# TOWARDS INTERPRETABLE POLYPHONIC TRANSCRIPTION WITH INVERTIBLE NEURAL NETWORKS

**Rainer Kelz**[1], **Gerhard Widmer**[1,2]
Austrian Research Institute for Artificial Intelligence (OFAI), Austria
Institute of Computational Perception, Johannes Kepler University Linz, Austria
rainer.kelz@ofai.at

## ABSTRACT

We explore a novel way of conceptualising the task of polyphonic music transcription, using so-called invertible neural networks. Invertible models unify both discriminative and generative aspects in one function, sharing one set of parameters. Introducing invertibility enables the practitioner to directly inspect what the discriminative model has learned, and exactly determine which inputs lead to which outputs. For the task of transcribing polyphonic audio into symbolic form, these models may be especially useful as they allow us to observe, for instance, to what extent the concept of single notes could be learned from a corpus of polyphonic music alone (which has been identified as a serious problem in recent research). This is an entirely new approach to audio transcription, which first of all necessitates some groundwork. In this paper, we begin by looking at the simplest possible invertible transcription model, and then thoroughly investigate its properties. Finally, we will take first steps towards a more sophisticated and capable version. We use the task of piano transcription, and specifically the MAPS dataset, as a basis for these investigations.

## 1. INTRODUCTION

For practitioners who apply deep neural network models to music information retrieval tasks, interpretability of predictions is of great interest. Knowing what the model was able to learn from the data, and examining the underlying causes for a prediction increases trust in the model. Being aware of the reasons for a classification result allows us to discover whether the model has learned rules that would pass a basic sanity check with a domain expert, or if it has picked up on seemingly irrelevant factors present in the data, which made it possible for the network to solve the task in a different, unexpected, possibly unwanted way [31]. There are quite a few ways to obtain an explanation from a neural network. Several methods use the gradient of an output with respect to the input as a starting point, such as [28, 32]. There are also methods that aim to provide model agnostic explanations, for instance [24, 27], and a specialization of one of the aforementioned methods to MIR systems [23] in particular.

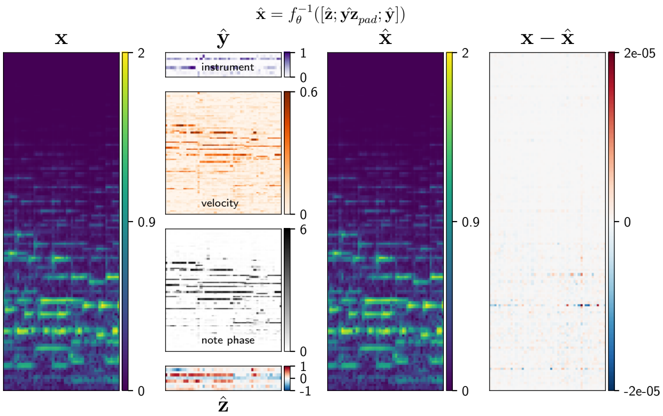

**Figure 1**: Computing the framewise transcription $\hat{\mathbf{y}}$ and nuisance variables $\hat{\mathbf{z}}$ from spectrogram input $\mathbf{x}$. The predictions $[\hat{\mathbf{y}}; \widehat{\mathbf{y}\mathbf{z}}_{pad}; \hat{\mathbf{z}}]$ are then used to exactly reproduce $\hat{\mathbf{x}}$. The elementwise difference $\mathbf{x} - \hat{\mathbf{x}}$ is negligible. An in-depth discussion of this figure is deferred until Section 4.2.

Beyond providing explanations for predictions, a model should ideally be able to provide an answer to the question “What do you consider representative examples for a concept of interest?”. Taking first steps towards producing models that are able to derive semantic information from the input, *and* are able to answer this question, we explore invertible neural networks (INNs) with respect to interpretability of predictions, their potential to identify biases and confounding factors inherent in the training dataset, and ability to generate samples for a semantic concept of interest.

Additionally, we consider ways in which these models could enable us to locate ambiguous or uncertain predictions on unlabeled data, to provide eventual users of the MIR model with hints on where manual postprocessing of the predictions might be advisable. We choose to conduct our investigation in the context of polyphonic piano transcription and provide a first glimpse at the capabilities of INNs in Figure 1. The input $\mathbf{x}$ to the INN is a magnitude spectrogram of an excerpt from a polyphonic piano piece, the output is split into a semantic part $\mathbf{y}$ containing variables of interest, and a nuisance part $\mathbf{z}$, optimistically containing all other factors of variation that are irrelevant for the MIR task the model was trained for. From these two output vectors, a hypothetical, perfectly converged invertible model can faithfully reproduce the input, down to a negligible numerical difference.

Invertible neural networks are parametrized, nonlinear and bijective functions, trainable from matched pairs, similar to any other neural network in a supervised learning task. The architectures we consider here are all constructed in such a way that the inverse is available in closed form.

Networks designed in this fashion have a few desirable properties. They are both discriminative and generative models unified in one function, sharing one set of parameters. To put this into context, training a transcription system also yields a synthesizer, and vice versa training a synthesizer yields a transcription system. The term "synthesizer" is used rather loosely here, as the transcription system is trained with magnitude spectrograms.

This setup enables a direct interpretation of predictions by looking at what samples the model produces, conditioned on the predictions. This can potentially be extended until after eventual postprocessing steps, to see whether the generated samples are still close to the input in data space.

Furthermore, in order for a practitioner to understand whether the discriminatively trained network has learned to distinguish multiple concepts reasonably well, she can directly obtain samples from the model for each different concept. As an illustrative example, we choose the task of transcribing polyphonic audio into a symbolic format. This is a multi-label problem, assigning multiple note labels to each (quantized) point in time. Transcription systems based on neural networks are commonly learned from corpora containing large amounts of polyphonic music. Due to having the inverse available to us in closed form, we are able to sample all different single notes from the network to directly see whether the concept of single, isolated notes could be learned by training on our polyphonic corpus, or if multiple notes have been "smeared" together, and could not be disentangled from each other, or if the concept could not be learned at all. To the best of our knowledge, this is still an open problem that mostly affects polyphonic transcription systems based on neural networks, as discussed in [20].

## 2. RELATED WORK

Invertible neural networks were first introduced in [6] and rediscovered in [2]. They define a nonlinear, bijective mapping between inputs and outputs. They can be used to transform arbitrarily complex distributions into simple, factorized distributions. This concept became more widely known as normalizing flows, introduced in [11], generalized in [33], and has been used in [8] for density estimation, and improving variational inference in [26]. Various types of (more expressive) normalizing flows have been introduced in [9,34,35]. In [21] normalizing flows are employed as generative models for high resolution samples comparable to those produced by high resolution generative adversarial networks (GANs), e.g., [18].

With a greater focus on the invertibility aspect, [1] uses bijective architectures to approximate physical processes with a well defined forward model, in order to obtain the posterior distribution over inputs conditioned on desired outputs. We adopt parts of their terminology and training procedure. The differences will be discussed in more detail in Section 3. In [17] injective and bijective i-RevNets are introduced, architectures similar to ResNets [14], which are invertible up to the last layer. In [16], fully invertible RevNets in conjunction with a new objective function are used to train classifiers which are more robust against adversarial attacks. We borrow their term "nuisance" variables to describe what information is supposed to end up in the output vector $\mathbf{z}$.

Distribution matching in this work is done using the sliced Wasserstein distance. Introduced in [4, 25] as a distance measure for texture synthesis in a computer graphics setting, it has been used for encouraging the codes of autoencoders to follow a proposal distribution [22], and has also been directly applied to generative modeling of images, replacing the domain regressor in GANs [7].

Finally, we draw inspiration from [13] where a transcription-resynthesis system was introduced, consisting of three separately trained parts, a transcription system, a language model and a (neural) synthesizer.

## 3. METHOD

We adhere closely to the invertible neural network architectures described in [1], with a minor modification to the training procedure that will be outlined after the formal introduction of invertible neural networks. Our notation also loosely follows the one used in [1]. Given a data space $\mathcal{X}$, a label space $\mathcal{Y}$ and a nuisance space $\mathcal{Z}$, we consider a directly invertible neural network as a parametrized function $f_\theta : \mathcal{X} \to \mathcal{Y} \times \mathcal{Z}$ where we have access to its closed form inverse $f_\theta^{-1} : \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$.

The function $f_\theta$ maps the input into a label space that carries semantic information that we are interested in, and maps the rest of information into a nuisance space. Given both the semantic and nuisance information, we are able to obtain the input again via $f_\theta^{-1}$.

There are a few different ways such a function can be implemented in practice, and they all come with various different architectural constraints. We will first define a small invertible building block with relatively weak capabilities. These building blocks are then used to construct a more expressive function.

A necessary structural restriction to a bijective block is that the dimensionality of the input must match the dimensionality of the output. Another restriction concerns the inner workings of blocks, so their inverse is available in closed form. We adopt the affine coupling layer design in [1], which is a more expressive version of the one in [9]. Its internal structure can be seen in Figure 2. The layer takes as input a vector $\mathbf{u}$, whose dimensions are first shuffled with a fixed random permutation matrix and then split into two halves $\mathbf{u}_1, \mathbf{u}_2$. Dimension shuffling causes the splits to be different from layer to layer and facilitates interaction between components whose indices might be far apart in the input vector. The permutation matrix is inverted by simply transposing it.

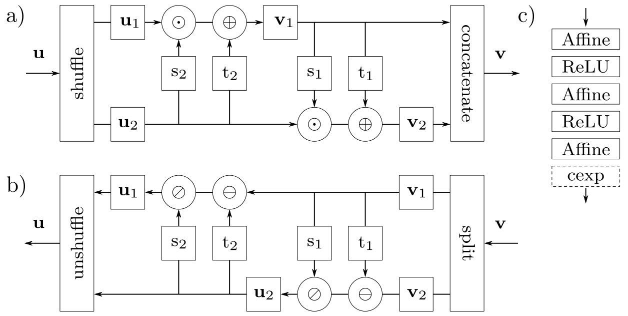

**Figure 2**: This sketch depicts the structure of the particular version of affine coupling layers we use. **a**) The operations as they are applied in the forward direction. **b**) The operations as they are applied in the backward direction. **c**) The parametrization of the $\mathrm{s}_{1|2}$ and $\mathrm{t}_{1|2}$ transforms. The cexp function is only applied after the $\mathrm{s}_{1|2}$ transforms.

Different operations are then applied to each half, after which the halves are concatenated again to yield the output $\mathbf{v}$. Equations (1) – (4) show the exact expressions used to compute results in both directions.

$$\mathbf{v}_1 = \mathrm{cexp}(\mathrm{s}_2(\mathbf{u}_2)) \odot \mathbf{u}_1 \oplus \mathrm{t}_2(\mathbf{u}_2) \tag{1}$$

$$\mathbf{v}_2 = \mathrm{cexp}(\mathrm{s}_1(\mathbf{v}_1)) \odot \mathbf{u}_2 \oplus \mathrm{t}_1(\mathbf{v}_1) \tag{2}$$

$$\mathbf{u}_2 = (\mathbf{v}_2 \ominus \mathrm{t}_1(\mathbf{v}_1)) \oslash \mathrm{cexp}(\mathrm{s}_1(\mathbf{v}_1)) \tag{3}$$

$$\mathbf{u}_1 = (\mathbf{v}_1 \ominus \mathrm{t}_2(\mathbf{u}_2)) \oslash \mathrm{cexp}(\mathrm{s}_2(\mathbf{u}_2)) \tag{4}$$

Operations $\oplus, \ominus, \odot, \oslash$ (addition, subtraction, multiplication, division) are applied elementwise. The function cexp is defined as $\mathrm{cexp}(x) = \exp(c \cdot \mathrm{atan}(x))$ with $c > 0$ being a hyperparameter. Its purpose is to constrain the output to a reasonable range, and to prevent runaway growth of activations. The transforms $\mathrm{s}_{1|2}$ and $\mathrm{t}_{1|2}$ are arbitrarily parametrizable functions, modeling input dependent scaling and translation respectively. All transforms are implemented as standard neural networks, and are not required to be invertible, because the transformed half of the output vector can be inverted using the untransformed half. The network structures we use are shown in Figure 2c.

If the dimensionalities of input and output vectors do not match, the vectors are padded with zeros during inference, or small scale Gaussian noise during training, to encourage the network to ignore the additional padding dimensions, as done in [1].

Each update of the model involves three passes, one forward pass, and two backward passes. Each pass has its own set of objective functions. The joint objective function to be minimized consists of a weighted sum of these terms. We specify the following notation: vectors are in boldface, writing vectors in square brackets separated by semicolons $[\mathbf{a}; \mathbf{b}]$ denotes concatenation. The vector $\mathbf{x}$ is the input to the model, $\mathbf{y}$ is the semantic part of the groundtruth, and $\mathbf{z}$ is a sample from a proposal distribution, which we choose to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The padding vectors used during training are denoted as $\mathbf{x}_{pad}$ and $\mathbf{yz}_{pad}$ respectively, and are drawn from $\mathcal{N}(\mathbf{0}, \varepsilon)$ for each update, with $\varepsilon > 0$ a hyperparameter. Symbols with a circumflex always refer to model outputs with a direct counterpart in the groundtruth. We denote a zero vector of a size appropriate in the context it appears in as $\mathbf{0}$. A sample from the model will be written as $\mathbf{x}_{sam}$. Equation (5) fully specifies all inputs and outputs for an invertible neural network used in the forward direction, equation (6) does the same in the backward direction, and (7) specifies how samples are drawn.

**Algorithm 1** Sliced Wasserstein Distance $d_{SWD}(\mathbf{A}, \mathbf{B})$

```
Let $S \leftarrow 0$ and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$ (two samples)
For $1 \mathinner{..} m$ do
    $\mathbf{p} \leftarrow \mathbf{p}' / \|\mathbf{p}'\|$ such that $\mathbf{p}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{p}' \in \mathbb{R}^{d \times 1}$
    $\mathbf{a} \leftarrow \mathrm{sort}[\mathbf{Ap}]$; $\mathbf{b} \leftarrow \mathrm{sort}[\mathbf{Bp}]$
    $S \leftarrow S + \|\mathbf{a} - \mathbf{b}\|_2^2 / n$
Return $S/m$
```

$$[\hat{\mathbf{z}}; \hat{\mathbf{yz}}_{pad}; \hat{\mathbf{y}}] = f([\mathbf{x}; \mathbf{x}_{pad}]) \tag{5}$$

$$[\hat{\mathbf{x}}; \hat{\mathbf{x}}_{pad}] = f^{-1}[\hat{\mathbf{z}}; \mathbf{yz}_{pad}; \hat{\mathbf{y}}] \tag{6}$$

$$[\mathbf{x}_{sam}; \hat{\mathbf{x}}_{pad}] = f^{-1}[\mathbf{z}; \mathbf{0}; \mathbf{y}] \tag{7}$$

Having defined these quantities, we can now proceed with defining the individual loss terms that will make up the joint objective function. Mean squared error (8) is used to fit the labels from the groundtruth, and the reconstruction of the input (9). We deviate from [1] and use the sliced Wasserstein distance ($d_{SWD}$) [25] instead of the maximum mean discrepancy ($d_{MMD}$), to measure the distance between distributions, as we found it to be better behaved for high dimensional data. The intuition behind $d_{SWD}$ is to decompose the high dimensional optimal transport problem into $m$ 1-dimensional ones, by randomly projecting samples $\mathbf{A}$ and $\mathbf{B}$ onto lines, allowing the resulting 1-dimensional problems to be solved by computing the distance between sorted components. In equation (10), $d_{SWD}$ is used to minimize the distance between samples from the joint distribution over the outputs $[\hat{\mathbf{y}}; \hat{\mathbf{z}}]$ and samples from the joint distribution over the labels and the proposal distribution $[\mathbf{y}; \mathbf{z}]$. Please note that following the advice laid out in [1], no gradient information from this objective is propagated back over $\hat{\mathbf{y}}$, to not unduly disturb the label fitting process. Informally stated, the purpose of including $\mathbf{y}$ and $\hat{\mathbf{y}}$ in the distribution matching process is to "group" samples together for which $\hat{\mathbf{z}}$ needs to follow a Gaussian distribution, resulting in the distributions $p(\hat{\mathbf{y}})$ and $p(\hat{\mathbf{z}})$ to gradually decouple, and become independent of each other, with the side effect of erasing label information from $\hat{\mathbf{z}}$. $d_{SWD}$ is also used in (11), to minimize the distance between the distribution of samples generated from the model and the groundtruth.

$$\mathcal{L}_{\mathbf{y}}(\mathbf{y}, \hat{\mathbf{y}}) = \mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}) \tag{8}$$

$$\mathcal{L}_{\hat{\mathbf{x}}}(\mathbf{x}, \hat{\mathbf{x}}) = \mathrm{MSE}(\mathbf{x}, \hat{\mathbf{x}}) \tag{9}$$

$$\mathcal{L}_{\mathbf{yz}}([\mathbf{y}; \mathbf{z}], [\hat{\mathbf{y}}; \hat{\mathbf{z}}]) = \mathrm{SWD}([\mathbf{y}; \mathbf{z}], [\hat{\mathbf{y}}; \hat{\mathbf{z}}]) \tag{10}$$

$$\mathcal{L}_{\mathbf{x}_{sam}}(\mathbf{x}, \mathbf{x}_{sam}) = \mathrm{SWD}(\mathbf{x}, \mathbf{x}_{sam}) \tag{11}$$

$$\mathcal{L}_{\mathbf{x}_{pad}}(\mathbf{x}_{pad}, \hat{\mathbf{x}}_{pad}) = \mathrm{MSE}(\mathbf{x}_{pad}, \hat{\mathbf{x}}_{pad}) \tag{12}$$

$$\mathcal{L}_{\mathbf{yz}_{pad}}(\mathbf{yz}_{pad}, \hat{\mathbf{yz}}_{pad}) = \mathrm{MSE}(\mathbf{yz}_{pad}, \hat{\mathbf{yz}}_{pad}) \tag{13}$$

Finally, the padding dimensions are taken care of with mean squared error terms (12) and (13), to encourage the network to disregard information in these dimensions. Following advice in [1], the individual loss terms that sum up to the joint objective are weighted such that their magnitudes are approximately equal to each other, by restarting the optimization process multiple times and adjusting the weights until this condition is met.

## 4. EXPERIMENTS

This section is split into multiple parts, starting out with a description of the data preparation procedure, followed by an empirical assessment of the usability of INNs for practitioners in subsection 4.1, a critical examination of the interpretability of a trained model in subsection 4.2, and finally an analysis of how well the concept of single notes could be learned from a polyphonic corpus in subsection 4.3.

All model training, testing and generative sampling experiments were carried out with the MUS subset of the MAPS corpus [10]. This subset contains 210 polyphonic piano pieces rendered with 7 sample based synthesizers, and 60 recordings of a YAMAHA Disklavier in two different recording conditions. After removing all synthesized pieces that also occur in the set of recordings, we are left with 139 pieces for training, and the 60 Disklavier recordings for testing, according to the procedure outlined in [12]. Evaluation measures are computed individually for each piece in the test set, and the mean over all pieces is reported. Groundtruth information is available as temporally aligned MIDI files. Sustain pedal control values are quantized, and the pedal considered fully engaged if its MIDI control value exceeds 64. All offsets of notes that are sounding while sustain is in effect are extended in time, until the pedal is released again.

The label information $\mathbf{y}$ that the model has to learn is derived from the MIDI groundtruth and consists of 3 parts: the note phase, its velocity and instrument information. For each piano key, the temporal evolution each note is modeled with an exponentially decaying curve, defined as $\mathrm{curve}(\tau) = 0.99^{\tau} \cdot 5$, with $0 \leq \tau < \mathrm{duration}$. It starts at the onset of a note, lasts for its duration, and drops off immediately after the offset. The velocity part is derived from the MIDI velocity value scaled into the interval $[0, 1]$. This procedure is also outlined in Figure 3, and repeated for each of the 88 piano keys. Finally, instruments are numbered from 0 to 8, corresponding to one of the 7 sample banks or alternatively one of the two microphone conditions for the Disklavier recordings, and are one-hot encoded. For each (quantized) point in time $t$ all three parts are concatenated into the vector $\mathbf{y}_t$, having $9 + 88 + 88 = 185$ components. This particular label vector derivation is chosen so that $\mathbf{y}_t$ contains all necessary information to generate spectrogram frames for different instruments and notes at the right volume and the right stage of a notes' temporal evolution without any additional context information from neighboring frames. The length of $\mathbf{z}_t$ was treated as a hyperparameter, and selected via cross validation on a small subset of the training set. Its length appears to have negligible influence given all the other settings, and was set to 9 for all models subsequently used. The corresponding data $\mathbf{x}_t$ are magnitude spectrograms processed by a semi logarithmic filterbank, and the resulting bins $\mathbf{b}_t$ are elementwise processed by the function $\log(1 + \mathbf{b}_t)$, approximating human loudness perception to finally yield a vector $\mathbf{x}_t$ of length 144. All spectral feature extraction and filtering is done with the `madmom` library [3]. The frame rate at which pairs $(\mathbf{x}_t, \mathbf{y}_t)$ are extracted from the audio and MIDI files is 25 frames per second. As all input is processed in a framewise fashion everywhere, we omit the subscript $t$, denoting time in frames, for all plots and most equations to not add additional clutter. To increase the capacity of the INN, we add zero padding vectors to both input ($\mathbf{x}_{pad}$) and output dimensions ($\mathbf{yz}_{pad}$), so the number of components in the padded vectors sum up to 256 in total. Training follows the procedure outlined in Section 3, and all code is released [1] to facilitate reproducability.

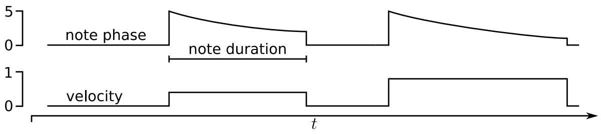

**Figure 3**: This illustration shows how the note phase and velocity part of the label information $\mathbf{y}$ is derived for multiple notes played by a single key.

### 4.1 Usability for MIR tasks

As a kind of quantitative viability test, the capability of INNs (in combination with simple temporal models) to produce predictions useful to MIR practitioners, is examined. We train small recurrent networks (RNNs) on the framewise predictions $\hat{\mathbf{y}}$ obtained from the INN, in the hope to obtain cleaner, denoised framewise predictions $\hat{\mathbf{y}}'$. The types of RNN cells we use are either LSTM [15] or GRU [5] cells. In a first attempt, RNNs with very limited capacity - 4 hidden units / cells for all keys - are employed. An input sequence to the RNN consists of the note phase and the velocity part of a *single* key over the whole length of the piece, leading to an input dimension of 2. The RNNs should output a smoothed, denoised version of the note phase and velocity sequence, and an additional framewise note activity indicator between 0 and 1, and thus all have 3 outputs. The binary piano roll used to compute framewise performance measures is obtained by thresholding the note activity indicator output of the RNNs at 0.5. Training proceeds one full sequence at a time, picked uniformly at random from all $139 \cdot 88$ single key sequences derived from all pieces in the training set. The models with highest $\mathrm{F}_1$-measure on a subset of the training set are then evaluated on the test set.

In order to evaluate framewise performance for a piece, we produce framewise transcriptions for all keys with the INN. Each key is then separately smoothed, denoised, and

[1] https://github.com/rainerkelz/ISMIR19

| Method | P | R | $F_1$ |
|---|---|---|---|
| INN + GRU (S) | 79.74 | 63.73 | 70.84 |
| INN + LSTM (S) | 80.12 | 63.91 | 71.10 |
| INN + biGRU (L) | 81.72 | 64.81 | 72.29 |
| CNN only [12, 19] | 81.18 | 65.07 | 71.60 |
| CNN + RNN-NADE [12, 29] | 71.99 | 73.32 | 72.22 |
| CNN + LSTM [12] | 88.53 | 70.89 | 78.30 |

**Table 1**: Framewise performance of different combinations of acoustic and temporal models on the testset.

its activity is inferred over the length of the piece. We report the results for the small models in Table 1, suffixed with “(S)”. We would like to note that there was next to no hyperparameter tuning done, aside from getting the learn rates for the two different RNN cell types approximately in the right regime. A slightly larger version uses 3 layers of bi-directional GRU cells with 8 hidden units, and dropout [30] with a probability of 0.5, applied to the output of each recurrent layer, before it is passed on to the next. Results in the table for this type of RNN are suffixed with “(L)”. The INN has 5 invertible layers, and 990.720 parameters in total. The parameter counts for the recurrent model variants “GRU (S)”, “LSTM (S)” and “biGRU (L)” are 111, 143 and 3123 respectively.

We can see that the combination framewise INN + biGRU performs on par with the CNN + RNN-NADE combination in terms of framewise performance, and slightly outperforms the standalone CNN. The last three rows in Table 1 are taken from [12], who re-implemented the approaches in [19] and [29], and ostensibly performed additional hyperparameter tuning to improve upon the original results. They also provide the current state of the art results for this train and test protocol in the last row, achieved by supplying an additional onset target to the network during training.

### 4.2 Interpretability of results

This section considers how the ability to modify the output of the model, and then using it in the backward direction, can assist the practitioner in determining the causes in the data that led to a particular prediction. We start with a thought experiment, and get closer to reality step by step. Let us assume the model works perfectly, and given an input $\mathbf{x}$, the model routes all semantic information (note phases, velocities, instrument) into the $\hat{\mathbf{y}}$ vector, all nuisance information (other acoustic variability, such as microphone characteristics, room reverberation or actual noise) ends up in $\hat{\mathbf{z}}$ which is distributed as $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and the padding vector $\hat{\mathbf{y}\mathbf{z}}$ is exactly zero. Sampling $\mathbf{z}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and using $f^{-1}([\hat{\mathbf{y}}, \mathbf{0}, \mathbf{z}_s])$ to obtain the corresponding input $\mathbf{x}_s$ will change *only nuisance characteristics* in the input. This would also mean that we have *full control* over the semantic content of the input. We could add or delete notes *in the input* simply by adding or zeroing them out in $\hat{\mathbf{y}}$, much like we can insert or delete a symbolic MIDI note, the implication being that every output has a *directly* interpretable correspondence in the input.

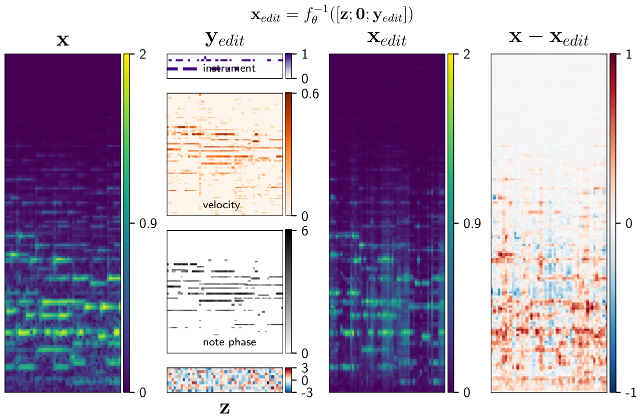

**Figure 4**: Gradually denoising the predictions with simple, ad-hoc rules, zero padding and sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In practice, this would be done iteratively, always keeping an eye on how the overall structure of the input is affected.

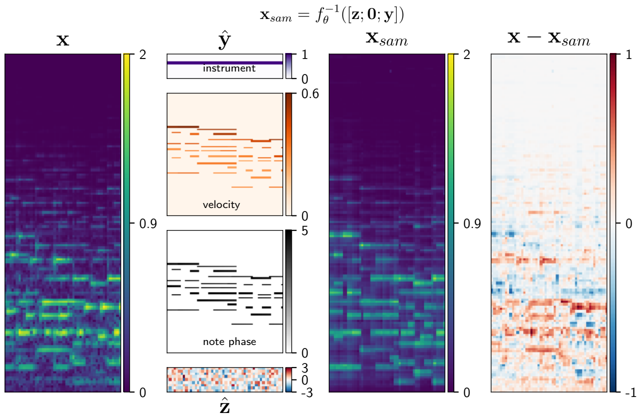

**Figure 5**: The hypothetical case where all predictions could be perfectly denoised. For purely demonstrational purposes, this was accomplished through consultation of an oracle, but one could imagine this to be achievable through interaction with the system. Although quite a bit of detail is missing, the majority of the structure in the original input is nonetheless recognizable in the generated sample.

A closer look at Figure 1 reveals what can realistically be achieved just by training a framewise INN with a rather limited amount of polyphonic data. It is immediately noticeable that $\hat{\mathbf{z}}$ does *not* appear to be normally distributed. There are still patterns discernible, making it apparent that it still contains semantic information. Similar patterns also exist in $\hat{\mathbf{y}\mathbf{z}}_{pad}$ (not shown).

It is also observable that the information that is routed into $\hat{\mathbf{y}}$ is somewhat noisy. We can now attempt to “separate the wheat from the chaff” by using the INN in the backward direction with cleaned up predictions. Figure 4 shows what happens when the predictions are partially denoised by setting all predictions below a certain threshold to zero, ignoring the padding vector by zeroing it out, and sampling $\mathbf{z}$ from a unit normal distribution. These simple, ad-hoc rules cannot get rid of all the noise and discontinuities in the predictions, but are useful to determine which outputs can be ignored by observing their (collective) im-

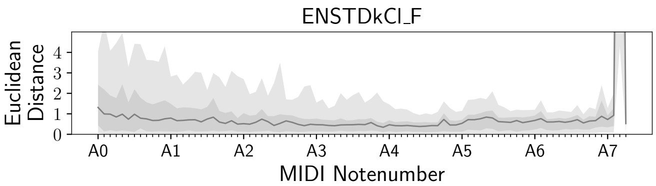

**Figure 6**: Interquantile ranges of the framewise Euclidean distances between isolated reference notes and samples from the model. The reference notes stem from an unseen instrument.

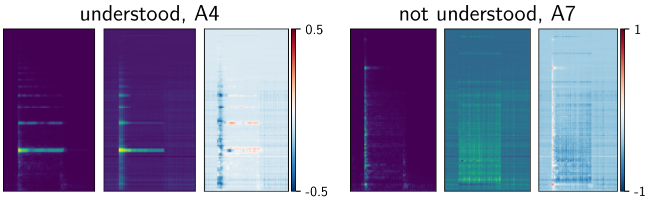

**Figure 7**: Samples from the model for single notes.

pact on the sample. Figure 5 depicts what can be generated by the model, assuming that the denoising process of the predictions were perfect, by consulting an oracle about the true contents of $\mathbf{y}$. In all figures discussed in this section, the same excerpt from the test set was used, meaning the model has never seen any of the examples during training.

### 4.3 Concept Understanding

Returning to a question raised in the introduction, in this section the model will be systematically queried about specific semantic concepts. Arguably, a polyphonic transcription system should be able to transcribe isolated notes. The MAPS dataset provides both renderings and recordings of isolated notes, which we utilize to formulate our queries. For each of the 88 keys, 30 samples are drawn from the model, using the groundtruth $\mathbf{y}$ paired with the reference recording for the key and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This ensures that each sample has the same length as the reference. For each frame at time $t$ in a sample, the Euclidean distance to the corresponding frame of the reference recording is measured, and $p$-quantiles are computed on the resulting lists of framewise distances, with $p \in \{0.05, 0.25, 0.5, 0.75, 0.95\}$. Figure 6 depicts the interquantile range $[0.05, 0.95]$ as light gray, the range $[0.25, 0.75]$ in a darker shade, and the median as a black line. It becomes immediately apparent that samples for rarely occuring (possibly omitted) notes, such as those in the lower and higher octaves, are highly dissimilar from the reference recordings, and indicate that these particular isolated notes could not be learned by the network (Figure 7). Admittedly, this question could have been answered for a regular feedforward network as well, but would have necessitated *more labeled* reference data of the *same* instrument. The ability to sample from the model allows us to sidestep the rather cumbersome way of aggregating prediction errors, as was necessary in [20], to arrive at a similar conclusion.

### 4.4 Improving Models with temporal context

The invertible models investigated so far all take single frames as input, without temporal context information. We trained fully invertible RevNets [16,17] on a variety of different context lengths, but were not yet able to observe either quantitative or qualitative improvements over the framewise models. RevNets tend to become rather large in terms of the number of parameters, input and output padding is not as straightforward as for framewise models, the input and output space dimensionality is much larger, making the sliced Wasserstein distance gradually less effective due to an increase in necessary computational resources, which in turn further slows down training. Finally, the amount of training data we use may simply be insufficient for higher capacity models. However, we believe that all these issues have appropriate remedies. An immediate next step would be to apply the same models to the much larger MAESTRO dataset [13]. We leave these steps for future work though.

## 5. CONCLUSION

The viability of invertible neural networks for a selected MIR task was shown quantitatively in terms of transcription performance and a brief numerical analysis of single concept understanding. A qualitative investigation of the direct interpretability of outputs back in input space was conducted. There is ample room for improvement, such as using an adversarial distance for distribution matching in both input and output space, or alternatively using the independent cross-entropy objective from [16] in latent space. The objective for the semantic part of the output space could be similarly augmented to encourage the disentanglement of (predictions for) individual notes. Another obvious improvement would be to skip the computation of filtered spectrograms altogether, and feed in waveforms to obtain models that can in turn generate waveforms we can directly listen to.

Beyond the interpretability aspect, we are confident that invertible neural networks will prove to be useful for other MIR tasks as well, such as musical content-aware style transfer (this is already doable with the models used in this work, by simply changing the instrument encoding when sampling, although changing from one piano to a different one is not as exciting as changing it into a trumpet). These models could also be adapted for (blind) source separation, to name only two examples.

## 6. ACKNOWLEDGEMENTS

We would like to express our deep gratitude to Lynton Ardizzone for pointing us in the right direction and Hamid Eghbal-zadeh for noting the pitch content-aware style transfer angle. This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 670035, project CON ESPRESSIONE). The Tesla K40 used for this research was donated by the NVIDIA Corporation.

## 7. REFERENCES

[1] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing Inverse Problems with Invertible Neural Networks. In *International Conference on Learning Representations*, 2019.

[2] L. Baird, D. Smalenberger, and S. Ingkiriwang. One-step neural network inversion with PDF learning and emulation. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 966–971 vol. 2, July 2005.

[3] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[4] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua 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 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014.

[6] Gustavo Deco and Wilfried Brauer. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995.

[7] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative Modeling Using the Sliced Wasserstein Distance. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3483–3491, 2018.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[10] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1643–1654, 2010.

[11] Esteban G. Tabak and Eric Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences - COMMUN MATH SCI*, 8, 03 2010.

[12] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and Frames: Dual-Objective Piano Transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27*, 2018.

[13] Curtis Hawthorne, Andrew Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. In *International Conference on Learning Representations*, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] Joern-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive Invariance Causes Adversarial Vulnerability. In *International Conference on Learning Representations*, 2019.

[17] Jörn-Henrik Jacobsen, Arnold W. M. Smeulders, and Edouard Oyallon. i-RevNet: Deep Invertible Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[19] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the Potential of Simple Framewise Approaches to Piano Transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 475–481, 2016.

[20] Rainer Kelz and Gerhard Widmer. An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. In *AES*

*International Conference Semantic Audio 2017, Erlangen, Germany, June 22-24, 2017*, 2017.

[21] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10236–10245, 2018.

[22] Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced Wasserstein Auto-Encoders. In *International Conference on Learning Representations*, 2019.

[23] Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Local Interpretable Model-Agnostic Explanations for Music Content Analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 537–543, 2017.

[24] Gregory Plumb, Denali Molitor, and Ameet S. Talwalkar. Model Agnostic Supervised Local Explanations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 2520–2529, 2018.

[25] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc" Bernot. Wasserstein Barycenter and Its Application to Texture Mixing. In Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, pages 435–446, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[26] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1530–1538, 2015.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[28] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 618–626, 2017.

[29] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An End-to-End Neural Network for Polyphonic Piano Music Transcription. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(5):927–939, 2016.

[30] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[31] B. L. Sturm. A Simple Method to Determine if a Music Information Retrieval System is a “Horse”. *IEEE Transactions on Multimedia*, 16(6):1636–1644, Oct 2014.

[32] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3319–3328, 2017.

[33] E. G. Tabak and Cristina V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[34] Jakub M. Tomczak and Max Welling. Improving Variational Auto-Encoders using Householder Flow. *CoRR*, abs/1611.09630, 2016.

[35] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 393–402, 2018.