CONVOLUTIONAL COMPOSER CLASSIFICATION

Harsh Verma **John Thickstun**
Allen School of Computer Science and Engineering,
University of Washington, Seattle, WA, USA
harshv@cs.washington.edu, thickstn@cs.washington.edu

ABSTRACT

This paper investigates end-to-end learnable models for attributing composers to musical scores. We introduce several pooled, convolutional architectures for this task and draw connections between our approach and classical learning approaches based on global and n-gram features. We evaluate models on a corpus of 2,500 scores from the KernScores collection, authored by a variety of composers spanning the Renaissance era to the early 20th century. This corpus has substantial overlap with the corpora used in several previous, smaller studies; we compare our results on subsets of the corpus to these previous works.

1. INTRODUCTION

Models for attributing composers to musical scores have been extensively studied in the music information retrieval community. The composer classification question has been posed for a variety of corpora, from Renaissance composers [2,3], to the narrow (and challenging) case of Haydn and Mozart string quartets [5, 8, 12, 22], and to various collections of classical era composers (most of the other papers discussed in Section 2). In this work we study an expansive collection of scores, from 13th century sacred music by Guillaume Du Fay to 20th century ragtimes by Scott Joplin.

A major challenge of this task is learning from limited data. While the corpus considered here is larger than most, this is largely due to the number of composers considered (19): for specific composers, we have at most 466 scores (Bach) and as few as 22 (Japart). Small datasets are an inherent problem for composer classification: the corpus used in this work contains, for example, all of the Bach chorales and all of the Mozart string quartets. We cannot resurrect these composers and have them write us more scores to include in our corpus. This situation contrasts starkly with many learning problems, where substantial progress can be made by collecting massive datasets and exhaustively training an expressive model (usually a deep neural network) with “big data.”

Further complicating this task, an individual score is itself a high dimensional object: the average score in our corpus consists of thousands of notes, each of which is encoded as a high dimensional vector to represent its pitch and value. Learning from a small number of examples in a high dimensional space is a formidable problem; thus much work on composer classification focuses on feature engineering, feature selection, dimensionality reduction, or some combination of these approaches to construct low-dimensional representations of scores to learn from.

In this paper we take a different approach: we dispense with feature engineering and explore end-to-end classifiers that operate directly on full scores. Specifically, we investigate shallow convolutional neural networks with an aggressive pooling operation. In this setting, all but the most impoverished linear classifiers achieve 100% training accuracy. We rely on implicit regularization introduced by the network structure and first-order optimization with early stopping to avoid overfitting to training data. While theoretical understanding of such an approach is in its early stages [18], we find empirically that this works quite well for composer classification.

2. RELATED WORK

The earliest works on composer classification [3, 15] analyzed highly preprocessed corpora of melodic fragments. Much of the subsequent work on classification focuses on engineering features to summarize full scores. These approaches can be broadly categorized, using the terminology of [7], into “global” summarization approaches that compute small sets of summary statistics as a feature set for each score [2, 5, 6, 10, 12, 16, 22] and local “event” featurizations that extract n-gram counts of a score as features [8, 9, 11, 24, 25]. There is also a line of work that applies compression-based dissimilarity metrics [1, 19, 20] to this task, which offers a substantially different perspective on classification problems.

The present work is most similar in spirit to [3] and [23]. Like [3], we adopt an end-to-end approach to feature learning using neural architectures. In contrast with [3], we learn on full scores with minimal preprocessing and consider a multi-class classification task over a broad variety of composers; this approach is made possible by modern hardware unavailable to researchers in 2002. We also take a more systematic approach to architecture exploration, and identify effective architectures that are simpler than hybrid convolutional-recurrent approach taken in [3].

Like [23], we exploit structure in musical scores using convolutional models. But where [23] use a fixed Morlet or Gaussian convolution filters, the convolutional filters in this work are parameterized and learned from the data to maximize classification accuracy. We also explore multi-layer "deep" convolutional models and demonstrate improvements using such architectures versus the single layer of convolutions explored in [23].

Comparing to the substantial body of work that emphasize feature-engineering, the present work can be seen as an unified framework for learning global and event features. We will draw analogies between linear convolutional filters and n-gram features, and also demonstrate how convolutional models can express many popular global features. We will also introduce a global pooling operation that can be interpreted as an counter that tracks the number of occurrences of learned features, which is directly analogous to the count and ratio statistics that comprise the bulk of metrics used in human-engineered featurizations.

3. CORPUS AND DATA REPRESENTATION

We train and evaluate models on a corpus of 2,500 scores spanning five centuries of choral, piano, and chamber compositions from the KernScores collection [17]. An overview of this collection is provided in Table 1. In this work, we consider each movement of a multi-movement composition to be a distinct score. Our models extract only the note data (pitch, note-value, and voicing) from scores, ignoring all other markings such as time signatures, key signatures, tempo markings, instrumentation, and movement names. For the Renaissance composers in this collection (Du Fay through Japart) we shorten the length of all note-values by a factor of 4 to crudely account for the shift in duration conventions between mensural and modern notation [4].

We represent a score by lossless encoding of its pitch, voice, and note-value contents, transcoded from a **kern file to a binary representation suitable for input to a neural network. Specifically, we encode a score S as a binary tensor $\mathbf{x} \in \mathcal{S} = \{0,1\}^{T \times P \times (N+D+1)}$ where T, P, N, D are defined as follows:

- T - The number of rows of pitch/note-value data in the score S.
- P - The maximum number of concurrent **kern columns (spines): 6 for this corpus.
- N - The range of note pitches: 78 for this corpus, ranging from C1 to F#7.
- D - The number of distinct note values (i.e. durations): 55 for this corpus.

For each $t \in \{0, \ldots, T-1\}$, $p \in \{0, \ldots, P-1\}$, $n \in \{0, \ldots, N-1\}$, and $d \in \{0, \ldots, D-1\}$ we set

$$\begin{aligned} \mathbf{x}_{t,p,n} &= 1 && \text{iff pitch } n \text{ occurs at time } t \text{ in spine } p, \\ \mathbf{x}_{t,p,N+d} &= 1 && \text{iff note-value } d \text{ occurs at time } t \text{ in spine } p, \\ \mathbf{x}_{t,p,N+D} &= 1 && \text{iff pitch } n \text{ continues at time } t \text{ in spine } p. \end{aligned}$$

Composer	Dates	Sub-Collection	Scores
Du Fay	1397-1474	Choral	35
Ockeghem	1410-1497	Choral	98
Busnois	1430-1492	Choral	68
Martini	1440-1497	Choral	122
Compere	1445-1518	Choral	27
Josquin	1450-1521	Choral	423
de la Rue	1452-1518	Choral	178
Orto	1460-1529	Choral	43
Japart	1474-1507	Choral	22
Corelli	1653-1713	Trio Sonatas	188
Vivaldi	1678-1741	Concertos	33
Bach	1685-1750	Chorales	370
		Well-Tempered Clavier	96
D. Scarlatti	1685-1757	Keyboard Sonatas	59
Haydn	1732-1809	String Quartets	209
Mozart	1756-1791	Piano Sonatas	69
		String Quartets	82
Beethoven	1770-1827	Piano Sonatas	102
		String Quartets	67
Hummel	1778-1837	Preludes	24
Chopin	1810-1849	Preludes and Mazurkas	76
Joplin	1868-1917	Ragtimes	47

Table 1. Details of the KernScores collection used for training and evaluation in this paper.

For example, consider how we would encode line 28 of the **kern excerpt shown in Figure 1. This is the 5th row of pitch/note-value data in the score, so we will encode data from this line into $\mathbf{x}_5$. The periods "." in columns (i.e. spines) 0 and 1 indicate that a note or (in this case) a rest is continued from a previous line, so we set $\mathbf{x}_{5,0,N+D} = 1$ and $\mathbf{x}_{5,1,N+D} = 1$. Two notes are indicated in spine 2: 8a and 8f. The number "8" indicates an eighth-note value. Each unique note-value is assigned an (arbitrary) index; we assign index 15 to the eighth-note value, so we set $\mathbf{x}_{5,2,N+15} = 1$. The letters "a" and "f" indicate the pitches A3 and F4, which lie 33 and 41 semitones above C1 (the base of our note range) respectively. Therefore we set $\mathbf{x}_{5,2,33} = 1$ and $\mathbf{x}_{5,2,41} = 1$. Finally, spine 3 indicates an eighth-note F5 so we set $\mathbf{x}_{5,3,53} = 1$ and $\mathbf{x}_{5,3,N+15} = 1$.

The encoding defined about is an essentially verbatim transcoding of the **kern text data to a binary structured format. Converting from text to this structured format will allow us to write convolution operations along the time and pitch axes of the data tensor $\mathbf{x}$. Encoding pitches with binary indicators in N-dimensional vectors is consistent with piano roll representations [23] but departs from the example of [3], which encodes pitch as a single numerical magnitude. The binary pitch encoding is required to support convolutions along the pitch domain, which we will introduce later in models (8) and (9).

The binary note-value encoding also differs from the numerical magnitude encoding used in [3]. We will not introduce models that convolve over durations (there is no translation invariant structure to exploit) so the motivation above for representing pitches with indicators does not ap-

```
23:  2..r     2..r     2..r     2..r
24:  8r       8r       8r       8dd
25:  =1       =1       =1       =1
26:  1r       1r       8r       4dd
27:  .        .        8f# 8a   .
28:  .        .        8a 8f#   8ff#
29:  .        .        8a 8f#   16ee
30:  .        .        .        16dd
```

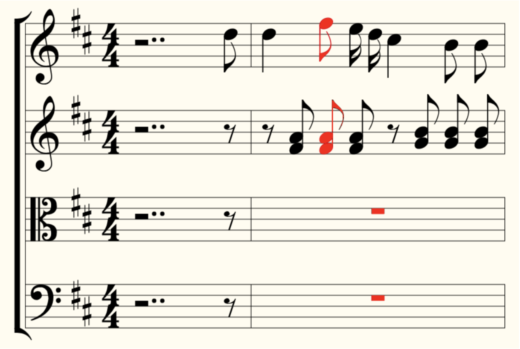

Figure 1. Left: An excerpt from a **kern encoding of of Haydn's Opus 33, No 1, consisting of the first 6 beats of the first movement. Right: A visual rendering of the first two measures of the same score as sheet music. In the **kern format, time proceeds from top to bottom, whereas in traditional notation time proceeds left to right. The contents of the beginning of the 6th beat is highlighted (red) in each format to aid comparison between the **kern format and sheet music.

ply to durations. Rather, we are motivated by the observation that note-values of similar duration may not be more alike in any musical sense than note-values with less similar duration. We avoid imposing this notion of similarity a-priori by encoding durations as categorical indicators: in this encoding, all note-values are equally distant in the Euclidean sense.

We also contrast our note-value encodings with piano roll representations, such as the representation used in [23]. In a piano roll representation, time is discretized: the value of a note is indicated implicitly by the number of discrete time-slices over which it is sustained. We choose an explicit representation of note-values because it more directly reflects the contents of a written score, and results in shorter time series overall than discretized representations.

4. PROBLEM FORMULATION

Our aim is to learn a classifier that predicts a composer y given a score $\mathbf{x}$. There are $C \equiv 19$ composers in our corpus: we assign each composer a label from 0 to $C-1$. We will construct a model $f_\theta : \mathcal{S} \to \{0,1\}^C$ that assigns vector $f_\theta(\mathbf{x})$ to a score $\mathbf{x}$ where each component $f_\theta(\mathbf{x})_i$ indicates model's (un-normalized) confidence that composer i wrote score $\mathbf{x}$. We predict $\hat{y}_\theta(\mathbf{x}) \equiv \arg\max_i f_\theta(\mathbf{x})_i$, the composer the model has most confidence in.

We evaluate our models via accuracy on holdout sets $\mathbf{x}^{\text{holdout}}$, where accuracy is the zero-one loss defined by

$$\text{Accuracy}(\mathbf{x}^{\text{holdout}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_\theta(\mathbf{x}_i^{\text{holdout}}) = y_i).$$

Here $\mathbf{1} : \text{Bool} \to \{0,1\}$ is the indicator function: $\mathbf{1}(p) = 1$ if the proposition p is true, otherwise $\mathbf{1}(p) = 0$. The results in Section 6 report 10-fold cross validated accuracies. It is standard practice in the machine learning community to report results on a single holdout set. But for for the small datasets considered in composer classification, cross-validating is essential to cut down the variance of estimated accuracy.

Given a collection of labeled scores (training data) $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and a parameterized family of models $\{f_\theta : \theta \in \Theta\}$ we learn an optimal model f_θ by empirical risk minimization of the negative log-likelihood under the softmax-normalized probability distribution of model outputs:

$$\min_{\theta\in\Theta} \sum_{i=1}^{n} -\log\left(\frac{\exp(f_\theta(\mathbf{x}_i)_{y_i})}{\sum_{k=1}^{C}\exp(f_\theta(\mathbf{x}_i)_{y_k})}\right). \tag{1}$$

For each of iteration of 10-fold cross-validation, in addition to the holdout fold $\mathbf{x}^{\text{holdout}}$, we hold out a second fold as validation data and optimize the objective (1) on the remaining 8 folds. We train our models using the Adam optimizer [13], regularizing with retrospective early stopping at the point with best accuracy on the validation fold.

5. MODELS

Every model class f_θ that we consider in this paper takes the following general approach.

1. Compute a set of local features at or around each time index in the score.

2. Average these features across time ("pool" the features, in neural networks parlance) into a single global feature vector.

3. Construct a linear classifier on top of this global feature vector to predict the composer of the score.

The approach above is motivated by the need to manage the high-dimensionality of a score: given even the first 5 indices of the score tensor $\mathbf{x}$ described in Section 3, we can easily fit a classifier that achieves 100% training accuracy but fails to generalize to new data. As discussed in Section 2, the classical approach to this overfitting phenomenon is to reduce a score to a low-dimensional summary of pre-determined features and fit a classifier to this summary. The present work aims to learn features from scratch, but if we permit our model to learn any features it wants then it will simply overfit to the training data.

We therefore cripple our models in two ways. First, step 1 of the general approach above limits our model to learn features that are local in scope. We will allow our models to learn features that are sensitive to correlations between co-occurring notes (harmony), or between short sequences of notes (melody, rhythm). But by construction, our models will not be able to learn features that capture correlations between (for example) the first and last notes of a score. This precludes us from learning certain high-level patterns that could have predictive power (e.g. Mozart is more likely to repeat a section verbatim than Bach) but saves of us from learning a multitude of spurious patterns that appear to have predictive power on the training data but fail to generalize to new observations.

Second, step 2 of the general approach prevents our model from learning features that occur at fixed time locations. As discussed above, even knowing the first 5 indices of the score tensor is enough to easily identify every score in the corpus. By pooling features together across time, we force our models to classify based on the overall prevalence of the features it learns, rather than the occurrence of a particular feature at a particular time.

Note that classical approaches to feature engineering largely follow the same modeling restrictions outlined above. The engineered features used in e.g. [6] (Table 1, page 7) or [2] (Table 1, page 2) consist primarily of overall frequencies, prevalences, and rates of occurrence. These features capture properties of either a single time index or short sequences, aggregated across an entire score. The use of n-gram features also fits this mold: an n-gram is by definition a local feature of n time indices, and an n-gram featurization computes aggregate (i.e. pooled) counts of the occurrences of each particular n-gram across a score. The use of genuinely non-local features is rare. They are used in [12]: see for example the "maximum fraction of overlap with opening material within first half of movement" feature. The use of these features may account for the effectiveness of [12] in the Haydn versus Mozart classification task, which our models underperform on.

5.1 Sub-sampling Scores

The approach outlined above requires us to average features across entire scores. Each score in our corpus has a unique length, ranging from 10 to 4000 time indices. As a practical matter, it is difficult to deal with such variable-length data in machine-learning systems; our tools are designed to operate efficiently on homogeneous batches of data. One solution to this problem is to sub-sample scores; for example, [23] train models on the first s quarter notes where $s = 70$ or $s = 400$. Those authors found that the larger sample consistently outperforms the smaller one. We confirm this finding with the experiments in Table 2, which show that our models consistently perform better with larger samples of the score.

We therefore make the following compromise between using all available information from a score and operating on homogeneous inputs: we sample the first s, middle s, and last s indices from our score $\mathbf{x}$, resulting in $3s$ time indices sampled from each score. We use $s = 500$ for all experiments except the experiments in Table 2 that explore how models behave as we vary this hyperparameter. The average score in our corpus has 534 time indices, so for most scores this means we sample the entire score (for scores shorter than 500 time indices we pad out our sample with zeros). Only for scores longer than $1,500$ time indices (there are 117 in our corpus) do we lose any information with this approach.

	Sample Size					
Model	10	20	50	100	250	500
Histogram (Eqn 2)	50.0	59.0	62.0	63.0	66.1	64.2
Voices (Eqn 5)	60.0	61.6	63.9	72.0	75.5	76.9
Hybrid (Eqn 9)	59.3	62.1	68.9	77.1	79.9	81.7

Table 2. Comparison of model accuracies using a variety of samples sizes: accuracy uniformly increases with larger samples of the scores. See referenced equations (Eqn) for formal definitions of the models.

5.2 Histogram Models

The simplest models we consider are histogram models. Averaging the input data $\mathbf{x}$ over voices and time gives us a histogram vector $h \in \{0,1\}^{N+D+1}$:

$$h(\mathbf{x};\theta) = \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P}\mathbf{x}_{t,p}.$$

Multiplying this histogram by a weight matrix $W_\theta \in \mathbb{R}^{(N+D+1)\times C}$ with parameterized entries gives us a simple linear model:

$$f_\theta(\mathbf{x}) = W_\theta^\top h(\mathbf{x};\theta). \tag{2}$$

No features are learned in this model; all that is learned are the linear weights W_θ on the histogram features. The model can be interpreted as a simplified version of the global feature models discussed in Section 2. In this case, the global features are the prevalences at which each of the $N + D + 1$ note and duration symbols occur in a score.

5.3 Voice Convolutional Models

Now let's consider a simple neural model inspired by n-gram features. Let k be a number of features we desire to learn and n be a number of time indices. Define the function relu : $\mathbb{R} \to \mathbb{R}$ by $t \mapsto t\mathbf{1}(t > 0)$. Given a weight matrix $W_\theta^1 \in \mathbb{R}^{n(N+D+1)\times k}$ we can construct a "convolutional" feature representation $h_{t,p} \in \mathbb{R}^{T\times P\times k}$ at each time index t in each voice p defined by

$$h_{t,p}(\mathbf{x};\theta) = \text{relu}\left((W_\theta^1)^\top \mathbf{x}_{t:t+n,p}\right). \tag{3}$$

We define $\mathbf{x}_{t:t+n}$ to be a slice of $\mathbf{x}$ from index t to index $t+n$ (non-inclusive); when $t + n > T$, we pad $\mathbf{x}$ with zeros. We then pool these features across voices and time to construct a single, global feature representation $h \in \mathbb{R}^k$, to which we can apply a linear classifier with weights $W_\theta \in \mathbb{R}^{k\times C}$:

$$\begin{aligned} h(\mathbf{x}_t;\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h(\mathbf{x};\theta). \end{aligned} \tag{4}$$

This is a non-linear model (because of the non-linear relu "activation") and we can view h as a learned feature representation of the score $\mathbf{x}$. The weights ("filters") W_θ^1 learn to extract k relevant patterns of length n from voices, analogous to–but more expressive and compact than–classical n-gram featurizations. In our experiments we set $k = 500$ and $n = 3$; the choice of n is consistent with the pervasive use of 3-grams features in prior work [8, 9, 12, 24].

5.4 Deeper Representations

A natural way to extend the convolutional feature extraction discussed in Section 5.3 is to compose multiple layers of convolutions. Given the feature representation $h_{t,p}$ given by Equation 3 and a parameterized weight tensor $W_\theta^2 \in \mathbb{R}^{nk_1 \times k_2}$, we can construct a second layer of features

$$h_{t,p}^2(\mathbf{x};\theta) = \text{relu}\left((W_\theta^2)^\top h_{t:t+n,p}(\mathbf{x};\theta)\right).$$

We can loosely interpret such a representation as building hierarchical features of features. In principle we can build arbitrarily deep stacks of features in this way; in our experiments, we were unable to realize significant gains using architectures with more than two convolutional layers.

Building a classifier over these features proceeds identically to the shallower models:

$$\begin{aligned} h_{\text{conv}}(\mathbf{x};\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}^2(\mathbf{x};\theta) \\ f_\theta(x) &= (W_\theta)^\top h_{\text{conv}}(\mathbf{x};\theta). \end{aligned} \tag{5}$$

For this model we set $n = 3$, $k = 300$, and $k_2 = 300$.

5.5 Full-Score Convolutional Models

The models considered in Sections 5.3 and 5.4 are largely monophonic: they extract features from individual voices (although they classify based on a pool of these features gathered from all the voices). Notably, those models have no ability to capture harmonic patterns in the interactions between voices. We now consider a model that can capture these interactions.

Let $W_\theta^1 \in \mathbb{R}^{nP(N+D+1)\times k}$, $W_\theta^2 \in \mathbb{R}^{nk\times k_2}$ and consider the model

$$\begin{aligned} h_t(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^1)^\top \mathbf{x}_{t:t+n}\right) \in \mathbb{R}^{T\times k}, \\ h_t^2(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^1)^\top h_{t:t+n}\right) \in \mathbb{R}^{T\times k2}, \\ h(\mathbf{x}_t;\theta) &= \frac{1}{T}\sum_{t=1}^{T} h_t^2(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h(\mathbf{x}_t;\theta). \end{aligned} \tag{6}$$

We parameterize this model with $n = 3$, $k = 300$ and $k_2 = 300$. This model is strictly more expressive than the part-wise models (4) or (5), capable of capturing patterns that the part models can't. However, the underperformance of this model (6) relative to less expressive models (4) and (5) suggest that it is prone to capture spurious patterns, leading to overfitting (compare results in Table 3).

5.6 Harmonic Models

All the models considered so far treat pitch classes as categorical data. We recognize, for example, that C4 is distinct from E4 or G4, but not that C4 is 4 semi-tones below E4 and 7 semi-tones below G4. This section introduces a model that exploits this structural order of pitch-classes, by convolving along the pitch-axis of the input tensor.

For notational convenience, we decompose the input tensor $\mathbf{x} = \mathbf{f} \oplus \mathbf{d}$ into separate pitch components $\mathbf{f} \in \{0,1\}^{T\times P\times N}$ and note-value components $\mathbf{d} \in \{0,1\}^{T\times P\times(D+1)}$. Let $W_\theta^1 \in \mathbb{R}^{jP\times k}$ and convolve along the pitch-axis to construct a features $h_{t,n}(\mathbf{f};\theta) \in \mathbb{R}^{T\times N\times k}$:

$$h_{t,u}(\mathbf{f};\theta) = \text{relu}\left((W_\theta^1)^\top \mathbf{f}_{t,:,u:u+j}\right).$$

Here j is a hyper-parameter indicating the height of the convolution; analogous to the width-n hyperparameter in our time-domain convolutions for models (4), (5), and (6). Unlike the time domain, we find that setting a large value of j (in our models, $j = N/2$) is desirable; a similar phenomenon is observed for frequency-domain convolutions in [21].

We proceed to pool the features $h_{t,u}$ together across the pitch domain to construct $h_t \in \mathbb{R}^{T\times k}$:

$$h_t(\mathbf{f};\theta) = \frac{1}{N}\sum_{u=1}^{N} h_{t,u}(\mathbf{f},\theta). \tag{7}$$

The idea of this pooling is to construct a feature-set that is invariant to pitch translation. We are interested in learning features such as, for example, the occurrence of general major chords rather than the occurrence of a particular major chord such as the one rooted at A3. The pooling operation above precludes us from learning the latter type of feature.

We then construct a second layer of features to integrate the harmonic features h_t together with the note-value features $\mathbf{d}_t$. Using weights $W_\theta^2 \in \mathbb{R}^{k\times k_2}$ and $W_\theta^3 \in \mathbb{R}^{(D+1)\times k2}$ we build $h_t^2(\mathbf{x};\theta) \in \mathbb{R}^{T\times k_2}$. We then pool the representations h_t^2 across time and construct a linear classifier on the resulting representation:

$$\begin{aligned} h_t^2(\mathbf{x};\theta) &= \text{relu}\left((W_\theta^2)^\top h_t(\mathbf{f};\theta) + (W_\theta^3)^\top \mathbf{d}_t\right), \\ h_{\text{harmonic}}(\mathbf{x}_t;\theta) &= \frac{1}{T}\sum_{t=1}^{T} h_t^2(\mathbf{x};\theta), \\ f_\theta(\mathbf{x}) &= (W_\theta)^\top h_{\text{harmonic}}(\mathbf{x}_t;\theta). \end{aligned} \tag{8}$$

We parameterize this model with $k = 64$ and $k_2 = 500$.

5.7 Hybrid Models

Looking back at the models we've introduced, observe that the voice models (4) and (5) exploit temporal structure within voices, but pool away any harmonic patterns between voices. In contrast, the harmonic model (8) exploits harmony between voices but pools away any sequential patterns across time indices. The full-score convolutional model can capture both types of structure, but is prone to capture spurious patterns and overfit.

This motivates the introduction of our final, hybrid model that weakly combines temporal and harmonic models to increase predictive power without overfitting. The idea is to feed the input tensor separately through temporal and harmonic models to construct features representations h_{conv} (5) and h_{harmonic} (8) respectively. We combine these features in a final, linear layer using weights $W_\theta^c \in \mathbb{R}^{k_2 \times C}$ and $W_\theta^h \in \mathbb{R}^{k_2 \times C}$ to make a prediction:

$$f_\theta(\mathbf{x}) = (W_\theta^{\mathrm{c}})^\top h_{\text{conv}}(\mathbf{x};\theta) + (W_\theta^{\mathrm{h}})^\top h_{\text{harmonic}}(\mathbf{x};\theta). \quad (9)$$

Because temporal and harmonic information are only combined in the final linear layer, this model is unable to learn expressive relationships between these features, such as the classical XOR relationship [14]. As we see in Table 3, this combination increases accuracy over either the temporal or harmonic models on their own.

6. RESULTS AND CONCLUSIONS

The results of all models discussed in this paper, evaluated on the full corpus, are presented in Table 3. We sort the rows in this table by the number of scores for each composer; we observe a trend towards increasing accuracy when we have more data (with some outliers).

	Models					
Composer	(2)	(4)	(5)	(6)	(8)	(9)
Japart	0.0	13.6	13.6	9.1	18.2	13.6
Hummel	41.7	54.2	66.7	62.5	87.5	91.7
Compere	0.0	25.9	22.2	25.9	40.7	37.0
Vivaldi	30.3	94.4	91.6	54.5	45.5	54.5
Du Fay	45.7	82.9	74.3	71.4	80.0	74.3
Orto	0.0	18.6	37.2	25.6	46.5	48.8
Joplin	85.1	91.5	93.6	93.6	95.7	91.5
D. Scarlatti	44.1	59.3	62.7	78.0	79.7	72.9
Busnois	13.2	48.5	48.5	51.5	60.3	60.3
Chopin	55.3	54.2	64.5	72.4	76.3	68.4
Ockeghem	13.3	55.1	69.4	52.0	66.3	72.4
Martini	44.3	68.0	75.4	59.8	68.0	73.8
Mozart	34.8	56.3	61.6	63.6	70.2	67.5
Beethoven	72.2	82.2	83.4	78.7	84.0	89.3
de la Rue	27.5	57.9	71.3	63.5	73.6	79.2
Corelli	89.4	89.9	86.2	93.1	93.6	95.2
Haydn	85.6	75.6	71.3	79.9	82.3	83.7
Josquin	81.1	78.7	76.4	75.9	77.3	82.3
Bach	92.3	95.7	96.1	97.2	97.2	97.6
Overall	64.2	75.4	76.9	75.5	79.8	81.7

Table 3. Results of the 19-way classification problem on the full corpus for each model considered in this paper.

To compare with previous work, we train additional models on subsets of the corpus. We invite comparisons between the results in Table 4 and the results of [2], and between the results in Table 5 and the results of [6]. These comparisons are imperfect: neither [2] nor [6] report the precise scores used in their experiments. Nevertheless our corpus is derived from the same KernScores sources as [2] and [6], and contains a comparable number of scores to the counts reported in [6]. Therefore we believe our subsets are similar to the corpora used in these works and that comparison is meaningful. For future reference, the exact dataset used for the present work can be found online. [1]

	Bach	Orto	Fay	Ock.	Josq.	Rue
Bach	100.0	0.0	0.0	0.0	0.0	0.0
Orto	0.0	39.5	0.0	7.0	51.2	2.3
Du Fay	0.0	0.0	82.9	11.4	5.7	0.0
Ockegham	0.0	2.0	5.1	81.6	9.2	2.0
Josquin	0.7	1.4	1.2	3.3	84.4	9.0
de la Rue	1.1	0.0	0.0	0.6	25.8	72.5

	Bach	Orto	Fay	Ock.	Josq.	Rue
(9)	**100.0**	**39.5**	**82.9**	**81.6**	**84.4**	72.5
KNN [2]	94.5	38.9	42.9	70.0	60.6	80.6
SVM [2]	98.5	33.3	25.0	60.0	60.0	**87.1**

Table 4. (Top) Confusion matrix for the hybrid model (9), trained and evaluated on a 6-composer subset of the corpus. Compare to the results in Tables 3 and 4 (page 6) of [2]. (Bottom) Accuracy comparisons of our hybrid model to the KNN and SVM models from [2].

	Bach	Haydn	Beethoven
Bach	99.8	0.2	0.0
Haydn	3.4	93.3	3.3
Beethoven	3.0	10.6	86.4

	Bach	Haydn	Beethoven
(9)	**99.8**	**93.3**	**86.4**
SVM [6]	94.6	80.3	64.8

Table 5. (Top) Confusion matrix for the hybrid model (9), trained and evaluated on a 3-composer subset of the corpus. Compare to the results in Table 9 (page 18) of [6]. (Bottom) Accuracy comparisons of our hybrid model to the SVM model from [6].

For the popular Haydn versus Mozart string quartet classification task [5, 8, 12, 22], we were unsuccessful. The standard evaluation metric for this task is LOOCV, which we could not perform due to the computational expense of our models. With 10-fold cross validation, we observed exceedingly high variance upon repeat optimizations of the same model. However none of our optimizations exceeded 80%. Due to imbalance between Haydn and Mozart quartets (209 versus 82 scores) a classifier that simply predicts Haydn given any input achieves 71.8%.

Overall, we conclude that the convolutional models proposed in this paper perform quite well. We find this notable, given that success in neural modeling is often associated with much larger datasets. Furthermore, we do not believe that the potential of these methods has been exhausted; further investigation may yield even better convolutional architectures for composer classification.

[1] http://homes.cs.washington.edu/~thickstn/ismir2019classification/

7. ACKNOWLEDGEMENTS

This work was supported by NSF Grant DGE-1256082. We also thank NVIDIA for their donation of a GPU.

8. REFERENCES

[1] Yoko Anan, Kohei Hatano, Hideo Bannai, Masayuki Takeda, and Ken Satoh. Polyphonic music classification on symbolic data using dissimilarity functions. In *ISMIR*, pages 229–234, 2012.

[2] Andrew Brinkman, Daniel Shanahan, and Craig Sapp. Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin. In *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, pages 91–97, 2016.

[3] Giuseppe Buzzanca. A supervised learning approach to musical style recognition. In *Music and artificial intelligence. Additional proceedings of the second international conference, ICMAI*, volume 2002, page 167, 2002.

[4] Julie E Cumming. Motet & cantilena. *A Performer's Guide to Medieval Music*, 2000.

[5] William Herlands, Ricky Der, Yoel Greenberg, and Simon Levin. A machine learning approach to musically meaningful homogeneous style classification. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[6] Dorien Herremans, David Martens, and Kenneth Sörensen. Composer classification models for music-theory building. In *Computational Music Analysis*, pages 369–392. Springer, 2016.

[7] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. Global feature versus event models for folk song classification. In *ISMIR*, volume 2009, page 10th. Citeseer, 2009.

[8] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. String quartet classification with monophonic models. In *ISMIR*, pages 537–542, 2010.

[9] María Hontanilla, Carlos Pérez-Sancho, and Jose M Inesta. Modeling musical style with language models for composer recognition. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 740–748. Springer, 2013.

[10] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Musical composer identification through probabilistic and feedforward neural networks. In *European Conference on the Applications of Evolutionary Computation*, pages 411–420. Springer, 2010.

[11] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Weighted markov chain model for musical composer identification. In *European Conference on the Applications of Evolutionary Computation*, pages 334–343. Springer, 2011.

[12] Katherine C Kempfert and Samuel WK Wong. Where does haydn end and mozart begin? composer classification of string quartets. *arXiv preprint arXiv:1809.05075*, 2018.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[15] Emanuele Pollastri and Giuliano Simoncelli. Classification of melodies by composer with hidden markov models. In *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pages 88–95. IEEE, 2001.

[16] Pasha Sadeghian, Casey Wilson, Stephen Goeddel, and Aspen Olmsted. Classification of music by composer using fuzzy min-max neural networks. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 189–192. IEEE, 2017.

[17] Craig Stuart Sapp. Online database of scores in the humdrum file format. In *ISMIR*, pages 664–665, 2005.

[18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[19] Ayaka Takamoto, Mayu Umemura, Mitsuo Yoshida, and Kyoji Umemura. Improving compression based dissimilarity measure for music score analysis. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–5. IEEE, 2016.

[20] Ayaka Takamoto, Mitsuo Yoshida, Kyoji Umemura, and Yuko Ichikawa. Feature selection for composer classification method using quantity of information. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pages 30–33. IEEE, 2018.

[21] John Thickstun, Zaid Harchaoui, Dean P Foster, and Sham M Kakade. Invariances and data augmentation for supervised music transcription. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245. IEEE, 2018.

[22] Peter Van Kranenburg and Eric Backer. Musical style recognition: a quantitative approach. In *Handbook of Pattern Recognition and Computer Vision*, pages 583–600. World Scientific, 2005.

[23] Gissel Velarde, Tillman Weyde, Carlos Eduardo Cancino Chacón, David Meredith, and Maarten Grachten. Composer recognition based on 2d-filtered piano-rolls. In *ISMIR*, pages 115–121, 2016.

[24] Jacek Wołkowicz and Vlado Kešelj. Evaluation of n-gram-based classification approaches on classical music corpora. In *International Conference on Mathematics and Computation in Music*, pages 213–225. Springer, 2013.

[25] Jacek Wołkowicz, Zbigniew Kulka, and Vlado Kešelj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(1):43–55, 2008.