A DEEP LEARNING METHOD FOR ENFORCING COHERENCE IN AUTOMATIC CHORD RECOGNITION

Gianluca Micchi **Katerina Kosta** **Gabriele Medeot** **Pierre Chanquion**
ByteDance
{name.surname}@bytedance.net

ABSTRACT

Deep learning approaches to automatic chord recognition of symbolic music have improved the state of the art, but they still face a common problem: how to deal with a vast chord vocabulary. The naive approach of writing one output class for each possible chord is hindered by the combinatorial explosion of the output size (∼10 million classes). We can reduce this complexity by several orders of magnitude by treating each label (e.g. key or chord quality) independently. However, this has been shown to lead to incoherent output labels. To solve this issue we introduce a modified Neural Autoregressive Distribution Estimation (NADE) as the last layer of a Convolutional Recurrent Neural Network. The NADE layer ensures that labels related to the same chord are dependently predicted, and therefore, enforce coherence. The experiments showcase the advantage of the new model both in chord symbol prediction and functional harmonic analysis compared to the model that does not include NADE as well as state-of-the-art models.

1. INTRODUCTION

Harmony, together with counterpoint and form, is traditionally considered to be one of the three main parts of musical composition in Western classical tradition [1]. This tradition is based on what is known as the *tonal system*, that is, a "theoretical formulation of certain psychological or physiological constraints upon the perception of sounds and their combination" [2][p.206]. Their musical effect can be summarised in a few rules that are followed by most Western music (and also some non-Western music) [3].

Nevertheless, harmonic interpretation of music is complex due to its ambiguity. The same audio content can acquire different perceptual significance depending on its context: As a simple example, the chord symbols A♯ major and B♭ major are acoustically indistinguishable but used in different contexts, hence the different spelling. Therefore, it is necessary to study the chord not as single entities but as a progression. This is usually done with the help of the Roman numeral notation (RN), which describes every chord in relation to the local key. RNs provide insights into harmony theory by exposing its invariances and symmetries. They highlight the function of each chord inside the progression and, for this reason, Automatic Chord Recognition (ACR) with RNs is also known as functional harmonic analysis.

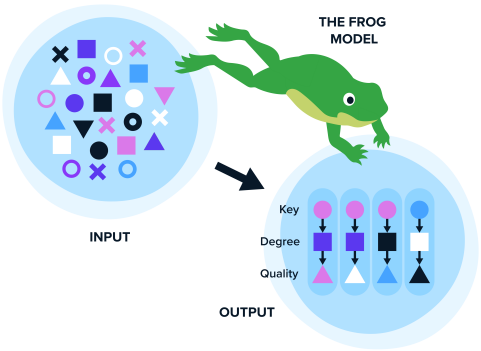

Figure 1: The proposed model, **frog**, analyses an input score in symbolic format and outputs the functional harmonic analysis through autoregressive predictions of its elementary components at each time step.

The problem of harmony has a long academic history, but remains central to modeling and understanding most music, including modern pop; indeed, harmony is one of the main categories in which submissions to the 2020 AI Song Contest were judged. [4]. Therefore, it is natural that computational analysis of harmony has attracted so much attention in the MIR community.

Previous work. There is a relatively vast body of work on ACR from audio signals (see [5] for a literature review on the topic). All these methods address chord symbol recognition instead of functional harmonic analysis. From the very first article published on the subject, the idea that has dominated the field is to interpret the audio signal using chroma features [6]. This amounts to identifying and annotating the pitch content of the audio signal at each given time frame. Given how close the resulting audio representation resembles a symbolic music score, it is a bit puzzling to see how little attention symbolic ACR has received.

There are only a few works, to our knowledge, that explicitly perform ACR on symbolic scores. Kröger et al. [7] collect a few early approaches, and a more recent one is

presented in [8]. However the interest on the topic has increased in more recent years, probably driven also by the growing attention to symbolic music for music generation [4, 9–13]. Symbolic music makes a perfect entry point for the harder task of functional harmonic analysis because, with respect to audio data, it offers a much more direct representation of the musical content. There are two popular MIR approaches to functional harmonic analysis: One uses generative grammars [14, 15], the other a deep learning based data-driven approach that can learn rules more flexibly [16–19].

Known issues in ACR. One difficult issue that prevents naive rule-based approaches from being successful is the identification of non-chord tones: Music often contains notes that are not part of the core harmony and designing a system that knows when to ignore these anomalies is a complex task to define algorithmically [20]. Specifically for the task of functional harmonic analysis, there is also the problem of automatic key recognition, a subject that is well known in the community [21–24] but still largely unsolved, partly due to ambiguities in its definition [24].

Finally, an important issue is that the number of possible output classes is very large. Even considering only the most common chords, the possibilities easily exceed 100,000. This is because there is a combinatorial explosion due to the presence of several elementary labels associated with each possible chord. In the case of functional harmonic analysis, these labels are key, tonicisation, degree, quality, and inversion. For chord symbol prediction, instead, they are root, quality, and inversion. To reduce the dimensionality of the output space, it has been proposed to predict each label independently [16–18]. However, this approach often leads to incoherent output labels. For example, for a harmony that can be interpreted as either A minor or C major, a system without coherence could equally well output A *major* or C *minor*.

Our proposal. We propose a chord recognition algorithm (CRA) —which we nickname **frog** after the sound they make in Italian, *cra-cra*— that analyses chords through separate but coherent predictions of their elementary labels (see Fig. 1). This is achieved through the addition of a Neural Autoregressive Distribution Estimator (NADE) [25–27] to a CRNN architecture. At each time step, we provide the NADE with a fixed ordering of the chord's elementary labels from which it sequentially samples, conditioning each sampling probability on the outcome of the previous labels. We release the code open-source and make it available at [28].

We evaluate the output of **frog** against two state-of-the-art models on a large dataset of functional harmonic analyses, containing over 300 scores that span over two centuries, from Monteverdi to Brahms. Due to the similarity in data representation, mentioned above, we believe our system to be extensible to the audio domain [17, 19].

2. DATA REPRESENTATION AND CORPUS

In RN notation, each chord is defined by its relation with the tonic of the local key. The basic components of RNs

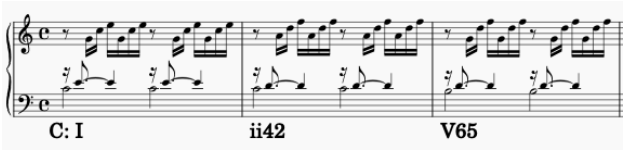

Figure 2: Example RN analysis of the Prelude in C from JS Bach's Well-Tempered Clavier, book first.

are key, degree of the scale on which the chord is built (expressed in Roman numerals), quality of the chord (i.e., the type of triad plus any possible extension), and inversion (i.e., which of the notes is the lowest). For example, from the RN analysis in Fig.2 we see the annotation V65 at the third measure. In the key of **C** (see first measure), this corresponds to a G (fifth degree of the scale) dominant seventh chord in first inversion (numerals 65). [1]

Sometimes, chords are borrowed from other keys for a very short period of time and introduce some colouring in the harmonic progression. For example a D7 chord contains an F♯. Whenever we find such a chord in the key of **C** resolving to a G chord we identify it as a dominant chord borrowed from the neighbouring key of **G** and encode it as V7/V. Those borrowed chords are known as tonicised chords, and the tonicisation defines the relation between the local key and the temporary tonic [24]. The boundaries of this relation are sometimes blurry. As Kostka and Payne put it, "The line between modulation and tonicization is not clearly defined in tonal music, nor is it meant to be" [30].

The tonicisation completes the set of elementary labels that we use to describe a chord in the RN notation.

RN encoding. The simplest data encoding for RN requires 24 keys, 7 degrees, 7 tonicisations, and 4 inversions per each quality of chord. In our analyses we use 10 chord qualities: 4 triads (major, minor, diminished, and augmented), 5 sevenths (major, dominant, minor, half-diminished, and diminished), and augmented sixth.

When predicting all these labels at once, their sizes multiply to make a total of 47k possible output classes. If one wants to add pitch spelling, support for alterations both in degree and in tonicisation, and a direct prediction of the root, the total number of combinations climbs up to 22 millions [18]. Also, while the ten qualities cover most of the cases in classical music, making up for 99.98% of the dataset we consider, they don't even come close to describing the wealth of extensions and colourings that are commonly used in jazz music [33]. In short, it is not desirable to deal directly with such a combinatorially explosive situation. Making individual predictions for each of the elementary labels that form the chord and then combining them together, instead, results in a summation of their output sizes, rather than a multiplication, making the problem tractable again.

Chord symbols. From the RN notation it is possible to derive chord symbols. Those are defined only by root, quality, and inversion. For example, a V65 in C major in

[1] The details of RN notation are complex and out of the scope of this paper. See [29] for an introduction to the syntax.

Dataset	Composer	Content	Crotchets	Annotations
Roman Text [29]	C Monteverdi	48 Madrigals	15,040	5,828
	JS Bach	24 Preludes	3,168	2,107
[31]	FJ Haydn	24 String Quartets Movements	9,113	4,815
	Various	156 Romantic Songs	22,254	11,851
	Various	4 Additional Compositions	2,649	1,165
BPS-FH [17]	Lv Beethoven	32 Sonata Movements	23,554	8,615
TAVERN [32]	WA Mozart	10 theme and variations	7,833	3,887
	Lv Beethoven	17 theme and variations	12,840	6,836
Total		315 scores	96,450	45,104

Table 1: The datasets included in our training / validation data. TAVERN has two alternative annotations: We have only included the anonymous annotator A.

RN notation would be written in chord symbols, following the same encoding as *mir_eval* [34, 35], as *G:7/3*. All information about local key and tonicisation is lost.

Datasets. In recent years, several datasets of RN annotations have been published. Up to our knowledge, they have all been collected and converted to the "rntxt" data format [29] inside the GitHub repository in [36]. We report the size and content of the corpora we used in Table 1.[2] We provide the dataset parsing code, including fixes on known issues, in [28].

3. NADE FOR HARMONIC ANALYSIS

Given a musical context, that is, the portion of score between t_0 and t_1, let us focus on the prediction of a single chord at time $t \in [t_0, t_1]$. As we have seen, a chord can be separated in several labels. This means that we can represent the output class of the chord as a variable in a multi-dimensional space. If those dimensions were all independent, one could project the distribution on each axis and independently estimate each projection of the distribution. This is the case, for example, of a rectangular uniform distribution in a 2D space, which can be written as a product of two independent uniform distributions: $p(x, y) = p_x(x)p_y(y)$. But if the distribution is more complex this is no longer true. What one can always do without loss of generality is to determine an ordering of the dimensions and estimate their value sequentially, *conditioning each dimension given the result of all the preceding ones.* This approach is at the heart of the Neural Autoregressive Distribution Estimator, or NADE [25–27].

Introduction to the NADE. The NADE is composed of two parts: a visible layer –which is made of as many neurons as there are dimensions in the distribution that we want to encode– and a hidden layer. At each step, the content of the hidden layer is used to determine the value of the next neuron of the visible layer. The output sampled from the newly-updated neuron is then reinjected into the hidden layer to inform the decision on the next step. The equations are the following [27]:

$$p(x_d|\boldsymbol{x}_{<d}) = \text{sigmoid}(\boldsymbol{V}_d \cdot \boldsymbol{h}_d + b_d), \quad (1)$$

$$\boldsymbol{h}_d = \text{sigmoid}(\boldsymbol{W}_{<d} \cdot \boldsymbol{x}_{<d} + \boldsymbol{c}), \quad (2)$$

where x_d is the output at dimension d, $\boldsymbol{x}_{<d}$ is the vector of all the outputs before d, $\boldsymbol{V}$ and $\boldsymbol{W}$ are respectively the tensor of hidden-to-visible and visible-to-hidden weights, $\boldsymbol{b}$ is the vector of biases in the visible layer and $\boldsymbol{c}$ in the hidden layer. Eqs. 1 and 2 are to be applied iteratively for all neurons in the visible layer.

Application of NADE to chords. NADE has been applied to music generation [10, 38], with the visible layer representing a frame of the piano roll. In the case of harmonic analysis, the visible layer represents instead a chord annotation. We separate the annotation along six dimensions (see Sec. 2) and organise them in the following order: key, tonicisation, degree, quality, inversion, and root.[3] Every element in this list is conditioned on all the elements that appear before it.

This situation is slightly different from the one in the original NADE formulation since the output is no longer a collection of binary units, but of categorical units with a variable number of classes. The same mechanism of NADE still works, but we have to make a few modifications to it: First, a softmax layer is applied instead of a sigmoid to Eq. 1. Then, to adapt to this change in the output size, the weight tensors $\boldsymbol{V}_d$, which was understood to be unidimensional and of size n_h in the original work, is instead two-dimensional and of size (n_d, n_h). Similarly, the shape of $\boldsymbol{W}_{<d}$ is $(n_h, \sum_{i<d} n_i)$ instead of $(n_h, d-1)$.

It is worth emphasising that, in this case, the NADE is used to autoregressively model the distribution of the output *on* the different dimensions of the chord and *at a specific instant of time* t.

The final frog model. So far, we only explained how to introduce correlation in the outputs but not how to derive those outputs from the inputs. Inspired by [38], we do so with the help of the biases, that we define as

$$\boldsymbol{b} = \text{sigmoid}(\boldsymbol{\theta}_v \cdot \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\beta}_v), \quad (3)$$

$$\boldsymbol{c} = \text{sigmoid}(\boldsymbol{\theta}_h \cdot \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\beta}_h).$$

Here $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are the weights and biases of a dense layer connecting an arbitrary function of the inputs $\boldsymbol{f}$ with the NADE biases.

The function $\boldsymbol{f}$ that we choose is a CRNN since it has already been proven to work well in this domain

[2] Due to the restrictive licence under which it is released, we decided not to include the ABC dataset [37] into our training data.

[3] The ordering of the output labels has been suggested by music theory, and namely by the fact that the degree of the Roman numeral directly depends on the key. For example, the same chord of G major could have degree I in the key of G or IV in D. Keys are a much broader structure than single chords and they change more slowly, therefore we decided to prioritise them in order to avoid an excessive amount of modulations in the output. Similarly, the degree depends on the tonicisation, since this latter expresses the temporary key from which the music borrows.

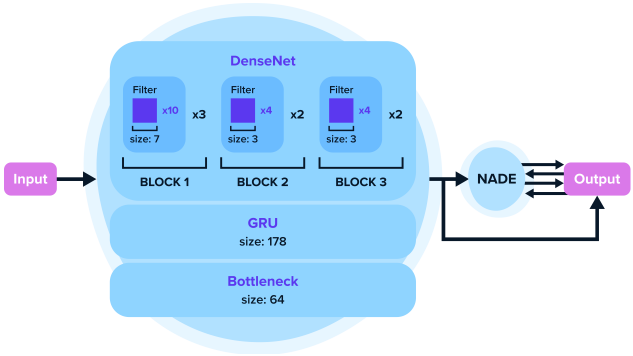

Figure 3: Schematic representation of the model structure.

[16, 18]. In particular, we follow Micchi et al. [18] and use DenseNet [39] for the convolutional part and a bi-directional GRU [40] for the recurrent part, which takes care of modelling the autoregressive part of the calculations in the time domain. A fully connected layer is introduced as a bottleneck in between the GRU and the NADE. We represent the final model, **frog**, in Fig. 3.

The hyper-parameters of **frog** have been selected with the help of hyperopt [41]. In particular, the DenseNet is made of three blocks: The first block has three convolutional layers, each made of 10 filters of size 7; the other two blocks are identical, each with 2 layers made of 4 filters of size 3. The GRU has 178 hidden neurons and is trained with a dropout of 0.2, while the bottleneck layer is made of 64 neurons. Finally, the hidden layer of the NADE is made of 350 neurons. The training is done with an ADAM optimiser with a learning rate of 0.003.

4. EXPERIMENTAL SETUP

We train our models on the task of functional harmonic analysis on symbolic scores. The input is a symbolic file (such as MusicXML, MIDI and **kern) and the output is an aligned harmonic analysis. We tested **frog** against two state-of-the-art models: the original CRNN architecture [18] that we used as a basis for our model and the improved Harmony Transformer model (HT*) [17, 19]. Our proposed model, **frog**, has in total 389k trainable weights, while the HT* has 750k and the original CRNN architecture only 83k. The size difference between **frog** and CRNN is partially due to **frog** including NADE (93k weights), as well as the increase of the GRU's size to 251k weights for **frog**. A CRNN model with the same GRU hyper-parameters as **frog**, only marginally improved the outcome compared to the original CRNN model, so the former was disregarded.

All the trainings use early stopping and typically require less than 20 epochs. The entire training of **frog** lasts for a little more than 2 hours on a recent laptop (no GPU needed). The loss function is the sum of all the categorical cross entropies applied separately to each output. Each individual collection in the dataset is split 80/20 between training and validation data.

4.1 Data encoding

Pitch. For CRNN and **frog**, we have implemented two different representations of the input data: "pitch class+bass" and "pitch spelling+bass". Pitch class+bass contains 24 elements, 12 indicating all the active pitch classes (multi-hot encoded) and 12 indicating the lowest active pitch class —the bass (one-hot encoded). If pitch class+bass is used, the output labels root and key are also encoded using only pitch classes, therefore having respectively size 12 and 24 (the keys can be major or minor).

Pitch spelling+bass, instead, contains 35 elements, that is, the seven notes times five alterations (double flats, flats, diatonic, sharps, double sharps). When pitch spelling+bass is used, the output label root has shape 35 and keys 36 — this is obtained keeping the 18 keys between C♭ and A♯ in the circle of fifths [4] in two modes, major and minor.

Meter. We tested whether or not the addition of metrical information has a positive impact on the outcome. In models that are trained with metrical information (tagged with "w/ meter" label in Section 5), the input includes two additional vectors. The first one-dimensional vector is 1 whenever a new measure begins and 0 otherwise, the second one-dimensional vector is 1 at the onset of a new beat and 0 otherwise.

Time. The input data is quantised in time frames of the length of a demisemiquaver (1/32nd note). Due to the presence of pooling layers in the convolutional part, the output resolution is reduced and corresponds to the quaver (1/8th note).

4.2 Comparison with HT* inputs and outputs

HT* has a slightly different approach. In the original paper, the authors present two separate HT* models. In both cases, the input is encoded in MIDI numbers following a piano roll representation and additionally contains information of the tonal centroids [42].

The first model is trained for functional harmonic analysis and has two outputs: the key (24 categories = 12 pitch classes × 2 modes) and the RN annotations (5,040 categories = 9 tonicisations × 14 degrees × 10 qualities × 4 inversions). We use these RN predictions to derive the root of the chord and therefore its chord symbol representation.

The other model is trained only for chord symbol recognition and has a single output with 25 possible categories: major and minor triads (possibly with extensions) for all the 12 pitch classes and a last category for all remaining chords. We decided not to include the latter model in our experiments because the output vocabulary is too small to be fairly compared with the other models. Such a variant would be comparable to the models we train only in case it contained the same roots, qualities, and inversions as the others, for a total of 480 output classes. Moreover, such chord symbol-oriented HT* can not produce predictions for functional harmonic analysis because of the absence of key, tonicisation, and degree.

[4] We do not keep all keys from F𝄫 to B𝄪 because most of those keys are never used in practice and also because they would require triple flats and sharps to encode all the diatonic pitches.

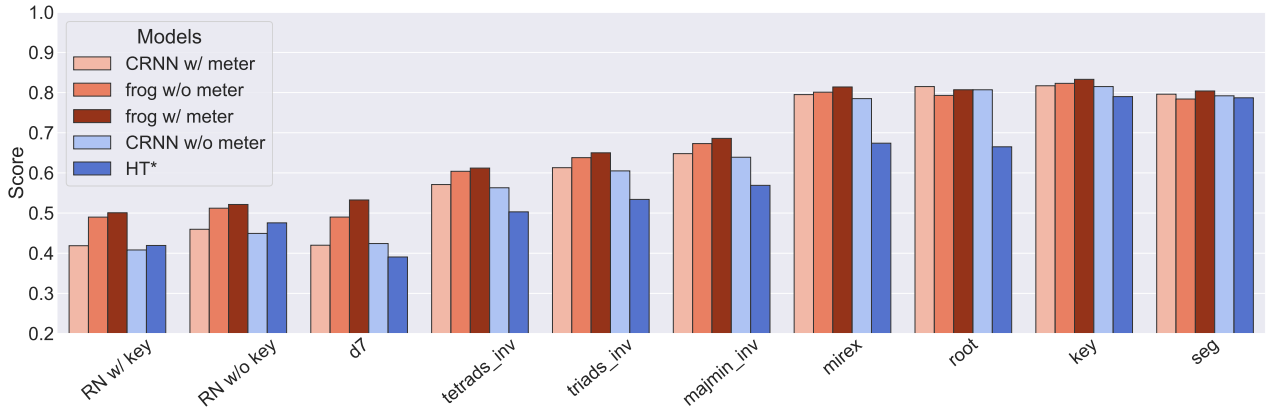

Figure 4: The score of all tested models on selected metrics. The models in shades of red (the first three) are our contributions, the blue ones (last two) are from the state-of-the-art. From left to right: the first three metrics report the ratio of frames with correct predictions to the total number of frames for the written tasks. The remaining seven report the results derived from the specific *mir_eval* tasks.

5. RESULTS

Evaluating the quality of a functional harmonic analysis is an extremely challenging task. First, there could be different analysis of the same music that are equally acceptable [18, 32] — this is a complex issue that might require a complete rethinking of our training strategies and we do not address it in this paper. Second, not all errors are equally important: one could argue that correctly identifying the inversion is less important than the root or the quality of the chord. To address this second issue, we report the scores on several metrics and let the readers decide which one is the most important for their task.

5.1 Analysing the metrics

Remarkably, **frog** shows better results when compared to the previous state-of-the-art models (CRNN w/o meter and HT*) in almost every metric considered (see Fig. 4). Notice that, to provide a better comparison with the HT* model, we report the results of the *pitch class + bass* input data representation (see Sec. 4).

Accuracy on Roman numerals. The most complete metric that we show is the accuracy on RNs. (see Fig. 4, first two metrics from the left). We present two versions: in the first ("RN w/o key"), the prediction is considered correct if and only if tonicisation, degree, quality, and inversion are all correct — this corresponds to the direct RN output of the HT* model (see Sec. 4). For this task, **frog** reports a 52.1% accuracy against the 47.6% that we obtained for HT* and 44.9% for CRNN (w/o meter, understood from now on). The new XL-CRNN achieves 47.1%. [5]

The second case ("RN w/ key") requires also a correct prediction of the key. Here, **frog** still gives a correct prediction in 50.1% of cases against 41.9% that we obtained for HT* and 40.8% for CRNN (43.1% on XL-CRNN). The absolute margin of improvement of **frog** on the best competing state-of-the-art algorithms goes from 4.5% on RN w/o key to 8.2% on the more complex task of RN w/ key.

The case of the diminished sevenths. Diminished seventh are a special chord in music theory because they divide the octave in 4 equal intervals. Therefore, these highly symmetrical chords are often used during modulations. This makes them very easy preys to problems of misclassification due to the lack of coherence. In addition, they are sporadic chords, making up 4.3% of our dataset, which makes correct predictions both difficult and important. The accuracy with **frog** makes a big leap from 39.1% of the HT* model and 42.4% of CRNN to 53.3%, showing a better than average result on these chords (See Fig.4, metric "d7").

The scores on *mir_eval* metrics. We then report a selection of the metrics included in the package *mir_eval* [35] (see Fig. 4, last seven metrics to the right).

The first conclusion we can draw from these results is that the HT*, which chooses its output among a large vocabulary of more than 5000 output classes, has the lowest accuracy of all systems. The more powerful variant of the ACR-oriented version of HT* that we mentioned in Sec. 4 would however probably obtain higher scores than this general-purpose HT* on these metrics.

The second conclusion is that all models perform almost equally on segmentation. The segmentation is reported as the minimum of the score on over-segmentation and under-segmentation and for all models the minimum score is given by the over-segmentation. This could be due either to an intrinsic limitation that is common to all architectures and that needs yet to be discovered; it could be also due to the fact that human annotators might prefer a more synthetic analysis: for example, some notes could be interpreted as passing tones by humans and considered instead as structural part of the chord by the algorithm.

The root coherence. As we saw in Sec. 2, it is possible to derive the root of a chord from its key, tonicisation,

[5] The HT* results we report show a significantly higher accuracy than the 41.7% reported in [19]. We assume that this is due to the larger size of the dataset we train all three models on.

and degree. The CRNN and **frog** models predict an additional redundant output, the root, with the assumption that it helps the systems learn faster. Comparing the root derived from the RN with the one directly estimated by the model we can obtain a measure of the internal coherence of the output labels. CRNN has a root coherence of 78.9%, compared to **frog** which has a root coherence of 99.0%.

Adding metrical information. We notice that the introduction of metrical information (cf. Section 4) has a positive but quite small impact on the results in all metrics.

5.2 Analysing the confusion matrix

In Fig. 5 we report the confusion matrix for the key obtained with **frog** when trained with pitch spelling. The keys are arranged according to the circle of fifths (F-C-G-D...) with the major keys preceding the minor keys, i.e., the top-left quadrant shows the major/major correlation while the bottom-right the minor/minor correlation. The values reported are the occurrences of each pair ground-truth/prediction and are presented in a logarithmic scale to enhance the different scales in prediction errors.

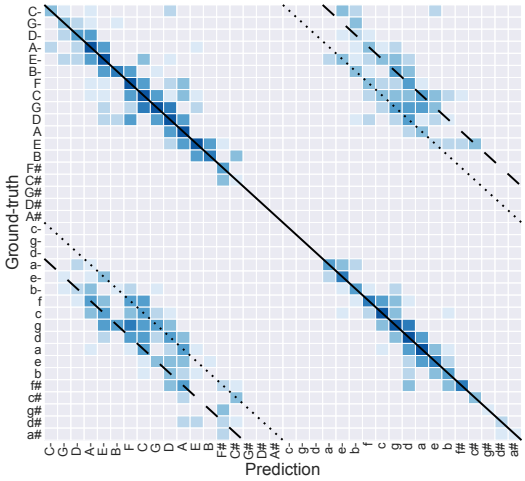

Figure 5: Confusion matrix for **frog** on label *key*.

The *mir_eval* key metric that we reported in Fig. 4 assigns 1 point to all keys that are correctly predicted, 0.5 points to dominant/sub-dominant predictions (**G/F** instead of **C**), 0.3 to relative (**a** instead of **C** or vice versa), and 0.2 to parallel (**c** instead of **C** or vice versa). Those cases are the ones reported on the five diagonals super-imposed to the plot: The main diagonal, in solid line, shows the correctly predicted keys. Dominant predictions are immediately to its right and sub-dominant to its left. Dashed lines show the relative predictions, while dotted lines show the parallel predictions. Some rows and columns of the confusion matrix are empty: These are the keys that are supported by the model but never used in the test dataset.

5.3 Using a key oracle

In a separate but related experiment, we have allowed **frog** to access a key oracle. We did that by reading the key from the test data and setting it as the first output of the visible layer of the NADE. Then, we sampled the remaining labels autoregressively in the given order, as usual.

We measured the impact of this key oracle on the results. Without a dedicated retraining, a multi-output model with no coherence between the different labels, such as the HT* or the CRNN, would report unvaried accuracies for all elementary labels except key. This entails that the accuracy for the RN w/ key prediction be equivalent to the one for RN w/o key. However, this is not what happens with **frog**: The degree accuracy goes from 72.6% to 80.3% and the tonicisation from 91.4% to 94.0%.[6] As a result, the accuracy on RN w/ key jumps to 60.3%, much higher than the 52.1% we would expect in absence of coherence.

6. CONCLUSIONS AND PERSPECTIVES

We report an advancement in the field of automatic chord recognition and especially functional harmonic analysis for symbolic music. The improvements are mostly due to the use of a modified version of the NADE algorithm, which allows us to separate the complex and large vocabulary of all possible output classes into a set of elementary labels (such as key, degree, and quality of the chords) while retaining strong coherence between them. This effectively reduces the size of the output classes by several orders of magnitude and at the same time offers better results, as we showed in Sec. 5.

A consequence of the reduction in complexity of the output labels is the increased flexibility that this model gives to the users, as changes to the chord labels do not dramatically alter the size of the model nor the complexity of the task. For example, one could easily introduce a larger amount of chord colourings, which makes **frog** a better candidate for analysing music such as jazz.

A lot still remains to explore on the details of this approach. We kept the six output labels that had already been presented in previous articles [16–18], where they had been used in absence of the coherence-enforcing NADE layer. Now, we expect to be able to improve the quality of the output even further by tweaking these six labels. For example, one could study the key by separating tonic and mode (major / minor); and the degrees could be separated on two axis: the position on the scale and the alteration.

Concerning the input representation, there are at least three strains of research that caught our interest: relative music representation [43, 44], use of Tonnetz for better convolutions in pitch spaces [45], and better representations for the metrical strength [46].

Another aspect to test is the introduction of OrderlessNADE [26]. The OrderlessNADE effectively trains one separate model for all the possible orderings and then averages the results obtained. This approach could improve the quality of the model both directly (because it demonstrates to be intrinsically superior to our ordered model) and indirectly (because it allows us to find a better ordering than the one we proposed).

[6] The remaining three labels —inversion, quality, and root— are not impacted, which is consistent with the fact that they are independent from the key from a music theory point of view.

7. ACKNOWLEDGEMENTS

We thank Raluca Semenescu for creating the main illustrations in the paper and Jordan B. L. Smith for his extensive paper review before submission.

8. REFERENCES

[1] A. Schoenberg, *Harmonielehre*. Universal Edition, 1922.

[2] N. Cook *et al.*, *Music, imagination, and culture*. Oxford University Press, 1990.

[3] D. Tymoczko, *A geometry of music: Harmony and counterpoint in the extended common practice*. Oxford University Press, 2010.

[4] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. Cai, "Human-AI Co-creation in Songwriting," in *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2020*, 2020.

[5] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 Years of Automatic Chord Recognition From Audio," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, nov 2019, pp. 54–63. [Online]. Available: https://qmro.qmul.ac.uk/xmlui/handle/123456789/62088

[6] T. Fujishima, "Real-time chord recognition of musical sound: A system using common lisp music," pp. 464–467, 1999.

[7] P. Kröger, A. Passos, M. Sampaio, and G. De Cidra, "Rameau: A system for automatic harmonic analysis," in *In Proceedings of ICMC*, 2008.

[8] K. Masada and R. C. Bunescu, "Chord recognition in symbolic music: A segmental crf model, segment-level features, and comparative evaluations on classical and popular music," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 1–13, 2019.

[9] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *International Conference on Machine Learning*, 2017, pp. 1362–1371.

[10] C. Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, 2017, pp. 211–218. [Online]. Available: https://coconets.github.io/

[11] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[12] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International Conference on Machine Learning*, 2018, pp. 4364–4373.

[13] M. Dinculescu, J. Engel, and A. Roberts, "Midime: Personalizing a musicvae model with user data," in *Workshop on Machine Learning for Creativity and Design, NeurIPS*, 2019.

[14] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.

[15] D. Harasim, M. Rohrmeier, and T. J. O'Donnell, "A generalized parsing framework for generative models of harmonic syntax." in *ISMIR*, 2018, pp. 152–159.

[16] T.-P. Chen and L. Su, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 90–97. [Online]. Available: https://github.com/

[17] T. P. Chen and L. Su, "Harmony transformer: Incorporating chord segmentation into harmony recognition," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019, pp. 259–267. [Online]. Available: https://aclweb.org/aclwiki/POS_

[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*, vol. 3, no. 1, pp. 42–54, may 2020. [Online]. Available: http://transactions.ismir.net/articles/10.5334/tismir.45/

[19] T.-P. Chen and L. Su, "Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–13, feb 2021. [Online]. Available: http://transactions.ismir.net/articles/10.5334/tismir.65/

[20] Y. Ju, N. Condit-Schultz, C. Arthur, and I. Fujinaga, "Non-chord tone identification using deep neural networks," in *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, 2017, pp. 13–16.

[21] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[22] S. T. Madsen, G. Widmer, and J. Kepler, "Key-finding with interval profiles," in *In Proceedings of ICMC*, 2007.

[23] N. Nápoles López, C. Arthur, and I. Fujinaga, "Key-finding based on a hidden markov model and key profiles," in *6th International Conference on Digital Libraries for Musicology*, 2019, pp. 33–37.

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

[25] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," *Journal of Machine Learning Research*, vol. 15, pp. 38–39, 2011.

[26] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 719–727, 10 2014. [Online]. Available: http://arxiv.org/abs/1310.1757

[27] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *Journal of Machine Learning Research*, vol. 17, pp. 1–37, 2016.

[28] G. Micchi, accessed 2021-07-29. [Online]. Available: https://gitlab.com/algomus.fr/functional-harmony

[29] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, "The romantext format: A flexible and standard method for representing roman numeral analyses," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019, pp. 123–129.

[30] S. Kostka and D. Payne, *Tonal harmony*. McGraw-Hill Higher Education, 2013.

[31] N. Nápoles López, "Automatic harmonic analysis of classical string quartets from symbolic score," Master's thesis, Universitat Pompeu Fabra, 2017.

[32] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, "Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, 2015, pp. 728–734.

[33] T.-P. Chen, S. Fukayama, M. Goto, and L. Su, "Chord jazzification: Learning jazz interpretations of chord symbols," in *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2020*, 2020.

[34] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez, "Symbolic representation of musical chords: A proposed syntax for text annotations." in *ISMIR*, vol. 5, 2005, pp. 66–71.

[35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A Transparent Implementation of Common MIR Metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 367–372.

[36] M. Gotham, accessed 2021-07-29. [Online]. Available: https://github.com/MarkGotham/When-in-Rome

[37] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdigh.2018.00016

[38] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 2, 6 2012, pp. 1159–1166. [Online]. Available: http://arxiv.org/abs/1206.6392

[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 2261–2269. [Online]. Available: https://github.com/liuzhuang13/DenseNet.

[40] 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 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[41] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *30th International Conference on Machine Learning, ICML 2013*, vol. 28, 2013, pp. 115–123.

[42] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proceedings of the ACM International Multimedia Conference and Exhibition*, 2006, pp. 21–26.

[43] S. Lattner, M. Grachten, and G. Widmer, "Learning transposition-invariant interval features from symbolic music and audio," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 661–667. [Online]. Available: https://github.com/SonyCSLParis/cgae-invar

[44] ——, "A predictive model for music based on learned interval representations," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 26–33.

[45] R. Abecidan, M. Giraud, and G. Micchi, "Towards custom dilated convolutions on pitch spaces," in *International Society for Music Information Retrieval Conference (ISMIR 2020), Late-Breaking Demo Session*, 2020.

[46] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, and K. Webster, "Structurenet: Inducing structure in generated melodies." in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 725–731.