Automatic Piano Fingering from Partially Annotated Scores using Autoregressive Neural Networks

Pedro Ramoneda*
Universitat Pompeu Fabra

Dasaem Jeong
Sogang University

Eita Nakamura
Kyoto University

Xavier Serra
Universitat Pompeu Fabra

Marius Miron
Universitat Pompeu Fabra

ABSTRACT

Piano fingering is a creative and highly individualised task acquired by musicians progressively in their first music education years. Pianists must learn to choose the order of fingers to play the piano keys because scores do not have engraved finger and hand movements as other technique elements. Numerous research efforts have been conducted for automatic piano fingering based on a previous dataset composed of 150 score excerpts fully annotated by multiple expert annotators. However, most piano sheets include partial annotations for problematic finger and hand movements. We introduce a novel dataset for the task, the ThumbSet dataset, containing 2523 pieces with partial and noisy annotations of piano fingering crowdsourced from non-expert annotators. As part of our methodology, we propose two autoregressive neural networks with beam search decoding for modelling automatic piano fingering as a sequence-to-sequence learning problem, considering the correlation between output finger labels. We design the first model with the exact pitch representation of previous proposals. The second model uses graph neural networks to more effectively represent polyphony, whose treatment has been a common issue across previous studies. Finally, we finetune the models on the existing expert annotations dataset. The evaluation shows that (1) we are able to achieve high performance when training on the ThumbSet dataset and that (2) the proposed models outperform the state-of-the-art hidden Markov models and recurrent neural network baselines. Code, dataset, models, and results are made available to enhance the task reproducibility, including a new framework for evaluation.

CCS CONCEPTS

• **Information systems** → **Music retrieval**.

KEYWORDS

Datasets, Neural Networks, Automatic Piano Fingering, Music Education Technologies, Music Information Retrieval

ACM Reference Format:
Pedro Ramoneda, Dasaem Jeong, Eita Nakamura, Xavier Serra, and Marius Miron. 2022. Automatic Piano Fingering from Partially Annotated Scores using Autoregressive Neural Networks. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548372

*Corresponding author: pedro.ramoneda@upf.edu

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548372

1 INTRODUCTION

Automatic Piano Fingering (APF) aims at modeling the finger movements of humans when playing a given score on the piano. It assumes automatically labeling each note in the musical score with a finger number from either the right or the left hand: thumb (1), index (2), middle (3), ring (4) and (5) Pinky. According to [24] piano fingering is one of the most difficult human tasks, requiring years of training. Because finger and hand movements are not usually indicated on scores as clearly as other technique elements, pianists must develop their piano fingering ability, i.e., determine the order of their fingers to play the piano keys. To that extent, pianists must adapt the fingering at each moment according to the subsequent fingering patterns' needs [23]. Fingering must support the musical content of the work in all its facets: articulation, tempo, dynamics, rhythm, style, and character [23]. Moreover, it has to be as comfortable as possible [3], considering that the pianists choose the fingers to play the notes in the score in a highly individualised manner [15]. The mutual agreement between fingering of different professional pianists is around 70% [19].

Piano fingering is one of the so-called psycho-motor skills and is fundamental in music education because pianists learn to move fingers and hands progressively throughout the first education years [27]. Therefore, fingering is one of the fundamental dimensions for assessing piano performance [12]. The main application of APF systems is to support piano students, and other piano-related tasks can be enhanced through understanding piano fingering, such as difficulty analysis [20, 27], polyphonic transcription [17], rhythm transcription [21] or score reductions [20].

APF has been studied over decades [18, 25, 34] and is still an actively studied topic [8, 19, 35], in the music information retrieval field. According to Nakamura et al. [19] the limitations of APF methods are related to: (a) temporal features and high-level contexts; (b) interdependence of the two hands; (c) individuality and model adaptation; and (d) cost vs. statistical viewpoints. Thanks to the recent release of a dataset [19], the APF task has seen more contributions [8, 35]. Although the previous state of the art has still held with the Hidden Markov Models (HMM) presented by Nakamura et al. [19]. They also presented two simple neural network models. However, the models performed worse than the HMMs, suggesting a limitation of applying deep learning methodology to this task with the available dataset. In the present research, we want to design more complex, and data-intensive models able to

(a) encode polyphony with graph neural networks and (b) learn high-level contexts with an autoregressive sequence-to-sequence (seq2seq) model.

Previous APF research [19] addresses the need for additional training data when training machine learning models which generalise better, given that the optimal fingerings differ across pianists. In particular, most music scores include partial annotations solely regarding the problematic finger and hand movements. The editor or the composer adds partial annotations in published scores, or the music teachers or students add them afterwards. However, there is no public dataset of partially annotated fingerings for research on APF modelling. Therefore, we collect public domain scores from MuseScore to build an open dataset, *ThumbSet*. *ThumbSet* allows training complex deep learning fingering models on a larger partially annotated dataset.

As a first contribution, we release *ThumbSet*, a dataset of 2523 pieces with partially annotated finger labels. As a second contribution, we empirically demonstrate that it is possible to train more extensive and complex deep learning architectures on the proposed noisy dataset with similar performance to training on fully annotated data. In addition, we propose two autoregressive models: a long short-term memory network (LSTM) and a graph neural network (GNN). While the LSTM takes the pitches as input, the GNN encodes the polyphonic relationships between notes, with the pitch as nodes and the relationships as edges, employing graph neural networks. The proposed models outperform the state-of-the-art hidden Markov model and previous neural networks methods. In our experiments, we propose a novel evaluation metric and use it to show that the autoregressive models improve the sequential coherence of fingerings. Finally, another challenge is to improve the open science on the APF task. For open science purpose, we release the *ThumbSet* dataset [1], code, models, results [2] and evaluation framework [3] as open source.

The remainder of this paper is organised as follows. In Section 2, we review the related work. We describe the novel *ThumbSet* dataset in Section 3. The proposed approach is given in Section 4. Section 5 summarises our experimental work and the paper is concluded in Section 6.

2 RELATION WITH PREVIOUS WORK

Several techniques have been proposed for modelling piano fingering. While expert systems [7, 25] were used for APF in the 1900s, later local search algorithms grounded in cost functions were developed [2], and more recently data-driven methods were used [18, 19, 34].

With regards to data-driven methods, we highlight the HMM models and deep learning methods proposed by Nakamura et al. [18, 19]. Although there is evidence that within a 6-note context, pianists may decide to finger note-by-note, they usually consider all past decisions. The neural network methods lack information about the final output of the previous sequence elements, while HMMs use Viterbi decoding to force the sequential coherence. In other words, the neural network methods estimate the finger probability for each note, while the proposed HMM understand the fingering process as a sequence, taking into account all other decisions. We want to fill this gap by using autoregressive neural networks in the present study. They introduced the PIG (PIano fingernG) Dataset [19] compiling 150 piano compositions by Western classical music composers and providing annotated fingerings by professional pianists. The dataset comprises the miscellaneous set containing 120 pieces, each with fingerings provided by one or two pianists, and the composer-specific set containing pieces by Bach, Mozart, and Chopin (10 pieces for each) with fingerings provided by 4, 5, and 6 pianists, respectively. The former set was used as training data and the latter as test data in the original study [19]. Most pieces in the PIG Dataset are excerpts of musical compositions. A typical length is one page, with roughly 20 bars and 300 notes. However, the dataset does not provide a validation set, and it only brings a train and test split because the first proposed models [18] do not need to tune hyperparameters. Furthermore, more recent research projects [8, 35] still do not provide a clear validation split. For these reasons, we propose an official validation split to be used, allowing the reproducibility and fairness of future research. The new validation set is separated from the training set, with the 30 pieces fingered by the annotator NE. We removed 20 pieces annotated by the same annotator from the test set to reduce possible biases between validation and test sets.

The evaluation metrics proposed by Nakamura et al. [19] quantify the accuracy and individuality of piano fingering: the general match rate (M_{gen}) indicates how closely the estimation agrees with all the ground truths; the highest match rate (M_{high}) focuses on the ground truth closest to the estimation for each piece; the soft match rate (M_{soft}) judges whether each estimated finger label matches at least one of the ground truths; and the recombined match rate (M_{rec}) considers the recombined ground truth that minimises the error of the estimation. Here we propose an additional metric to quantify the amount of hand movement on the piano, which translates into the increased effort from the pianists.

Recently, Guan et al. [8] proposed to use a bidirectional LSTM instead of a unidirectional one. However, they are still modelling the system as a local classification problem, i.e., for a sequence of notes, they only predict the note at the middle of the sequence, as the neural network methods proposed by Nakamura et al. [19]. The Guan et al. research is interesting since they dealt with the difficulty of fingering chords and presented a methodology and metrics that highlight the problem of fingering for polyphonic piano music. However, the performance is below par with Nakamura et al. proposal. In the same line, Zhao et al. [35] proposed a new input representation based on reducing the keyboard to pitch differentials as Nakamura et al. did in the HMM approaches, with additional constraints for the last neural network layer. Although very inspirational, the research paper has not been published yet, and we do not have access to the code.

In addition to the proposed dataset, methods, and metric, we aim at proposing a open evaluation framework towards standardizing the assessment and establishing good evaluation practices for the APF task. We believe that research reproducibility may improve future APF research.

[1] ThumbSet dataset available at: https://doi.org/10.5281/zenodo.6433702
[2] Code, models, and results available at: https://github.com/PRamoneda/Automatic-Piano-Fingering
[3] Evaluation framework available at: https://github.com/PRamoneda/APF_eval

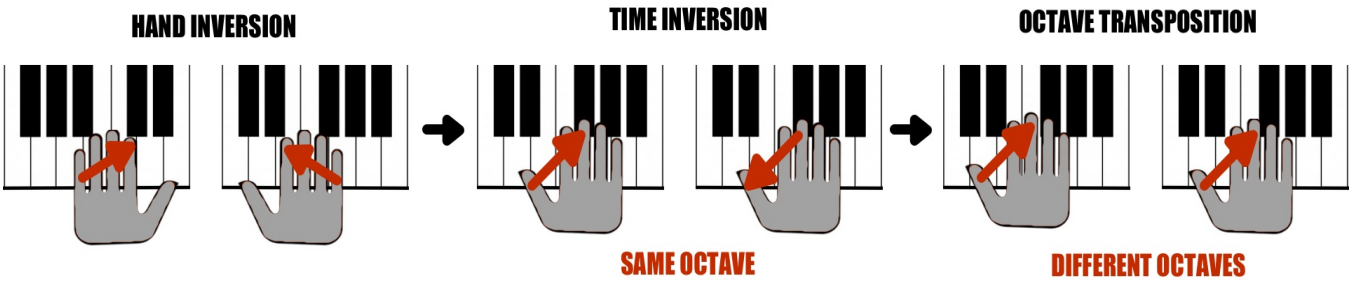

Figure 1: Data augmentation pipeline.

3 THUMBSET DATASET

Here we introduce *ThumbSet* dataset, an open dataset resulting from collecting all the musicXML piano scores published as public domain on the MuseScore website [4] with finger label annotations. We relied on music publishers adding partial or full annotations to support piano learning when creating this dataset. Note that the annotations solely reflect a single editor's expertise and do not represent global ground truth. The source of the annotations on the MuseScore website is not clear. While for some scores, the labels are not correctly engraved, for other scores, the fingers may be provided by non-expert users. Therefore, the dataset may contain a considerable amount of noise data, and the data quality is worse than the *PIG* dataset.

ThumbSet is composed of 2523 music scores as is shown in Table 1. The genres, transcription quality and fingering quality is highly heterogeneous, and the level of difficulty of the pieces might tend to the early years of music education as contrasted to the PIG dataset. However, it is not possible to quantify all these claims with the existing metadata. We can claim there are more finger labels annotated in the right hand (61%) than in the left hand (39%), while there are more pieces with only left annotations (742) than with only right annotations (153). The proportion of annotated fingers and the window's lengths are similar in both hands. Moreover, to make *ThumbSet* available for research purposes [6], we sliced the data in several windows. Each window takes the context, other notes, and symbols, between 32 and 64 notes and symbols around each symbol of interest, including other finger label annotations. In addition, the excerpts are encoded in the *PIG* encoding, a text format proposed in [18] that does not allow reverting to the original score in order to protect the copyright of the pieces. We distribute *ThumbSet* as variable-length music windows in the PIG encoding. The format contains information about pitch, time onset, time offset, and the finger label annotation if it exists for all notes. We limit access to the data upon request under the Zenodo platform. In addition, we distribute the links to all the MuseScore source pieces we have used in creating *ThumbSet*.

	ThumbSet Statistics		
	LH	RH	total
n pieces	2370	1781	2523
n windows	70613	108124	178737
ave length of window	43	44	44
prop of annotated notes	52%	52%	52%

Table 1: Statistics about *ThumbSet* dataset.

4 METHODOLOGY

We propose two autoregressive neural network methods, autoregressive long short-term memory method *ArLSTM* and autoregressive graph neural network method *ArGNN*, to learn general features from *ThumbSet* and perform fine-tuning on the *PIG* dataset. In addition, we structure the models as a sequence-to-sequence model with one-to-one mapping, using an encoder and decoder architecture. The *ArLSTM* encoder uses as input solely a pitch vector, while the *ArGNN* encoder can handle voices and chords simultaneously with graph neural networks. Moreover, considering previous sequence outputs, the autoregressive decoder works in a high-level context.

4.1 Data representation

Piano fingering annotations may come in two different formats, depending on the annotation process. The first format is a set of finger numbers annotated manually on sheet music to suggest a fingering (or fingerings) for a composition. The second format is a list of finger numbers employed by a musician during a performance and are directly derived from the performance. The latter form of fingering data is found in the *PIG* and *ThumbSet*. Given a sequence of T notes presented in a piece, the fingering of the piece outcomes by assigning a finger label y_t for the t-th note. The finger label ranges from 1 to 5, in the right hand and from -1 to -5 in the left hand. We represent the y_t without finger annotation available as 0. The MIDI pitch of each note can be represented by x_t, where t denotes the note index in the piece.

4.2 Data Augmentation

Augmenting training data is a common strategy for increasing the generalisation of DNN-based methods [31]. In this work, we adapt the APF data augmentation from the procedures proposed by Nakamura et al. [19] since that paper did not present data augmentation procedures concerning symmetries of fingering. Nakamura et al. [19] employed physical symmetries for reducing the transition states of the HMM proposed models, and we derive similar procedures to perform data augmentation, depicted in Figure 1.
Hand inversion assumes transforming any sequence of notes between the two hands with inverted physical distances given an

[4] www.musescore.com

infinite keyboard. Consequently, the hand's disposition of fingers is also inverted. To compute the inverted physical distances, we get the number of semitones between each sequence of notes, the direction (ascendent or descendent), and each note's key type (black or white). The inverted sequence has the same number of intervals and physical types but in the opposite direction. Note that to keep the same distribution of physical distances and key types, we compute a grid search, which also guarantees that the new sequence does not surpass the range of the piano keyboard. However, although the hand inversion keeps the exact physical distances between keys and fingers, it does not keep the same melody and harmony relationships, producing other musical content. Note that solely eight sequences of *ThumbSet* do not have any possible augmentations, while every piece of *PIG* has possible augmentations.

Time inversion assumes that the fingering of a sequence of notes is the same when one plays the sequence from the end to the beginning and the opposite. Note that the time inversion property of piano fingering was studied hundred years ago [3]. To compute time inversion, we reverse the sequence of fingers and notes. Therefore, this augmentation can be applied to any piano sequence of notes and fingers.

Octave transposition is based on the fact that a sequence of fingers and notes can be transposed to any piano octave keeping the same fingers without a change in harmony or pitch class. The transposition method is to sum or subtract multiples of 12 to the original sequence of pitches without surpassing the range of the piano keyboard. The keyboard size also limits the number of possible augmentations. The number of feasible augmentations depends on the pitch range of the sequence, allowing a maximum of 7 augmentations.

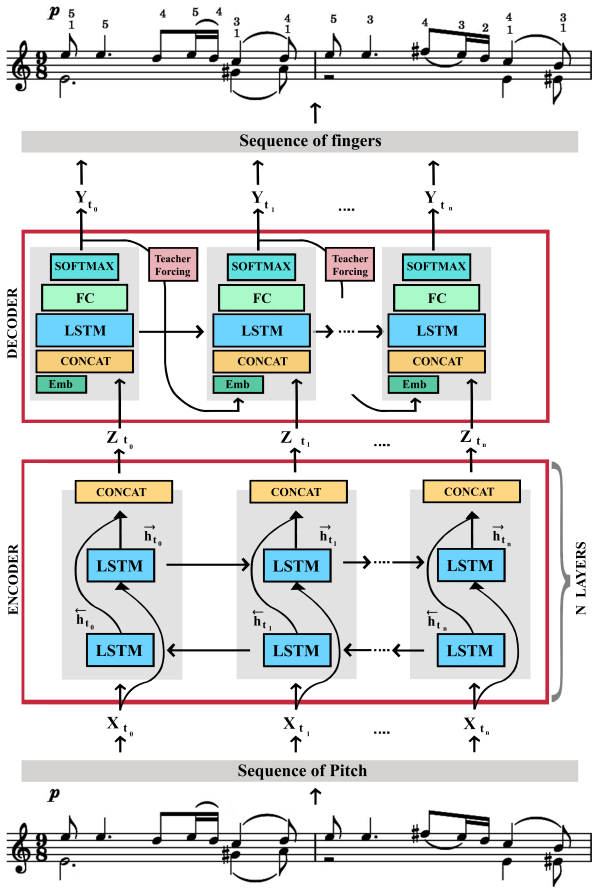

Figure 2: *ArLSTM* encoder-decoder architecture diagram.

4.3 Encoder. Bi-LSTM vs Graph Neural Networks

We propose two methods, *ArLSTM* and *ArGNN*, the former-latter with pitch sequences as input and the latter with a graph of pitches as nodes and the edges distinguishing between chords and single notes. The encoder is the main difference between both methods, as is explained in the following paragraphs. The *ArGNN* encoder is depicted in Figure 3, and it follows a similar architecture to *ArLSTM* depicted in Figure 2. The sole difference is that *ArGNN* replaces the bi-LSTM encoder with a graph neural network encoder. Note that the previous methods considered on this study, *HMM1*, *HMM2*, *HMM3*, *FF-base* and *LSTM-base*, as well as the proposed *ArLSTM* method, have only the sequence of pitches as input and the finger labels as output. Moreover, the proposed *ArGNN* method also considers the onset time of each note to create the edges of the graph.

Autoregressive long-short-term-memory neural network. The autoregressive LSTM is given as input the pitch sequence. Nakamura et al. [19] showed that a high performance can be achieved by using only pitch information. For a fair comparison to [19], we use a similar architecture to theirs. We further adjusted *ArLSTM* encoder adding a bidirectional layer. The directional layer was suggested in Nakamura et al. [19] and Guan et al. [8] for a better feature representation.

Autoregressive graph neural network. The challenge of finding a fingering for polyphonic passages has been addressed in musicology studies [23], and unsuccessfully in the existing APF literature [8, 19]. With the goal of a method to handle polyphonic passages, the encoder is substituted with a graph gated recurrent neural network (GGNN). We employ directional multi-edge-type GGNN [16], to learn note-level hidden representations from an input music score. We use GGNN [16] because of its advantage in learning node-level representations in a graph. The graph layer succeeds in understanding the complex music representation of polyphonic piano scores in previous research [10]. Moreover, it has similar parameters with the *ArLSTM* encoder by adding polyphonic edges information.

Regarding the *ArGNN* input, the score may be represented as a graph $G = (V, E)$, where V and E denote nodes and edges, respectively. A node v corresponds to a single note in the score. An edge e corresponds to a connection between two musically neighbouring notes encoding the polyphonic relationship between two notes of a score. We draw our architectural inspiration from the GGNN related layers. In addition, we employed the simplified representation of Jeong et al. [10], to encode piano scores as graphs. We want to discriminate between chords and single notes, where the nodes

are also the pitch, and the edges mark the polyphonic relationship of the notes. The polyphonic relationship between the notes is marked with two edge types: (onset) encoding notes sounding simultaneous and (next) defining two sequential notes in time. The encoder-decoder *ArGNN* diagram is shown in Figure 3.

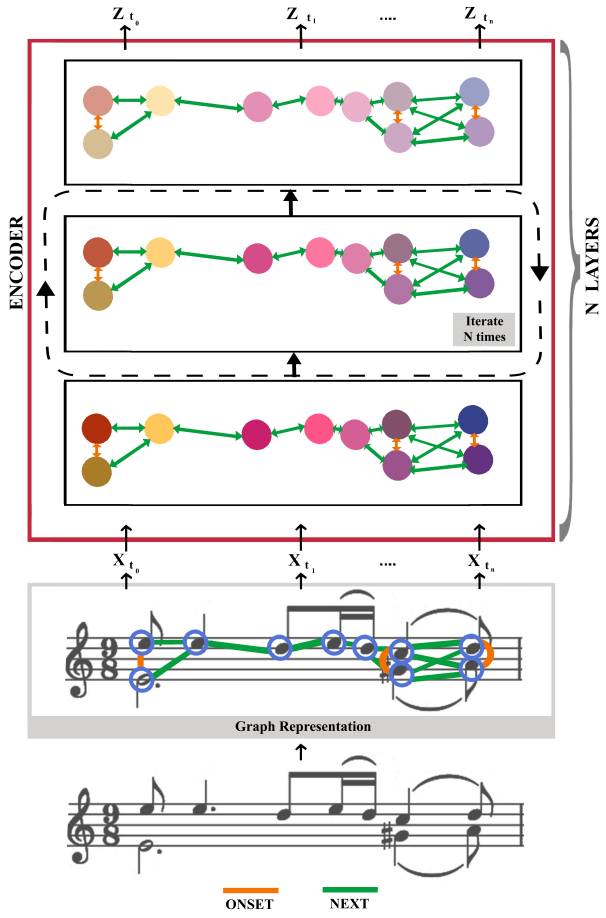

Figure 3: *ArGNN* encoder diagram.

4.4 Decoder. Autoregressive and beam search decoding

One of the important characteristics of piano fingering is that it is necessary to be considered as a progression sequence rather than as a simple piece of note-by-note information because the difficulty of performing a given fingering is decided by considering when to move the hand position or make fingers crossing.

To handle this problem, we propose to adapt autoregressive (AR) decoding, which has been widely used for sequence generation of waveform samples [33] or natural language [1]. AR decoder takes its prediction of the previous time step as an input for predicting the current time step. It has been extensively used in previous research on music-related tasks, such as music transcription [14], performance modelling [11], or music generation [9]. Instead of modelling the probability distribution of y_t as a conditional probability for a given input sequences like $P(y_t|x_0, \ldots, x_T)$, an AR neural network models it as a conditional probability for a given input sequences and a given previous output fingering sequences, like $P(y_t|x_0, \ldots, x_T, y_0, \ldots, y_{t-1})$. In our model, y_t corresponds to the fingering of t-th note while x_t corresponds to the pitch of t-th note. The main difference between AR and non-AR is that AR models predict a given time step while all prior steps' outcomes have already been determined. Therefore, Ar helps the model generate coherent prediction sequence, which is crucial for fingering estimation.

During the training procedure, we have applied teacher-forcing training, which feeds the ground-truth y_{t-1} instead of the predicted result $\hat{y}_{t-1}$ to predict $\hat{y}_t$. The model uses its prediction of previous steps during the inference.

To estimate the most probable fingering sequence for given notes, we employed beam search instead of greedy search. Beam search tracks k candidates of most probable output sequences during the sequence generation. Rather than greedily choosing the most probable next step while building the sequence, beam search extends all possible next steps and retains the most probable k candidates, where k is a user-defined parameter. The probability of a decoded sequence can be represented as $\prod_{t-1}^{N} p(y_t|x_0, \ldots, x_T, y_0, \ldots, y_{t-1})$. The music cognition and perception literature [28] suggest $k = 6$ because a good pianist can retain in memory 5 future notes for correct piano fingering.

Beam search is useful for finding a sequence with a low probability choice in the beginning but eventually composes a sequence with higher accumulated probability in the end. Choosing a less desirable option to achieve a better result later on is also a common tactic adopted by pianists when deciding their fingerings, as one unfavorable move on the preceding note might make succeeding notes much easier to play.

4.5 Fine-tuning and Soft Labels

The labels corresponding to the fingers in the *ThumbSet* dataset (Section 3) are partially annotated. In addition, the source of the annotation is not clear. It may be copied from expired copyright editions or annotated by the user who engraved the score. We observed several errors by exploring the quality of the annotations, although it is difficult to quantify the dataset noisiness. Given this scenario, we assess whether a model trained on *ThumbSet* is generalising by evaluating it in the *PIG* dataset domain with multi-annotated expert fragments. For this purpose, we perform domain adaptation to the *PIG* dataset by fine-tuning the autoregressive models from the *ThumbSet* to the *PIG* training set. By fine-tuning the model into the *PIG* domain, we can guarantee fair evaluation on expert annotations, measure how the model handles multi-annotation, and compare it with previously proposed models.

On the one hand, catastrophic forgetting [13] may degrade all the knowledge acquired in the *ThumbSet* domain. For the sake of simplicity, we decided to explore the freezing of the encoder to retain the knowledge of the first domain in the second. On the other hand, training on data labelled by non-experts may degrade performance because DNNs easily overfit to noisy labels [29]. Maximizing the log-likelihood of the (possibly) correct label encourages

Hand	right					left					both					
Metric	M_{gen}	M_{high}	M_{soft}	M_{rec}	M_{cp}	M_{gen}	M_{high}	M_{soft}	M_{rec}	M_{cp}	M_{gen}	M_{high}	M_{soft}	M_{rec}	M_{cp}	inference
HMM1	58.34	65.08	81.1	72.04	1.09	65.08	70.17	81.1	74.12	1.12	61.77	67.66	81.09	73.15	1.12	0.30 ms
HMM2	60.65	66.72	83.92	75.17	0.98	66.3	71.88	82.98	76.61	1.05	63.78	69.49	83.59	76.12	1.02	0.49 ms
HMM3	60.54	66.86	83.38	74.42	1.05	66.24	71.7	82.41	75.91	1.05	63.63	69.4	83.05	75.42	1.06	0.66 ms
FF-base	56.89	61.77	79.98	66.57	1.17	66.38	71.41	83.33	73.59	1.21	61.44	66.31	81.34	69.82	1.19	73.57 ms
LSTM-base	58.45	63.07	81.58	67.49	1.07	67.51	72.17	84.56	75.26	1.11	62.71	67.34	82.69	71.09	1.07	385.06 ms
ArLSTMThumb-f	60.41	66.91	84.07	74.75	**0.96**	70.69	**77.3**	87.57	**81.36**	**0.99**	65.34	71.73	85.49	77.8	**0.99**	215.78 ms
ArGNNThumb-s	**62.77**	**68.87**	**86.15**	**76.6**	1.01	**70.92**	76.32	**87.58**	80.37	1.01	**66.84**	**72.62**	**86.83**	**78.55**	1.01	161.71 ms
Human	70.2	78.9	91.0	84.5	1	73.1	79.3	90.5	84	1	71.4	79.1	90.8	84.3	1	

Table 2: Results comparison

the model to be confident in its predictions, which might be dangerous in the presence of noisy labels in the training data. Therefore, we propose to experiment with a simple technique, soft labels, for training in the noisy domain [32]. We smooth the label distribution by substituting float soft objectives for the 1 and 0 hard original classification targets, as explained by Szegedy et al. [30]. In addition, previous research has highlighted that noisy labels are helpful in very subjective and multi-label tasks, where many options are feasible [5]. Therefore, the smooth percentage is selected with the prior knowledge of the data with the value of 30%, because the typical agreement between piano fingering annotators is 70% in *PIG* data, assuming the maximum confidence would be that.

5 EXPERIMENTS

Because we aim at leveraging the large size of the *ThumbSet* dataset, we pre-train all methods on this dataset and then we fine-tune the resulting machine learning models on the *PIG* dataset. Note that baselines from Nakamura et al. [19], *HMM-1*, *HMM-2* and *HMM-3*, *LSTM-base*, and *FF-base*, were not designed to work with incomplete annotations. Pre-training them did not improve their performance and we did not include these results. In contrast, the proposed autoregressive methods, *ArLSTM* and *ArGNN*, benefit from *ThumbSet* pre-training and then are fine-tuned on the *PIG* dataset. We denote the corresponding models as *ArLSTMThumb* and *ArGNNThumb*. Moreover, in the case of *ArGNN*, the soft labels improved accuracy on the validation set, while for *ArLSTM*, soft labels led to a decrease in performance. Thus, we report the best performing models on the validation set, *ArLSTMThumb-f* and *ArGNNThumb-s*, where *Thumb* denotes fine-tuning on the *ThumbSet*, *s* the use of soft labels and *f* freezing the encoder on the fine-tuning.

We evaluate the APF systems using the expert labels provided by the *PIG* test dataset. The multi-annotated ground truth allows us to compare the proposed method with the previous work and consider piano individualisation. Due to the quality of annotations, we reserve the novel *ThumbSet* dataset for training. Additionally, we present metrics for left hand (LH) and right hand (RH) separately. We further compare the impact of the autoregressive decoder, beam decoding, data augmentation, soft labels, training dataset and fine-tuning in an ablation study in Section 5.3. Lastly, we analyse two selected examples from a musicology perspective in Section 5.4.

5.1 Experimental Setup

For training the proposed methods, we used the Adam optimizer and a batch size of 64 and a learning rate of $5 \cdot 10^{-5}$. We designed the encoders and decoders of LSTM and GGNN with 32 units and an encoder of three layers, the same parameters used by Nakamura et al. [19] for the *LSTM-base*. We used the negative log likelihood loss as the criterion for the optimizer while masking the outputs without labels in the case of the *ThumbSet* dataset. The hyperparameters of the *HMMs* were trained by grid search on the proposed *PIG* validation set.

We rely on the four metrics, M_{gen}, M_{high}, M_{soft}, and M_{rec}, proposed by Nakamura et al. [19] and reviewed in Section 2, considering the multi-annotated *PIG* dataset. Note that we use a Python wrapper on the original C++ implementation of the evaluation metrics and due to the floating precision the results are slightly different to original evaluation metrics. In addition, we introduce another metric to account for the sequential characteristics of fingering, the change position rate (M_{cp}). There are two position change types in piano fingering: finger-crossing, which use the thumb finger to cross fingers and change the hand position, and shift movement, changing the hand position without crossing the fingers. After counting the number of occurrences of these position changes of the fingerings, M_{cp} results from normalising it with the average number of position changes on the expert annotations. M_{cp} does not need human annotation to compare the result of the models. It uses information from expert fingerings, but just for normalising the value. Moreover, hand change position is an established term in the piano community [4, 15, 22]. It gives importance not only to the fingers but also to the hand. Pianists change the hand's position while playing the instrument depending on the composition requirements. However, pianists tend to minimise the effort needed to play a sequence of notes. Consequently, the fewer hands change position, the less effort is required.

5.2 Results

In Table 2, we display the metrics for the five previous methods and the two proposed methods. The proposed autoregressive models achieve the best performance on every metric evaluated. The results of the previous methods are in a different order from the original ones [19] with *HMM2* as the best baseline instead of the *HMM3*. The difference seems to indicate the importance of decoupling the train and test with the validation set [26]. Moreover, the

change position ratio is slightly better on the *HMM2*. The relative difference on M_{gen} with the best baseline, *HMM2*, is 3.06% in the case of *ArGNNThumb-s* and only 5% of the human agreement. In addition, in the case *ArLSTMThumb-f*, with the same representation as compared previous methods, the difference is $1,74\%$.

The main takeaway from these experiments is that *ThumbSet* noisy dataset allows the training of more complex deep learning models that outperform previous methods. The state-of-the-art performance, not only on the data based metrics, M_{gen}, M_{high}, M_{soft}, and M_{rec}, also in the M_{cp}, indicates that the proposed autoregressive methods learn representations related to the effort reduction of hand movement. Apart from the mentioned results of *ArGNNThumb-s* and *ArLSTMThumb-f* on the evaluation metrics M_{gen} and M_{cp}, the proposed methods outperform in the other metrics. The results of M_{high} for both hands have a similar increment, $3,14\%$ than the M_{gen} comparing the *ArGNNThumb-s* experiment with the *ArLSTMThumb-f* experiment. The *LSTM-base* and *FF-base* experiments level-off on the M_{rec} and increase on the M_{cp}, indicating a worse understanding of sequential coherence. Although the proposed methods outperform in all evaluation metrics the previous methods, the comparison between both is also important. *ArGNNThumb-s* outperform *ArLSTMThumb-f* in all the metrics, and the increment of performance is similar for all the evaluation metrics, which might indicate the difference is only gotten by the polyphonic information provided.

We observed that the previous methods, *FF-base* and *LSTM-base*, without an autoregressive decoder, achieve sub-par performance when pre-trained on the partially annotated data of the *ThumbSet* dataset. Introducing an autoregressive encoder leads to better results. This suggests that the proposed methods, *ArLSTM* and *ArGNN*, are able to leverage more extensive training data. The results are promising since annotating datasets such as *PIG* is expensive, and large score repositories come with incomplete and noisy annotations. However, in terms of computational cost, the inference time is significantly smaller for *HMMs*. We note that *ArLSTM* and *ArGNN* have similar inference time to the two deep learning baselines, *FF-base* and *LSTM-base*. The hardware used to carry out the inference experiment is a machine with 32GB ram and i7-7700 CPU.

We further observe a gap between the models tested on the left hand and right hand, not mentioned in previous research, even though there is a little gap between the left hand and right-hand human agreement. The gap could indicate a difference between the left-hand and the right-hand domains. Further experiments are needed to understand if the difference is linked to the data. We can also see visualising each hand performance separately that the increment of performance of the proposed methods compared with the *HMMs* are mainly based on a better understanding of the left hand, where there is a greater improvement. The greater improvement in the understanding of the left hand than the right hand can be observed in all the evaluation metrics. For example, comparing the M_{gen} in the *ArGNNThumb-s* and *HMM2*, we can see the first has a difference of 8.17% between the right and left hand while the latter only 5.8%. We can not affirm if the gap between the left and right hand is produced because of the increment of the domain with *ThumbSet* or the autoregressive architecture, and further research could answer this question. The drop-off of improvement in the right hand is particularly evident in the *ArLSTMThumb-f* experiment where M_{gen} and M_{rec} are even below par than *HMM3*. At the same time, the M_{cp} results outperform the rest of the experiments. This is so interesting because although the M_{cp} denotes a high understanding of how to play the piano, sometimes, the composition requires other fingers not entirely comfortable for character or style requeriments [22]. Note that the *ArLSTMThumb-f* has a similar performance on the left hand to the *ArGNNThumb-s* on the M_{high}, M_{rec} and M_{cp}, indicating a great sequential coherence. Pretraining the previous state-of-the-art methods, *HMMs*, *FF-base* and *LSTM-base*, on the incomplete annotations of *Thumbset* did not improve their performance, in comparison to not using pretraining. To that extent, the recurrent neural networks had similar performance, and the HMM baselines performed worse.

	M_{gen}	ΔM_{gen}	M_{cp}	ΔM_{cp}
Models proposed				
ArGNNThumb-s	66.84	0.00	1.01	0.00
ArLSTMThumb-f	65.34	0.00	0.99	0.00
Without Ar decoder				
FcThumbGNN-s	63.55	-3.29	1.20	0.17
FcThumbLSTM-f	27.39	-37.95	3.22	2.23
Without beam decoding				
ArGNNThumb-s	64.7	-2.14	1.05	0.04
ArLSTMThumb-f	64.25	-1.09	0.98	-0.01
Without data augmentation				
ArGNNThumb-s	60.36	-6.48	1,06	0.05
ArLSTMThumb-f	64.69	-0.65	1,01	-0.02
Soft labels ablation				
ArGNNThumb	66.48	-0.36	1,03	0.02
ArLSTMThumb-sf	65.32	-0.02	0.98	-0.01
Freeze encoder ablation				
ArGNNThumb-sf	65.99	-0.85	1.06	0.05
ArLSTMThumb	64.01	-1.33	0.95	-0.04
Only PIG training				
ArGNN	60.53	-6.31	1.14	0.13
ArLSTM	26.82	-38.52	2.27	1.28
Only ThumbSet training				
ArGNN-s	65.24	-1.60	1.03	0.02
ArLSTM	64.38	-0.96	0.99	0

Table 3: Ablation study results. Higher is better for M_{gen}, and lower is better for M_{cp}.

5.3 Ablation study

We analyse the influence of different components of the models, *ArLSTMThumb* and *ArGNNThumb-s*, by substituting each component with a simple alternative. The results are displayed in Table 3.

Without an autoregressive decoder. We substitute the decoder with a fully connected network (FC) that adapts the encoder's output size to 5 classes or fingers. As shown in Table 3, the LSTM alternative can not learn to get piano fingering. The *ArGNNThumb-s* without autoregressive decoding learns how to find single notes fingering, achieving a considerable M_{gen}. However, on the M_{cp} increase of 0.17, similar to Nakamura et al. [19] *LSTM-base* and *FF-base*. The M_{cp} increase probably indicates *ArGNNThumb-s* is

not considering the coherence of the sequence, producing several hand change positions artifacts.

Without beam decoding. We replace the beam decoding with a greedy search. In other words, instead of having a beam search of six notes, we select the most likely finger. The decrease of M_{gen} and increase of M_{cp} suggests that beam decoding or forcing the sequential coherence in fragments of six notes increases the APF performance.

Without data augmentation. Not using data augmentation leads to a smaller drop in performance on *ArLSTMThumb-f* than on *ArGNNThumb-s*, supporting the idea that the former model is more straightforward and less data-hungry than the latter. We note that some augmentation may produce sequences of notes that do not exist in the test set, although they are musically feasible.

Soft labels ablation. We train *ArGNNThumb* and *ArLSTMThumb* without soft labels and this leads to a small decrease in M_{gen}.

Freeze encoder ablation. We freeze the parameters of the encoder from *ArGNNThumb-s* while fine-tuning it on *PIG* and we do the opposite from *ArLSTMThumb-f*. This results in a slight difference in M_{gen} and a considerable improvement of M_{cp}. Further experiments are needed to understand why this happens.

Only *PIG* training. We train the *ArLSTMThumb* in the *PIG* domain with below-par results. In the *ArGNNThumb* case, although M_{gen} is higher, M_{cp} is also subpar, indicating it is not possible to learn the autoregressive decoding on small datasets.

Only *ThumbSet* training. The proposed models achieve state-of-the-art results trained solely on the *ThumbSet* dataset. We observe solely a 1.6% and a 0.96% difference in M_{gen} for *ArLSTMThumb* and *ArGNNThumb* respectively. Similarly, the increment in the M_{cp} metric is small. This confirms the fact that it is possible to train high-performance APF systems solely on the noisy dataset.

5.4 Case study

We analyse two selected examples showing the refinements of the proposed methods in Figure 4. The autoregressive excerpt, *Ar example*, shows a right-hand monophonic passage where sequential coherence is needed. In contrast, the polyphonic excerpt, *Poly example*, is a left-hand critical polyphonic passage. For each of the excerpts, Figure 4 shows a comparison between the previous methods (*HMM2* and *LSTM-base*), the proposed methods *ArLSTMThumb-f* and *ArGNNThumb-s*, and the human expert annotations. The human annotations coincide in the *Ar example*, while in the *Poly example*, different alternatives are shown.

The *Ar example* shows that in the previous methods, the *HMMs* were able to better capture sequential coherence than *LSTM-base*, the latter having critical errors as the repetitions of the finger in different consecutive notes (the semiquavers 6th and 7th). The autoregressive finger outputs are the same as the human expert annotations. Nevertheless, we observe that *HMM2* predictions have sequential coherence, although they do not match with the human annotations.

The *Poly example* shows a polyphonic excerpt that requires all notes to be legato, a difficult case for a pianists. To solve this, the pianists in the *PIG* dataset change the fingers in the central repeated chords. The machine learning models, with the exception of *ArGNNThumb-s* model that uses information about the simultaneous sounding notes, represent flat chords as broken chords. In

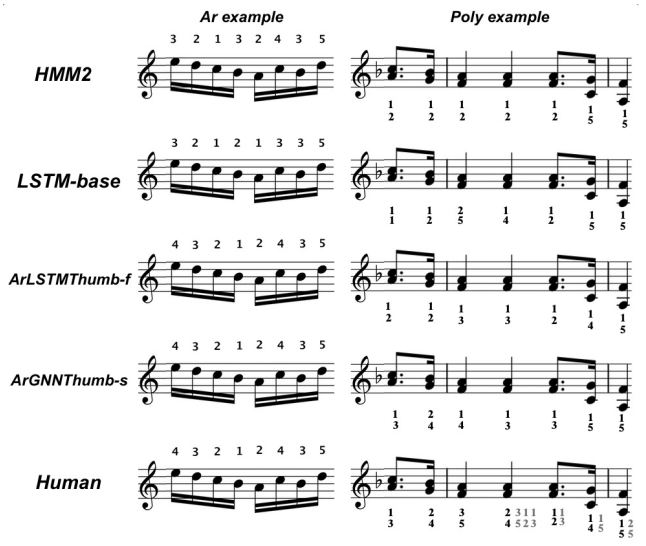

Figure 4: Two examples with piano fingering provided by the compared methods and human expert annotators. The *Ar example* is from *Two-part invention in C major, J. S. Bach* and the *Poly example* comes from *Piano Sonata K 332 in F major, 1st mov., W. A. Mozart.*

many occasions, finding the fingering of a flat chord is the same as finding a broken chord with the same notes. In the opposite case, the *ArGNN* method outperforms the rest methods. In Figure 4, we can see *HMM2* and *ArLSTMThumb-s* consider the chords as broken chords, *LSTM-base* has critical errors in the first half of the excerpt, and *ArGNNThumb-s* handles the polyphonic representation differing solely in the third chord with the human annotators. However, the difference is not a critical error.

6 CONCLUSIONS

In the present work, we propose a methodology to train complex deep learning APF models from partially noisy data, outperforming the current state of the art. Namely, we propose two autoregressive models that learn the sequential coherence of fingering and work towards reducing the hand change position effort, as measured by new proposed metric M_{gen}. In addition to the openly available source code, models, and *ThumbSet* dataset, we also provide an evaluation framework to standardise the evaluation of the task. The novel *ThumbSet* dataset opens the door to future research on automatic piano fingering with data-intensive methods. As future work we plan on exploring more complex techniques for handling noisy labels and domain adaptation.

ACKNOWLEDGMENTS

This work is supported in part by the project Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033 funded by the Spanish Ministerio de Ciencia, Innovacion y Universidades (MCIU) and the Agencia Estatal de Investigacion (AEI), Sogang University Research Grant of 202110035.01 and JSPS KAKENHI Nos. 21K12187 and 22H03661.

REFERENCES

[1] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

[2] Matteo Balliauw, Dorien Herremans, Daniel Palhazi Cuervo, and Kenneth Sörensen. 2017. A variable neighborhood search algorithm to generate piano fingerings for polyphonic sheet music. *International Transactions in Operational Research* 24, 3 (2017), 509–535.

[3] Malwine Bree and Seymour Bernstein. 1997. *The Leschetizky method: A guide to fine and correct piano playing*. Courier Corporation.

[4] Luca Chiantore. 2001. *Historia de la técnica pianística*. Alianza Madrid.

[5] Haytham M Fayek, Margaret Lech, and Lawrence Cavedon. 2016. Modeling subjectiveness in emotion recognition with deep neural networks: Ensembles vs soft labels. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 566–570.

[6] Christophe Geiger, Giancarlo Frosio, and Oleksandr Bulayenko. 2018. The exception for Text and Data Mining (TDM) in the Proposed Directive on Copyright in the Digital Single Market-legal aspects. *Centre for International Intellectual Property Studies (CEIPI) Research Paper* 2018-02 (2018).

[7] Martin Gellrich and Richard Parncutt. 1998. Piano technique and fingering in the eighteenth and nineteenth centuries: Bringing a forgotten method back to life. *British Journal of Music Education* 15, 1 (1998), 5–23.

[8] Hongzhao Guan. 2021. Automatic Piano Fingerings Estimations using Recurrent Neural Networks. In *2nd Nordic Sound and Music Conference, online*.

[9] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer: Generating Music with Long-Term Structure. In *International Conference on Learning Representations*.

[10] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. 2019. Graph neural network for music score data and modeling expressive piano performance. In *International Conference on Machine Learning*. PMLR, 3060–3070.

[11] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. 2019. VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance. In *The 20th International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval Conference (ISMIR), 908–915.

[12] H. Kim, P. Ramoneda, M. Miron, and X. Serra. 2022. An Overview of Automatic Piano Performance Assessment within the Music Education Context. In *Proceedings of the 14th International Conference on Computer Supported Education*.

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.

[14] Taegyun Kwon, Dasaem Jeong, and Juhan Nam. 2020. Polyphonic Piano Transcription Using Autoregressive Multi-State Note Model. In *The 21th International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval.

[15] Denis Levaillant, Carme Poch, and Carles Guinovart. 1986. *Le piano*. Vol. 1. J.-C. Lattès.

[16] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[17] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. 2018. Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 101–105.

[18] Eita Nakamura, Nobutaka Ono, and Shigeki Sagayama. 2014. Merged-Output HMM for Piano Fingering of Both Hands.. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2014)*. Taipei, 531–536.

[19] Eita Nakamura, Yasuyuki Saito, and Kazuyoshi Yoshii. 2020. Statistical learning and estimation of piano fingering. *Information Sciences* 517 (2020), 68–85.

[20] Eita Nakamura and Kazuyoshi Yoshii. 2018. Statistical piano reduction controlling performance difficulty. *APSIPA Transactions on Signal and Information Processing* 7 (2018).

[21] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. 2017. Rhythm transcription of polyphonic piano music based on merged-output HMM for multiple voices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25, 4 (2017), 794–806.

[22] Heinrich Neuhaus. 2008. *The art of piano playing*. Kahn and Averill.

[23] Albert Nieto. 1988. *La digitación pianística*. Fundación Banco Exterior.

[24] Caroline Palmer. 1997. Music performance. *Annual review of psychology* 48, 1 (1997), 115–138.

[25] Richard Parncutt, John A Sloboda, Eric F Clarke, Matti Raekallio, and Peter Desain. 1997. An Ergonomie Model of Keyboard Fingering for Melodic Fragments Sibelius Academy of Music , Helsinski. *Music perception: An Interdisciplinary Journal* 14, 4 (1997), 341–382.

[26] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. In *1st Mathematical Reasoning in General Artificial Intelligence Workshop, ICLR 2021*.

[27] Pedro Ramoneda, Nazif Can Tamer, Vsevolod Eremenko, Marius Miron, and Xavier Serra. 2022. Score difficulty analysis for piano performance education based on fingering. In *ICASSP 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[28] John Sloboda. 1974. The eye-hand span-an approach to the study of sight reading. *Psychology of music* 2, 2 (1974), 4–10.

[29] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[31] Luke Taylor and Geoff Nitschke. 2018. Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1542–1547.

[32] Christian Thiel. 2008. Classification on soft labels is robust against label noise. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 65–73.

[33] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *Computing Research Repository* (2016).

[34] Yuichiro Yonebayashi, Hirokazu Kameoka, and Shigeki Sagayama. 2007. Automatic Decision of Piano Fingering Based on a Hidden Markov Models.. In *IJCAI*, Vol. 7. 2915–2921.

[35] Haoyue Zhao, Xin Guan, and Qiang Li. 2021. Estimation of Playable Piano Fingering by Pitch-difference Fingering Matching Model. http://arxiv.org/abs/2108.09058