K will result in $2K + 1$ times as much data as the original dataset. We consider pitch-shifting at both training time as well as at test time. For the latter, we pitch shift a query bootleg score by up to $\pm K$ positions, pass all $2K + 1$ bootleg scores through the trained model, and average the predictions to generate a single ensemble prediction.

The second type of data augmentation is based on removing noteheads in the bootleg score. This strategy is based on the simple observation that randomly adding a note to a column of the bootleg score is unlikely to yield a musically plausible event, whereas randomly *removing* one or more notes from a column of the bootleg score *is* likely to yield a musically plausible event. For example, consider a column in the bootleg score that contains octaves in the left hand and a chord in the right hand. If a single note or even an entire hand is removed, the result is still a musically plausible event. We consider three different types of removal: randomly dropping each individual notehead with some probability, randomly dropping an entire hand (i.e. all noteheads in the left or right hand staff) within a single bootleg score column, or randomly drop-

ping an entire column of the bootleg score. Since these methods correspond to applying a form of dropout regularization directly to the bootleg score representation, we refer to these three types of removal as DropNote, DropHand, and DropColumn regularization, respectively.

2.3 Experimental Setup

We use the same experimental setup as [22] for fair comparison. A brief summary is provided below for completeness.

The unlabeled data for language model pretraining consists of all solo piano sheet music images in IMSLP. It contains 29,310 PDFs, 255,539 pages, and 48.5 million bootleg score features. We used the precomputed bootleg score features provided in [26]. We will refer to this unlabeled dataset as the IMSLP data. During language model pretraining, 90% of the data is used for training and 10% for validation.

The labeled dataset for the composer classification task is a carefully curated subset of the IMSLP data. It contains one representative sheet music version from every solo piano piece in IMSLP from nine different classical music composers: Bach, Beethoven, Chopin, Haydn, Liszt, Mozart, Schubert, Schumann, and Scriabin. These PDFs were manually filtered to remove filler pages like title page, foreword, etc. The resulting set contains 787 PDFs, 7,151 pages, and 1.47 million bootleg score features. The labeled data is split by piece into training (4347 pages), validation (1500 pages), and test (1304 pages) sets.

The labeled dataset was further preprocessed to form a fragment dataset in order to solve two problems: too little data (only 4347 training images) and significant class imbalance. A fixed number of bootleg score fragments of length 64 were randomly sampled from each composer. The resulting fragment dataset contains 32400, 10800, and 10800 fragments for training, validation, and test, respectively.

We report classification results on both the fragment classification task and the full page classification task. The fragment dataset is used for training all models, since it has more training samples and is class-balanced. When evaluating a model on the full page task, fragments of length 64 are taken from the query bootleg score with 50% overlap, all fragments are passed through the fragment classification model, and the predictions are averaged to form a single prediction for the entire page. We report results in terms of classification accuracy for the fragment classification task and macro F1 for the full page classification task (since the classes are imbalanced with pages).

2.4 Results

Figure 1 compares the performance of several classification models on the fragment composer classification task (left) and full page classification task (right). The results without data augmentation are shown as black horizontal lines, and the results with optimal data augmentation settings (described in Section 2.5) are shown as colored bars. Note that the results without data augmentation correspond
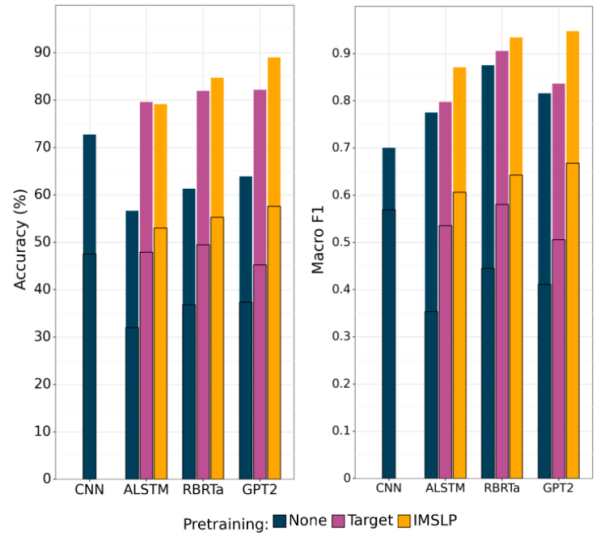

Figure 1. Results for sheet music composer classification of fragments (left) and full pages (right). The colored bars show performance with optimal data augmentation settings, and the horizontal black lines show performance without any data augmentation.

to the results reported in [22]. Results are reported for four different classification models and across three different pretraining conditions. The four model architectures are a CNN model (based on [19]) that has two convolutional layers followed by global pooling across the time dimension and a final output layer, AWD-LSTM [28], RoBERTa [30], and GPT-2 [29]. The three pretraining conditions are: (a) no pretraining, in which models are trained from scratch only on the labeled fragment dataset, (b) target pretraining, in which language models are pretrained on the labeled data and then finetuned for the classification task, and (c) IMSLP pretraining, in which we pretrain the language models on the IMSLP data, finetune the language model on the labeled data, and then finetune the classifier on the labeled fragment dataset.

There are a few things to notice about Figure 1. Across all models and all pretraining conditions, there is an extremely large benefit to using data augmentation. In all cases, the benefit of data augmentation is larger than the benefit of pretraining. For example, for the RoBERTa model, data augmentation improves performance on the full page classification task from 0.44 to 0.88 macro F1 (without any pretraining), while pretraining on IMSLP improves performance from 0.44 to 0.64 (without any data augmentation). When both data augmentation and pretraining are combined, the benefit is enormous: the RoBERTa model increases from 36.8% accuracy to 84.7% and from 0.44 macro F1 to 0.93.

2.5 Analysis

Figure 2 shows the benefit of adding a single type of training data augmentation in isolation. The results on the fragment classification task on the validation set are shown at left, and the results on the full page classifica-
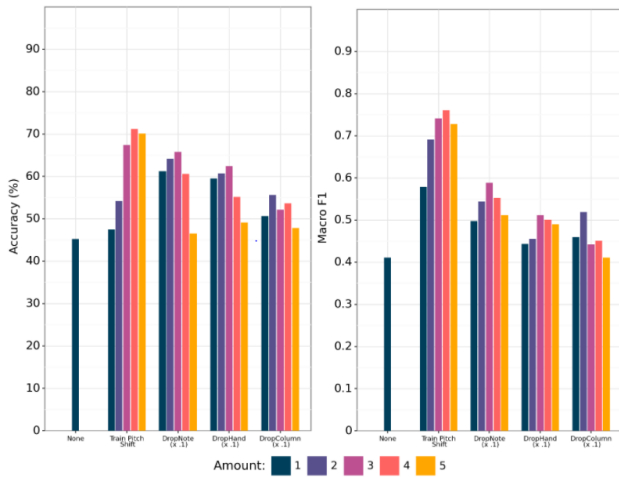
Figure 2. Effect of adding a single type of data augmentation in isolation to the GPT-2 model. Individual bars within each group show the effect of different hyperparameter settings. The leftmost standalone bar shows the performance without data augmentation for comparison.

Figure 3. Effect of adding each type of data augmentation cumulatively to the GPT-2 model. Each group uses the optimal settings from previous augmentation types.

tion task are shown at right. Within each figure, the leftmost (standalone) bar shows the performance of the GPT-2 model without data augmentation for reference. The four groups of bars correspond to four different types of training data augmentation: pitch shifting, DropNote, DropHand, and DropColumn. Within each group, individual bars show the performance with different hyperparameters settings (e.g. $K = 1, 2, 3, 4, 5$ for pitch shifting and $p = 0.1, 0.2, 0.3, 0.4, 0.5$ for the dropout variants). We can see that pitch shifting seems to be the most effective form of data augmentation, followed by DropNote, DropHand, and then DropColumn. The optimal amount of pitch shifting seems to be $K = 4$, above which the results get slightly worse. Most likely this is because large pitch shifts result in many noteheads simply being removed from the bootleg score canvas.

Figure 3 shows the benefit of adding various forms of data augmentation cumulatively. Again the leftmost (standalone) bar shows the performance without data augmentation. The first (leftmost) group of bars shows the performance with only training pitch shift augmentation. These results are identical to those shown in Figure 2. The second group of bars shows the performance with training pitch shift augmentation ($K = 4$) and DropNote with various values of p. Each successive group adds an additional form of augmentation with the optimal settings of previous augmentation types. This figure allows us to see the cumulative benefit of adding multiple forms of data augmentation. We see dramatic improvements from adding the most effective forms of augmentation, and modest but nontrivial improvements after that. The optimal settings are training pitch shifting with $K = 4$, DropNote with $p = 0.3$, DropHand with $p = 0.3$, DropColumn with $p = 0.3$ and test-time pitch shifting with $K = 4$. These are the settings used in the results shown in Figure 1.
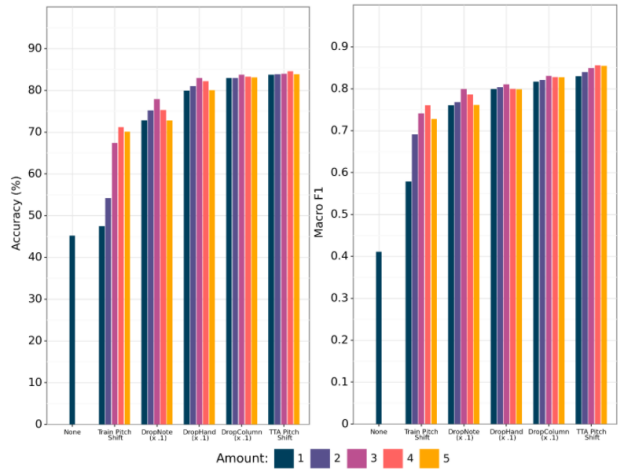
3. CROSS-MODAL TRANSFER LEARNING

This section describes our exploration into cross-modal transfer learning, in which a model trained entirely on sheet music is used to perform composer classification of audio and MIDI data. In the next three subsections, we describe the methodology (Section 3.1), experimental setup (Section 3.2), and experimental results (Section 3.3).

3.1 Methodology

The key to cross-modal transfer learning is representing audio, MIDI, and sheet music in a common feature space. That feature space is a modified bootleg score representation. Below, we describe a way to bridge the gap between MIDI and sheet music using this modified bootleg score representation. For audio performances of piano music, we first apply an automatic music transcription (AMT) system [32], and then follow the procedure below.

We can extract a bootleg score from MIDI by mapping MIDI note onset events to staff line positions in sheet music using the conventions of Western musical notation. However, there are two obstacles that prevent a MIDI-generated bootleg score and a sheet music-generated bootleg score from being directly comparable. First, there is ambiguity about left/right hand attribution. For example, if a C4 note onset occurs in a MIDI file, it could appear in the left hand staff (one ledger line above the topmost staff line) or the right hand staff (one ledger line below the bottom staff line). In the sheet music, it will only appear in one of these locations. Second, there is ambiguity about enharmonic representations. For example, a MIDI note number 61 could appear in the sheet music as a C-sharp or a D-flat, and these correspond to two different staff line positions.

These two ambiguities can be resolved in different ways. To handle the ambiguity due to left/right hand attribution, we can simply place noteheads in the middle register in *both* the left hand and right hand staves. For example, if the sheet music contains a notehead one ledger line

below the right hand staff (i.e. C4 in treble clef), an additional notehead will be placed one ledger line above the left hand staff (i.e. C4 in bass clef). Likewise, a MIDI note onset at C4 will result in two noteheads at the same two locations in the MIDI-generated bootleg score. By making this modification to the bootleg score representation, the MIDI-generated bootleg score and sheet music-generated bootleg score will match. To handle the ambiguity due to enharmonic representations, we can generate two different versions of the MIDI bootleg score: one in which all black keys on the piano are interpreted as sharps, and one in which all black keys on the piano are interpreted as flats. We can then pass both versions of the MIDI bootleg score through our classification model and average the resulting predictions. This method for bridging the gap between MIDI and sheet music was first proposed in [27] for a MIDI-sheet retrieval task. Here, we use the same technique for cross-modal transfer learning in composer classification.

Cross-modal transfer learning thus requires two changes to the system described in Section 2. The first change is to use the modified bootleg score representation when converting sheet music to a sequence of words. The models are otherwise trained exactly as before. The second change is to consider both sharp and flat versions of the MIDI-generated bootleg score during inference, and to average the model's predictions from both.

3.2 Experimental Setup

In order to assess the effectiveness of cross-modal transfer learning, we need to introduce additional datasets of MIDI and audio for the composer classification task. These datasets are derived from the MAESTRO dataset [33], which contains MIDI and audio files of real piano performances. We preprocess the dataset in the following manner. First, we take all MIDI performances of pieces composed by the same nine composers in our labeled sheet music dataset. Because some pieces are performed many times, we take one representative performance for each piece to avoid overemphasizing a small set of popular pieces. Second, we randomly sample $X = 5000$ fragments of length Y seconds from each composer, spread evenly across the composer's pieces. This sampling strategy produces a dataset of fragments that is class-balanced, and it allows us to study the effect of fragment length on model accuracy. Because we are not doing any additional training or finetuning, we use all of the MIDI data as a test set. This constitutes our MIDI fragment composer classification dataset. A corresponding audio fragment composer classification dataset can be generated using the same methodology.

Note that the above dataset contains many pieces that were in the labeled sheet music dataset, albeit in a different modality. To further study the generalizability of our trained models, we constructed two different versions of the audio/MIDI fragment classification datasets: one in which all of the data is present, and another in which pieces that were in the labeled sheet music training data
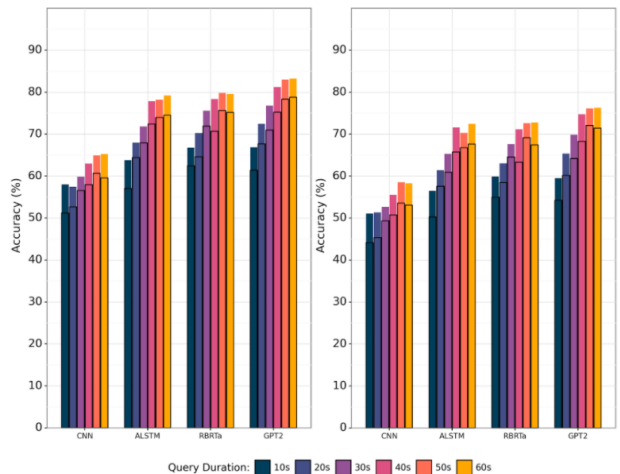

Figure 4. Results for composer classification of MIDI (colored bars) and audio (horizontal black bars) on the full (left plot) and reduced (right plot) datasets. The results are with a GPT-2 model trained only on sheet music.

are excluded. Because the latter has many fewer pieces, we reduce the number of fragments per composer to $X = 500$. The number of pieces for each composer in the full/reduced datasets are: Bach – 101/27, Beethoven 87/21, Chopin – 101/31, Haydn – 29/13, Liszt – 108/25, Mozart – 36/12, Schubert – 103/33, Schumann – 32/8, Scriabin – 25/1. In total, there are 622 pieces in the full dataset and 171 pieces in the reduced dataset.

3.3 Results

Figure 4 compares the performance of four different models on the MIDI and audio fragment classification tasks. The left plot shows performance on the dataset containing all pieces, and the right plot shows performance on the dataset that excludes pieces in the sheet music training dataset. The performance on the MIDI fragment classification task is shown by colored bars, and the performance on the audio fragment classification task is indicated with horizontal black lines. The four models shown are the best version of each model architecture from Figure 1: the CNN, AWD-LSTM, RoBERTa, and GPT-2 models with optimal data augmentation settings and IMSLP pretraining (for the language models). The four groups of bars in each plot correspond to the four different models. Within each group, the individual bars show the model performance for different durations of the audio/MIDI query. Similar to the full page sheet music classification task, we convert each audio/MIDI query to a bootleg score, take fragments of length 64 with 50% overlap, pass each fragment through the classification model, and average the predictions from all fragments. When processing audio queries, the Onsets & Frames AMT system [32] is used to convert the audio to an estimated MIDI representation.

There are four things to notice about Figure 4. First, the results do clearly demonstrate effective cross-modal transfer learning. Because the datasets are balanced by class,

random guessing would correspond to an accuracy of 11%. In contrast, the GPT-2 model is predicting the correct composer of 60-second MIDI fragments 83% and 76% of the time on the two versions of the MIDI classification data. This shows that we can train a model on sheet music data, and then use it to classify MIDI and audio data without any additional training or finetuning. Second, there is a 6-8% difference in accuracy between the two versions of datasets (i.e. comparing the left plot to the right plot). This reflects the benefit of having seen a piece before in training in a different modality. But even when a piece has never been seen before – in any modality – the results in the righmost plot show that the models still perform well. Third, there is a 4-6% difference in accuracy between the MIDI classification task and the audio classification task (i.e. comparing colored bars to the black horizontal lines). This gap comes from failures in the AMT system when converting from audio to MIDI. Fourth, the query duration strongly affects model performance for shorter queries, but plateaus at a duration of about 50 seconds (i.e. comparing individual bars within each group). This suggests that 40-50 seconds is enough context to recognize the style of a piece, and that using more context beyond that is unlikely to help much.

4. ONE-SHOT LEARNING

This section describes our exploration of using trained models in a 1-shot learning context, in which the model is expected to classify pieces from a set of C unseen composers given only a single representative piece from each composer. In the next three subsections, we describe our methodology (Section 4.1), experimental setup (Section 4.2), and experimental results (Section 4.3).

4.1 Methodology

For 1-shot learning, we use our trained classification models as a feature extractor that projects sheet music into an embedding space that captures some notion of compositional style. We take the penultimate layer activations of the model as our feature representation. When processing a MIDI or audio performance, we first compute the bootleg score (using AMT to convert the audio to MIDI, if necessary), take multiple bootleg score fragments of length 64 with 50% overlap, and then extract the embedding features from each fragment. On training data, each fragment's embedding serves as a single sample in our KNN database. For a given test bootleg score fragment, we find the $K = 3$ closest samples in Euclidean distance from each composer, and then rank composers by their average KNN distance.

4.2 Experimental Setup

Our data for 1-shot classification experiments also comes from the MAESTRO dataset. We exclude the original nine composers used in training and identify nine other composers with sufficient data: Rachmaninoff, Debussy, Scarlatti, Mendelssohn, Brahms, Mussorgsky, Tchaikovsky, Clementi, and Handel. We sample five pieces from each

Model	MIDI		Audio	
	Acc	Std	Acc	Std
Random	16.6%	2.2%	13.9%	2.0%
CNN	39.1%	2.1%	38.2%	3.7%
AWD-LSTM	45.5%	2.2%	42.8%	4.2%
RoBERTa	50.3%	3.1%	49.1%	3.4%
GPT-2	52.8%	3.9%	52.7%	3.1%

Table 1. Results for 1-shot learning experiments on composer classification of MIDI (left) and audio (right) fragments. The trained models are used to classify among $C = 9$ unseen composers given a single representative piece from each composer.

composer to ensure equal representation. For each episode, we randomly select one of the five pieces from each composer to serve as our training data, and use the remaining data for testing. Each test query is a single bootleg score fragment of length 64 extracted from one of the $9 \times 4 = 36$ test pieces (with 50% overlap between fragments). For each episode, we calculate the classification accuracy across the test queries. We ran 30 episodes with different train/test splits, and report the mean and standard deviation of the classification accuracy across the episodes.

4.3 Results

Table 1 shows the results of our 1-shot classification experiments. We report results with the best versions of each of the four model architectures. We also include the performance of a random guessing baseline as reference.

There are three things to notice about these results. First, all models perform much better than the random guessing baseline. This strongly indicates that the models are learning a more generalizable notion of compositional style that goes beyond the original nine composers in the training data. Second, we see the same relative ordering of performance as before: GPT-2 performs best, followed by RoBERTa, AWD-LSTM, and the CNN model. This suggests that better results on the original composer classification task are indicative of a more useful representation in the style embedding space. Third, we again observe a consistent difference in performance between 1-shot MIDI and 1-shot audio classification due to AMT errors.

5. CONCLUSION

This paper expands upon composer classification models in three ways. First, we propose several forms of data augmentation that lead to dramatic improvements in model performance. Second, we show that it is possible to modify previous models in order to enable cross-modal transfer learning, in which a model trained entirely on sheet music is used to perform composer classification on audio and MIDI data. Third, we show that trained models are effective in a 1-shot learning context, indicating that the models learn a representation of style that generalizes beyond the original composers used in training.

7. REFERENCES

[1] Y. Anan, K. Hatano, H. Bannai, M. Takeda, and K. Satoh, "Polyphonic music classification on symbolic data using dissimilarity functions," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 229–234.

[2] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Musical composer identification through probabilistic and feedforward neural networks," in *European Conference on the Applications of Evolutionary Computation*, 2010, pp. 411–420.

[3] W. Herlands, R. Der, Y. Greenberg, and S. Levin, "A machine learning approach to musically meaningful homogeneous style classification," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[4] E. Backer and P. van Kranenburg, "On musical stylometry–a pattern recognition approach," *Pattern Recognition Letters*, vol. 26, no. 3, pp. 299–309, 2005.

[5] K. C. Kempfert and S. W. Wong, "Where does haydn end and mozart begin? composer classification of string quartets," *arXiv preprint arXiv:1809.05075*, 2018.

[6] L. Mearns, D. Tidhar, and S. Dixon, "Characterisation of composer style using high-level musical features," in *Proc. of the 3rd International Workshop on Machine Learning and Music*, 2010, pp. 37–40.

[7] P. Van Kranenburg and E. Backer, "Musical style recognition—a quantitative approach," in *Handbook of Pattern Recognition and Computer Vision*, 2005, pp. 583–600.

[8] R. Hillewaere, B. Manderick, and D. Conklin, "String quartet classification with monophonic models," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 537–542.

[9] D. Herremans, D. Martens, and K. Sörensen, "Composer classification models for music-theory building," in *Computational Music Analysis*, 2016, pp. 369–392.

[10] C. McKay and I. Fujinaga, "jSymbolic: A feature extractor for midi files," in *Proc. of the International Computer Music Conference*, 2006.

[11] A. Brinkman, D. Shanahan, and C. Sapp, "Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin," in *Proc. of the Biennial International Conference on Music Perception and Cognition*, 2016, pp. 91–97.

[12] P. Sadeghian, C. Wilson, S. Goeddel, and A. Olmsted, "Classification of music by composer using fuzzy min-max neural networks," in *Proc. of the 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 189–192.

[13] M. Hontanilla, C. Pérez-Sancho, and J. M. Inesta, "Modeling musical style with language models for composer recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013, pp. 740–748.

[14] J. Wołkowicz and V. Kešelj, "Evaluation of n-gram-based classification approaches on classical music corpora," in *International Conference on Mathematics and Computation in Music*, 2013, pp. 213–225.

[15] J. Wołkowicz, Z. Kulka, and V. Kešelj, "N-gram-based approach to composer recognition," *Archives of Acoustics*, vol. 33, no. 1, pp. 43–55, 2008.

[16] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Weighted markov chain model for musical composer identification," in *European Conference on the Applications of Evolutionary Computation*, 2011, pp. 334–343.

[17] E. Pollastri and G. Simoncelli, "Classification of melodies by composer with hidden markov models," in *Proc. of the First International Conference on WEB Delivering of Music*, 2001, pp. 88–95.

[18] G. Buzzanca, "A supervised learning approach to musical style recognition," in *Proc. of the International Conference on Music and Artificial Intelligence (IC-MAI)*, vol. 2002, 2002, p. 167.

[19] H. Verma and J. Thickstun, "Convolutional composer classification," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 549–556.

[20] G. Velarde, C. C. Chacón, D. Meredith, T. Weyde, and M. Grachten, "Convolution-based classification of audio and symbolic representations of music," *Journal of New Music Research*, vol. 47, no. 3, pp. 191–205, 2018.

[21] G. Velarde, T. Weyde, C. E. C. Chacón, D. Meredith, and M. Grachten, "Composer recognition based on 2d-filtered piano-rolls," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 115–121.

[22] T. Tsai and K. Ji, "Composer style classification of piano sheet music images using language model pretraining," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 176–183.

[23] D. Yang, K. Ji, and T. Tsai, "A deeper look at sheet music composer classification using self-supervised pretraining," *Applied Sciences*, vol. 11, no. 4, p. 1387, 2021.

[24] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, "MIDI passage retrieval using cell phone pictures of sheet music," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 916–923.

[25] M. Shan and T. Tsai, "Improved handling of repeats and jumps in audio-sheet image synchronization," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 62–69.

[26] D. Yang and T. Tsai, "Camera-based piano sheet music identification," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 481–488.

[27] T. Tsai, "Towards linking the Lakh and IMSLP datasets," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 546–550.

[28] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *arXiv preprint arXiv:1708.02182*, 2017.

[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[31] P. Gage, "A new algorithm for data compression," *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[32] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 50–57.

[33] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019.