PKSpell: Data-Driven Pitch Spelling and Key Signature Estimation

Francesco Foscarin, Nicolas Audebert, Raphaël Fournier-S'Niehotta

▶ **To cite this version:**

Francesco Foscarin, Nicolas Audebert, Raphaël Fournier-S'Niehotta. PKSpell: Data-Driven Pitch Spelling and Key Signature Estimation. International Society for Music Information Retrieval Conference (ISMIR), Nov 2021, Online, India. hal-03300102

HAL Id: hal-03300102
https://hal.science/hal-03300102

Submitted on 26 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PKSPELL: DATA-DRIVEN PITCH SPELLING AND KEY SIGNATURE ESTIMATION

Francesco Foscarin
CNAM Paris
francesco.foscarin@cnam.fr

Nicolas Audebert
CNAM Paris
nicolas.audebert@cnam.fr

Raphaël Fournier S'niehotta
CNAM Paris
fournier@cnam.fr

ABSTRACT

We present PKSpell: a data-driven approach for the joint estimation of pitch spelling and key signatures from MIDI files. Both elements are fundamental for the production of a full-fledged musical score and facilitate many MIR tasks such as harmonic analysis, section identification, melodic similarity, and search in a digital music library.

We design a deep recurrent neural network model that only requires information readily available in all kinds of MIDI files, including performances, or other symbolic encodings. We release a model trained on the ASAP dataset. Our system can be used with these pre-trained parameters and is easy to integrate into a MIR pipeline. We also propose a data augmentation procedure that helps re-training on small datasets.

PKSpell achieves strong key signature estimation performance on a challenging dataset. Most importantly, this model establishes a new state-of-the-art performance on the MuseData pitch spelling dataset without retraining.

1. INTRODUCTION

Music listening is a complex cognitive process that involves the organization of sound events in time and frequency structures. This process creates patterns of thought that depend either on general principles of cognitive psychology or on prior knowledge and common practices of the relevant musical style system [1]. As far as tonal music is concerned, pitches are related and arranged to create a complex system of perceived stability and instability that gravitates around a tonal center, usually identified by a scale or a chord [2]. In this paper, we focus on two aspects of the tonal framework: pitch spelling and key signature.

1.1 Pitch Spelling

In tonal music, the classification of pitches in 12 *pitch-classes*, each one uniquely identifying a set of frequencies that are n octaves apart, is enriched with some tonal information and creates the higher-level representation of *tonal-pitch-classes*. Each *tonal-pitch-class* consists of a *diatonic name* in $[C, D, E, F, G, A, B]$ and an *accidental* among double-flat (𝄫), flat (♭), natural(♮), sharp (♯), double-sharp (𝄪). [1] For every pitch-class, there are multiple corresponding tonal-pitch-classes, called *enharmonic equivalents* (see Fig. 1). The task of choosing a single name between the possible enharmonic equivalents is called *pitch spelling*.

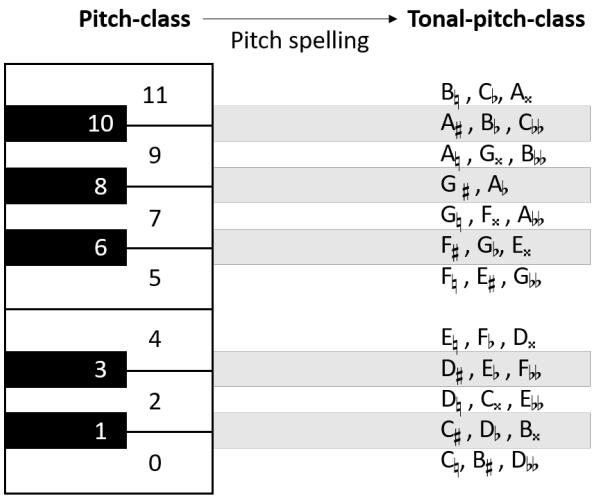

Figure 1. Pitch-classes and corresponding tonal-pitch-classes.

1.2 Key Signature

Key signatures inform about the tonal center of the piece and are commonly identified in music by a certain number of sharps or flats (whose positions are fixed) or by a tonal-pitch-class (see Fig. 2). Combined with pitch spelling, they also improve readability, according to the principle of *notation parsimony* [3], *i.e.,* the minimization of the number of symbols (accidentals) displayed in the score.

Tightly related to the key signature, the *key* also adds information about the mode (major or minor). Keys are considered in literature either on a global scale (*i.e.,* one *global key* for one piece) or a local scale (*i.e.,* multiple *local keys* for one piece) resulting from modulations and tonicizations (see [4] for a complete description of these concepts). A formalization of the relation between local keys, global key, and key signatures is missing in the literature and it is not the goal of this paper to have a musicological discussion on this topic. As a first approximation, the key signature can

© F. Foscarin, N. Audebert, and R. Fournier S'niehotta. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Foscarin, N. Audebert, and R. Fournier S'niehotta, "PKSPELL: Data-Driven Pitch Spelling and Key Signature Estimation", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.,* Online, 2021.

[1] More accidental types, like triple sharps could exist in theory, but they are not used in common music notation.

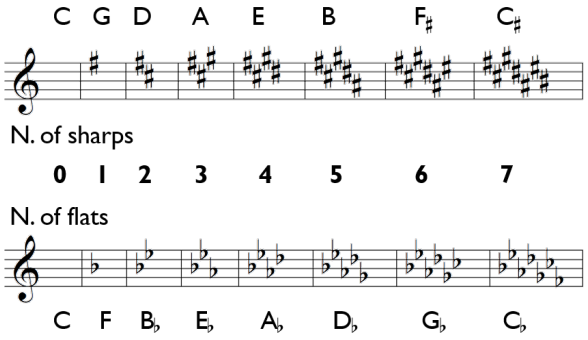

Figure 2. Corresponding key signatures representation in a musical score, tonal-pitch-class and number of accidentals.

be considered a global feature of the piece. It occasionally changes if the tonal center shifts significantly from its previous position, although it does not move as much as the local key. A possible motivation is that, since the key signature is employed in music notation, the principle of parsimony applies and it "smoothes" short key changes.

1.3 Pitch Spelling and Key Signature Estimation

Multiple reasons motivate systems that perform pitch spelling and key signature estimation. Such information is necessary for the creation of full-fledged musical scores, *e.g.,* as the last step of automatic music transcription or music generation systems. Furthermore, those elements are employed for other MIR tasks: key signature annotations are useful for music search and section identification, and pitch spelling facilitates harmonic analysis and melodic pattern matching [3, 5]. However, they are not represented in "low-level" formats, such as audio and MIDI files [2] , which constitute most of the available digital musical encoding.

1.4 Related Works

Several authors have developed automatic systems that perform pitch spelling from MIDI data. Most of them consist in algorithmic approaches based on musicological insights only, without any learning. Some authors think the selection of the appropriate tonal-pitch-class is a function of the local key [6, 7] or a combination of the local key and voice-leading information [5, 8], *i.e.,* the temporal evolution of notes within each voice. Others base pitch spelling on a type of interval optimizations [9], on the principle of notational parsimony, or a combination of the two criteria [3]. Another formulation of the problem as a generative probabilistic model is proposed by Teodoru et Raphael [10]. They model voices with independent Markov chains conditioned on a hidden state that contains the local key information.

For evaluation purposes, the early algorithmic approaches were compared by Meredith [7] on a dataset of 216 pieces by 8 baroque and classical composers. The highest accuracy at the time was obtained by Meredith's ps13 algorithm, especially when small temporal variations were introduced to simulate a MIDI file generated from a performance. Teodoru and Raphael [10] also tested their approach on the same dataset (without tempo deviation) with high accuracy. More recent works [9, 11] did not increase the total accuracy. In this paper, we compare our results with Meredith's ps13 algorithm and Teodoru and Raphael's approach with conditional independent voices (CIV).

While those approaches yield a very high accuracy ($\geq$ 99% of notes are correctly spelled), they are not designed to be easily employed in larger MIR pipelines. ps13 has 3 different outputs, whose tonal pitch classes are transposed by diminished seconds (*e.g.,* one piece in C♯, one in D♭ and one in B𝄪) and does not indicate how to select the "best" version. Both ps13 and CIV parameters are set by hand [3] , thus making them difficult to generalize across composers and datasets. Moreover, the code of CIV is not publicly available. Finally, both approaches do not provide key signatures, so they cannot be used alone to produce a complete pitch encoding for a musical score. Other pitch spelling systems are implemented in music notation software (*e.g.,* Finale, MuseScore) but no information about their functioning mechanism and performance is available and they require key signatures as input.

Little attention has been given in the literature to the *key signature estimation* problem from MIDI files, in favor of the related task of key estimation [12–15] (see Section 1.2). A direct comparison of our results with other approaches is therefore not possible, but we put our results into perspective by comparing them with the state-of-the-art approach for global key estimation of López et al. [15].

1.5 Our approach

We propose the PKSpell system for jointly estimating pitch spelling and key signature. It yields high accuracy, is easy to integrate into a MIR pipeline, and works on any kind of MIDI file (including MIDI generated from human performance) or other symbolic encodings. Trained on the ASAP dataset [16], PKSpell achieves strong key signature estimation performance on the Albrecht dataset [14] and establishes a new state-of-the-art pitch spelling performance on the MuseData dataset [7] without retraining. Implementation and pre-trained model are publicly available [4] .

We design a deep learning approach based on recurrent neural networks (RNN) that can model correlations in input sequences of variable lengths [17]. We use a dedicated network structure inspired by musicological considerations on the relation between local keys, pitch spelling, and key signatures (details in Section 2.3). Our system models each input note with a pair (pitch-class, duration) that does not require high-level temporal information such as downbeat and time signature or voice separation to be produced. With the proposed dedicated preprocessing of the note durations and data augmentation procedure (see Sections 2.1 and 2.2), our approach can generalize to different tempos, time signatures, and key signatures. This makes it possible to train our model on a small-size dataset of musical scores. Moreover,

[2] Albeit key signatures can be encoded in a MIDI file as metadata, they are often absent.

[3] CIV could be trained, but the authors report that the results are worse than with handset parameters.

[4] See `https://github.com/fosfrancesco/pkspell`

by cross-validating on a separated dataset, we show that with the pre-trained parameters, our system can correctly handle a variety of different tonal pieces. Finally, our multi-task approach to pitch spelling and key signature estimation increases the performance on both tasks.

2. METHOD

The pitch spelling and key signature information are the results of the relations between different notes. We employ a recurrent neural network architecture, which can learn all those relations for pieces of different lengths. This class of models employs a big number of parameters to correctly learn the input-output function. Our choice of input/output representation helps to keep their number as low as possible, and the data augmentation procedure we propose allows us to learn them from a relatively small dataset.

2.1 Input and output formats

Multiple approaches have been proposed for the problem of transforming a MIDI file into a sequential representation [18–20]. Since we target all kinds of MIDI inputs (see [21] for a description of different MIDI formats), we cannot assume to have information such as voice separation, time signature, downbeat, and beat positions. We employ a simple note-based representation that was proposed by Lopez [15]: a piece is modeled as a sequence of notes $\aleph = \{\eta_1, \eta_2, ..., \eta_N\}$, ordered according to the temporal position of their onsets. If multiple notes have the same onset position, we take the notes in low-to-high order. [5] For each note η, we consider two features: pitch-class $p[\eta]$ and duration $d[\eta]$. The usage of note durations is common in key estimation approaches (*e.g.,* [1, 12]), and stems from the musicological intuition that longer notes have a stronger impact than shorter notes in defining the tonal context.

To group all possible note durations in a limited number of classes, we run an (unsupervised) 1-dimensional *k-means* algorithm for all note durations (see Fig. 3). We use the dynamic programming algorithm presented by Gronlund et al. [22] that has complexity $O(kN + NlogN)$, for k classes and a sequence of length N. We select $k = 4$. This classification cannot be considered more than a rough indication of relative duration and it is not meant to precisely identify beats, downbeats, and other metrical information. Moreover, it will show its limits if there are tempo changes or time signature changes inside the piece. Yet, we found that it improved our model results at a small computational cost and, compared to other possible approaches, *e.g.,* quantization to some discrete durations, it has the advantage of generalizing to different tempos and time signatures.

The output is a sequence of tonal-pitch-classes $\tau[\eta]$ (among the 35 in Fig. 1) and a sequence of key signatures $\kappa[\eta]$ (among the 15 in Fig. 2), one for each note η in input. Computing a key signature for each note might seem excessive, since we may expect the key signature to change only a few times during a piece. However, this is necessary as we do not know in advance how many changes there are in a piece, nor do we have precise metrical information.

[5] However, the ordering choice does not impact the results.

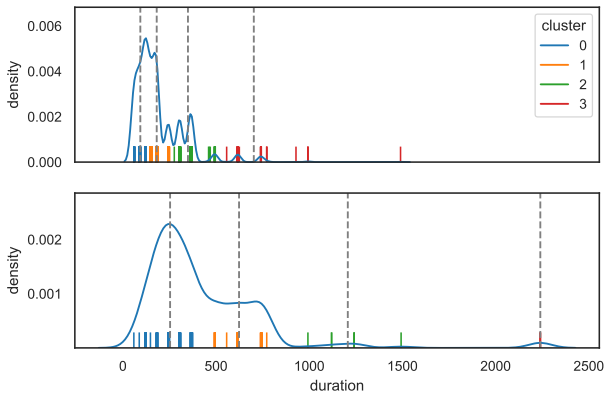

Figure 3. One-dimensional clustering of note durations for 2 pieces expressed in milliseconds. Each vertical line represent a duration and dotted grey lines are cluster centroids. The piece at top is faster than the piece at the bottom. To facilitate the understanding, the kernel density estimation of the note durations is also displayed.

2.2 Data augmentation

Studies in cognitive psychology [23] have proved that listeners perceive as identical two pieces if all their notes are transposed by the same *interval*. It is common to use this property of music perception to augment a dataset by transposition [18, 24]. For our goals, we must transpose pitch-classes (input) and tonal-pitch-classes and key signatures (output) to have a correct ground truth for training.

The possible transpositions of tonal-pitch-classes and key signatures move through a *spiral of fifths* [12]. When reported to the same octave, each transposition can be identified with a *diatonic interval*, notated with an interval number and type, *e.g.,* augmented 4th, perfect 5th, minor 2nd. For pitch-classes, instead, we are limited to 12 *chromatic intervals* that can be simply identified by integers. In Fig. 4 we report the most common diatonic intervals (the ones closer to the center of the spiral of fifths) associated with their respective chromatic interval.

Since our goal is to learn an input-output mapping, we cannot accept multiple sequences of tonal-pitch-classes that correspond to the same sequence of pitch-class. That means that we need to select only one diatonic interval for every chromatic interval. To do so, we use a heuristic based on the principle of parsimony, *i.e.,* we choose the diatonic transposition which generates the set of tonal pitch classes with the lowest number of accidentals. For each piece and each chromatic transposition, we transpose the tonal pitch class by all corresponding diatonic intervals, then we discard transpositions that contain non-accepted accidentals (*e.g.,* triple sharps). Finally, we select as the "valid" diatonic transposition the one with the smallest count of accidentals, where 𝄫 and 𝄪 count as 2, and ♯ and ♭ count as 1. This allows us to produce up to 11 variants for each piece, although the number is lower if all diatonic intervals for a certain

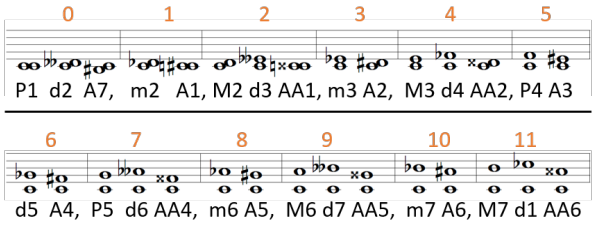

Figure 4. Chromatic intervals (orange), corresponding diatonic intervals (black) and examples from the tonal-pitch-class C. The abbreviation for diatonic types are "P": perfect, "M": major, "m":minor, "A": augmented, "d": diminished, "AA": doubly augmented.

chromatic interval generate a discarded transposition.

2.3 RNN Architecture

With our input and output choice, the pitch spelling and key signature estimation problems for one piece can be seen as a sequence of multi-task classification problems. Our goal is to assign to each input (*i.e.,* pitch-class p + duration d) two labels: one among the available tonal-pitch-classes τ and one among the key signatures κ. Formally, for a piece C and each note $\eta \in \aleph_C$ we seek to learn the function $F : x[\eta] \rightarrow y[\eta]$, where $x[\eta] = p[\eta], d[\eta]$ is our input and $y[\eta] = \tau[\eta], \kappa[\eta]$ is our output.

We want the output for each note to be dependent on the notes around it, and the length of our sequences (*i.e.,* the number of notes in a piece) is not fixed. We select as the core of our model a recurrent neural network (RNN), that can store information about the "context" in its internal state for variable-length sequences of inputs. Such a model can be trained in a supervised fashion on an annotated dataset that contains for each piece a sequence of pairs $\{(x[\eta], y[\eta]), \forall \eta \in \aleph_C\}$. We optimize the parameters θ w.r.t. a classification loss function $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \sum_{C} \sum_{\eta \in \aleph_C} \mathcal{L}(F_\theta(x[\eta]), y[\eta])$$

We select $\mathcal{L}$ as the sum of two cross-entropy-loss functions [25], one for the key signature and one for the tonal-pitch-class. Since $\mathcal{L}$ is differentiable, the model parameters can be optimized using Stochastic Gradient Descent (SGD).

We use a bidirectional RNN to tie the output at a certain note to the past and future inputs. From a musicological standpoint, "seeing the future" is useful *e.g.,* to know where a certain note will *resolve*. As shown in Fig. 5, the model has two recurrent layers, each one coupled with a linear layer. The first produces the tonal-pitch-classes and the second produces the key signatures. This architecture is based on the musicological hypothesis that pitch spelling depends on the local key [6, 7] and the key signature is a "smoothed" version of the local key. Assuming the first layer encodes information about the local key, we aggregate this information in the second recurrent layer to infer the key signature.

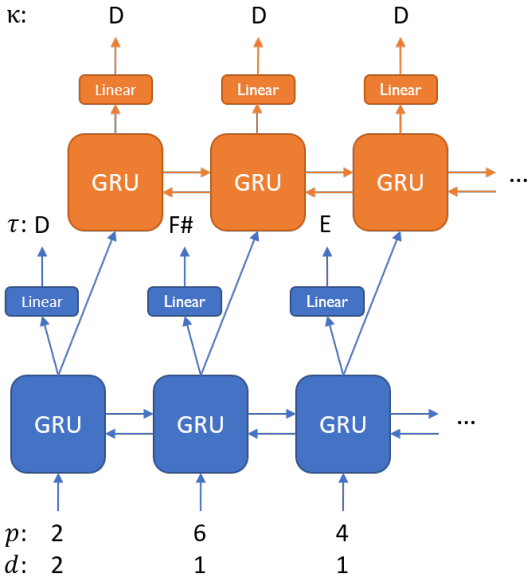

Figure 5. Model architecture. For each pitch-class p and duration d in input, a tonal-pitch-class τ and a key signature κ are produced.

According to the principle of *multi-task learning* [26,27], jointly producing two outputs allows the model to learn shared representations, thus enabling a better generalization on both our original tasks.

3. EXPERIMENTS AND RESULTS

We train our model on the pieces from the ASAP dataset [16]. The dataset provides 222 compositions from several composers over two centuries: Bach, Beethoven, Chopin, Schubert, Haydn, Liszt, Schumann, Mozart, Rachmaninoff, Ravel, Debussy, Scriabin, Glinka, Brahms, Prokofiev, Balakirev. We remove two pieces because they overlap with the dataset that we use for pitch spelling evaluation (see Section 3.2). All the information we need for training can be easily extracted from digitally encoded musical scores, *i.e.,* MusicXML scores for the ASAP dataset. Such scores contain note durations, tonal-pitch-classes, and key signatures. From tonal-pitch-classes it is straightforward to produce pitch-classes using the function in Fig. 1. Duration and pitch-classes are then encoded as one-hot vectors, concatenated to a single vector, and fed into our model. Training on real MIDI performances could also be performed as long as a note-wise score to performance alignment is available. Unfortunately, ASAP does not provide this alignment, and other datasets that provide such information are considerably smaller.

To evaluate the benchmarked approaches, we use the accuracy, *i.e.,* the percentage of correctly inferred symbols. We also use the error rate (1 − accuracy) to improve the readability of some results.

3.1 Hyperparameter search

Our model has five major hyperparameters to consider: type of optimizer, learning rate, batch size, type of recurrent cell, and number of parameters (number of layers and hidden state dimension for each RNN). To find the best combination, we perform a grid search by training our model on 85% of ASAP (187 pieces, 2033 after data augmentation) and validating on the remaining 15% (33 pieces).

Optimizer. We compare various optimizers from the deep learning literature, including gradient descent algorithms with adaptive learning rates or momentum. We find no significant difference in accuracy; though Adam [28] is easier to tune and converges faster than traditional SGD.

Learning rate. All networks are trained for 40 epochs, *i.e.,* 40 passes over the whole augmented training set. We find that a starting learning rate of 0.01, divided by 10 after 20 epochs, allows for fast and robust convergence.

Batch size. We feed mini-batches of sequences in parallel to our model. For batches larger than 32, changing the batch size has no impact on performance. Larger batches tend to increase convergence speed but with a higher GPU memory consumption; thus we settle for a batch size of 32.

Recurrent cell. We compare LSTM [29] and GRU [30] cells. While the accuracy is similar with the two cells, we find that GRU cells are faster and use less memory.

Number of parameters. There are two ways to increase the number of parameters in the RNN: widen it by increasing the hidden dimension, or deepen it by stacking more layers. More parameters generally entail better performance on the train set but not necessarily on the test set due to overfitting. Dropout [31] is required to alleviate overfitting for a depth greater than 1 or a hidden dimension higher than ≈ 200. For each of the two RNN in our final model, we set depth 1 and hidden dimension 300 (150 in each direction).

3.2 Main results

We evaluate the pitch spelling and key signature estimation results separately.

For pitch spelling, we compare with ps13 and CIV on the MuseData dataset proposed by Meredith [7]. It consists of 216 pieces (195 972 notes) from 8 classical composers (see Table 1) and is also available in a "noisy" version, where some noise is artificially introduced in the onset and offset positions to roughly simulate human performances. There are no pieces in common with our training dataset. It is worth noting that both ps13 and CIV tune their parameters on the test data; this can induce overfitting and thus an optimistic estimate of the system's performance. We may notice, for example, that the pieces used to train CIV (all Beethoven's, one-third of Haydn's, and one-third of Mozart's), correspond to the pieces in which CIV has the highest performance compared to our system. Moreover, both ps13 and our system are evaluated on the "noisy" dataset, while CIV is evaluated on the quantized dataset. We report our evaluation results in Table 1. PKSpell establishes new state-of-the-art performances on the pitch spelling task. It does 256 errors, around 25% less than CIV (343) and 75% less than ps13 (1064). The learned model has an excellent generalization, going close to zero error if the pieces have a unique strong tonal center (*e.g.,* Corelli, Handel, Telemann, Vivaldi). More difficult are instances when multiple modulations happen during a short time span (*e.g.,* Bach chorales), and especially around abrupt key signature changes (*e.g.,* Beethoven). The error count for Haydn is particularly high because of the piece Symphony No. 100 in G major, where, at measure 166 there is a sudden change from D♭ major to C♯, to make the music easier to read with fewer accidentals. This kind of enharmonic key change is rare in music and our system fails to detect it. [6] It is worth noticing that, while our model allows a pitch-class to be classified in a non-correspondent tonal-pitch-class (*e.g.,* 2→G♯), this kind of error is completely absent in our results. Our model learns very easily to perform the mapping of Fig. 1, especially when using the augmented dataset.

While a direct evaluation of the key signature estimation is not possible due to the lack of other approaches targeting this task, we put into perspective our results by evaluating our system on the Albrecht dataset [14] and comparing it with the state-of-the-art results for the global key signature estimation of López et al. [15]. Our task can be considered easier because we do not need to separate major keys from minor keys; however, while we are considering all enharmonic key signatures (*e.g.,* D♭ with five flats is considered equivalent to C♯ with seven sharps), López is flattening all enharmonic versions of a key to the same class. After removing the pieces in common with ASAP, we are left with 932 pieces. We correctly classify 97% of the global key signatures. For comparison, López et al. correctly classify 94% of the keys. Of the 29 misspelled pieces, eight are mapped to enharmonically equivalent key signatures, in particular, there is confusion between the C♯ and D♭ and between F♯ and G♭. The remaining 21 wrong estimations are off by one accidental, *i.e.,* the predicted key signature is the relative 4th or 5th of the true key signature.

3.3 Ablation Studies

We perform several ablation studies to understand how our design choices impact the model performance. For each experiment, we remove one element from PKSpell and see how this affects the model performance. If the element is useful, we expect a reduction in the accuracy (see Fig. 6).

Single RNN. As previously introduced, PKSpell uses two recurrent layers, one for pitch spelling and another for key signature estimation. To evaluate this idea, we build a model that has a single recurrent layer for both tasks. A detailed analysis shows that one-layer-PKSpell slightly outperforms the two-layer model when the tonal center is very stable. However, the two-layer model majorly improves the results for more pieces with modulations and key changes. On the ensemble of considered composers, the usage of two separate RNN layers boosts the accuracy, especially for key signature estimation (+3%).

Separate learning. Multi-task learning can help leverage domain-specific information contained in related but

[6] ps13 manually transposes half of this piece before evaluation. There is no indication of its treatment by CIV.

	Bach	Beethoven	Corelli	Handel	Haydn	Mozart	Telemann	Vivaldi	**Total**
ps13 [7]	0.17% (42)	1.41% (345)	0.04% (11)	0.26% (68)	0.94% (220)	0.23% (282)	0.34% (82)	0.39% (114)	0.59% (1064)
CIV [10]	0.10% (25)	**0.10% (25)***	0.08% (20)	0.02% (6)	**0.45% (110)***	0.33% (79)*	0.05% (13)	0.27% (65)	0.18% (343)
PKSpell	**0.08% (19)**	0.23% (56)	**0.02% (5)**	**0.02% (5)**	0.49% (121)	**0.14% (35)**	**0.02% (4)**	**0.04% (11)**	**0.13% (256)**

Table 1. Error rate and the number of errors (between parentheses) for different composers in Meredith's Musedata pitch spelling dataset [7]. For ps13 and CIV, we took these values from the respective papers. The symbol * marks the results for the composers used to set the parameters for CIV. The best result for each composer is highlighted in bold.

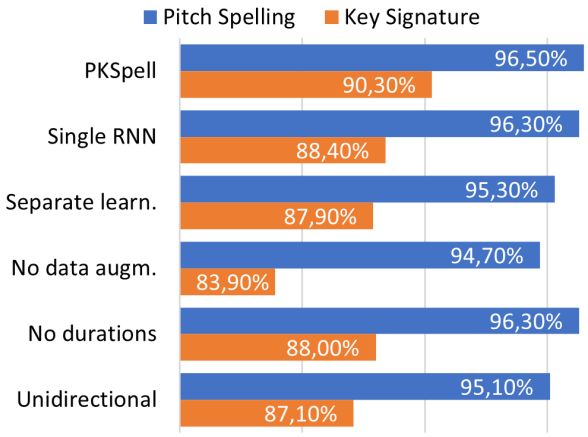

Figure 6. Accuracies for the ablation tests. Results are reported on the validation set of ASAP (33 pieces).

different tasks. This is the case in our system, as the accuracy for both PS and KS estimation improves when trained jointly compared to two distinct RNNs used separately.

No data augmentation. In theory, more data should improve the generalization of the model, provided that the augmented samples are representative of future observations. In our case, data augmentation provides a significant accuracy boost, especially for key signature estimation.

No durations. We use the duration of notes as an input feature for our model. While this is common in key estimation systems, multiple approaches to pitch spelling in the literature do not use this information [3, 7]. We find that durations improve our results, especially for KS estimation.

Unidirectional RNN. The RNN layers we consider can process a sequence either in one direction (usually left-to-right, LTR) or in a bidirectional manner, both LTR and RTL. We expect the bidirectional processing to perform better since the model also "sees" future notes. In our ablation experiment, both PS and KS accuracies increase by more than 1 point by using both directions (we keep the same number of parameters by dividing the hidden dimension by 2 when we use the bidirectional model). Note, however, that the unidirectional LTR model still works reasonably well and could be used for a real-time version of our system.

4. CONCLUSIONS AND PERSPECTIVES

We introduce PKSpell, a novel system for joint pitch spelling and key signature estimation that reaches new state-of-the-art performances on pitch spelling, is easy to integrate into a MIR pipeline, and works on any MIDI file, including human performances. To reach this goal, we perform multi-task learning with an RNN model inspired by musicological insights. We propose a data augmentation procedure and a preprocessing of note durations to generalize to different transpositions, tempos, and time signatures. We consider a minimal set of inputs (pitch-classes and note durations) that do not require high-level information such as time signature, voice separation, and downbeat position. The pre-trained model we provide can be used "as-is" and offers good performance for classical music of different centuries. Thanks to the ablation study, we directly observe the impact of the design choices of our system.

Possible future work concerns the evaluation of PKSpell on different styles of tonal music *e.g.,* pop, folk, and jazz. We expect it to perform well on pop and folk due to their low harmonic complexity. Jazz can be more challenging because of the extensive use of chord extensions and alterations. It is also interesting to study how non-strictly-tonal music in the ASAP dataset (*e.g.,* Debussy) impacts the training of our model. While concepts such as key signature and pitch spelling are less significant for non-tonal music, they are still used to write musical scores. Since PKSpell is not based on strong tonal principles, we expect it to be able to learn those rules as long as some coherent rules exist for pitch spelling and a large enough dataset is provided. A more in-depth analysis can be performed on the treatment of rhythmical information, by considering different ways to group duration values and by varying the number of groups. Other improvements may be done on the model architecture: the state of the art for sequence-to-sequence problems has shifted toward recurrent attentional models and transformers. However, the length of the sequences we are considering (more than 15 000 notes for certain pieces) poses a considerable challenge for full attentional mechanisms, whose memory requirements are quadratic with the input sequence length. Models that scale linearly have been recently proposed [32, 33] and could be a solution for this problem. Finally, it would be interesting to employ our model to perform local and global key estimation.

5. ACKNOWLEDGMENTS

We thank David Meredith for releasing the MuseData corpus and ps13 Lisp implementation. Additional thanks go to Tim Bradshaw for help running ps13 on current systems.

6. REFERENCES

[1] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 1990.

[2] R. Van Egmond and D. Butler, "Diatonic connotations of pitch-class sets," *Music Perception*, vol. 15, no. 1, pp. 1–29, 1997.

[3] E. Cambouropoulos, "Pitch spelling: A computational model," *Music Perception*, vol. 20, no. 4, pp. 411–429, 2003.

[4] N. N. López, L. Feisthauer, F. Levé, and I. Fujinaga, "On local keys, modulations, and tonicizations," in *Digital Libraries for Musicology (DLfM)*, 2020.

[5] D. Temperley, *The cognition of basic musical structures*. MIT Press, 2001.

[6] E. Chew and Y.-C. Chen, "Real-time pitch spelling using the spiral array," *Computer Music Journal*, vol. 29, no. 2, pp. 61–76, 2005.

[7] D. Meredith, "The ps13 pitch spelling algorithm," *Journal of New Music Research*, vol. 35, no. 2, pp. 121–159, 2006.

[8] H. C. Longuet-Higgins and M. Steedman, "On interpreting Bach," *Machine intelligence*, vol. 6, 1971.

[9] B. Wetherfield, "The minimum cut pitch spelling algorithm," in *International Conference of Technologies for Music Notation and Representation (TENOR)*. Hamburg University for Music and Theater, 2020, pp. 149–157.

[10] G. Teodoru and C. Raphael, "Pitch Spelling with Conditionally Independent Voices." in *International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 201–206.

[11] A. K. Honingh, "Compactness in the Euler-lattice: A parsimonious pitch spelling model," *Musicae Scientiae*, vol. 13, no. 1, pp. 117–138, 2009.

[12] E. Chew, "The spiral array: An algorithm for determining key boundaries," in *International Conference on Music and Artificial Intelligence*. Springer, 2002, pp. 18–31.

[13] D. Temperley and E. W. Marvin, "Pitch-class distribution and the identification of key," *Music Perception*, vol. 25, no. 3, pp. 193–212, 2008.

[14] J. Albrecht and D. Shanahan, "The use of large corpora to train a new type of key-finding algorithm: An improved treatment of the minor mode," *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 1, pp. 59–67, 2013.

[15] N. N. López, C. Arthur, and I. Fujinaga, "Key-finding based on a hidden Markov model and key profiles," in *Digital Libraries for Musicology (DLfM)*, 2019, pp. 33–37.

[16] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[17] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, pp. 132–306, 2020.

[18] G. Micchi, M. Gotham, and M. Giraud, "Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.

[19] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.

[20] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Coupled recurrent models for polyphonic music composition," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[21] D. Back, "Standard MIDI-file format specifications," 1999, accessed May 5, 2021. [Online]. Available: http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html

[22] A. Grønlund, K. G. Larsen, A. Mathiasen, J. S. Nielsen, S. Schneider, and M. Song, "Fast exact k-means, k-medians and Bregman divergence clustering in 1d," *arXiv preprint arXiv:1701.07204*, 2017.

[23] W. J. Dowling, "Scale and contour: Two components of a theory of memory for melodies." *Psychological review*, vol. 85, no. 4, p. 341, 1978.

[24] T.-P. Chen and L. Su, "Harmony transformer: Incorporating chord segmentation into harmony recognition," *neural networks*, vol. 12, p. 15, 2019.

[25] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[26] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *International conference on Machine learning*, 2008, pp. 160–167.

[27] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 8599–8603.

[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[30] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724–1734.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[32] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[33] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A Nyström-based algorithm for approximating self-attention," 2021.