An End-to-End Neural Network for Polyphonic Music Transcription

Sigtia, S; Benetos, E; Dixon, S

For additional information about this publication click this link.
http://qmro.qmul.ac.uk/xmlui/handle/123456789/9490

An End-to-End Neural Network for Polyphonic Music Transcription

Siddharth Sigtia, Emmanouil Benetos and Simon Dixon.

Abstract—**We present a neural network model for polyphonic music transcription. The architecture of the proposed model is analogous to speech recognition systems and comprises an *acoustic model* and a *music language model*. The acoustic model is a neural network used for estimating the probabilities of pitches in a frame of audio. The language model is a recurrent neural network that models the correlations between pitch combinations over time. The proposed model is general and can be used to transcribe polyphonic music without imposing any constraints on the polyphony or the number or type of instruments. The acoustic and language model predictions are combined using a probabilistic graphical model. Inference over the output variables is performed using the beam search algorithm. We investigate various neural network architectures for the acoustic models and compare their performance to two popular state-of-the-art acoustic models. We also present an efficient variant of beam search that improves performance and reduces run-times by an order of magnitude, making the model suitable for real-time applications. We evaluate the model's performance on the MAPS dataset and show that the proposed model outperforms state-of-the-art transcription systems.**

Index Terms—**Automatic Music Transcription, Deep Learning, Recurrent Neural Networks, Music Language Models.**

EDICS Category: AUD-MSP, AUD-MIR, MLR-DEEP

I. INTRODUCTION

AUTOMATIC Music Transcription (AMT) is a fundamental problem in Music Information Retrieval (MIR). AMT aims to generate a symbolic, score-like transcription, given a *polyphonic* acoustic signal. Music transcription is considered to be a difficult problem even by human experts and current music transcription systems fail to match human performance [1]. Polyphonic AMT is a difficult problem because concurrently sounding notes from one or more instruments cause a complex interaction and overlap of harmonics in the acoustic signal. Variability in the input signal also depends on the specific type of instrument being used. Additionally, AMT systems with unconstrained polyphony have a combinatorially very large output space, which further complicates the modeling problem. Typically, variability in the input signal is captured by models that aim to learn the timbral properties of the instrument being transcribed [2], [3], while the issues relating to a large output space are dealt with by constraining the models to have a maximum polyphony [4], [5].

The majority of current AMT systems are based on the principle of describing the input magnitude spectrogram as

The authors are with School of Electronics Engineering and Computer Science, Centre for Digital Music (C4DM) at Queen Mary University of London, E1 4NS, London, U.K.
Email: {s.s.sigtia,emmanouil.benetos,s.e.dixon}@qmul.ac.uk

a weighted combination of basis spectra corresponding to pitches. The basis spectra can be estimated by various techniques such as non-negative matrix factorisation (NMF) and sparse decomposition. Unsupervised NMF approaches [6], [7] aim to learn a dictionary of pitch spectra from the training examples. However purely unsupervised approaches can often lead to bases that do not correspond to musical pitches, therefore causing issues with interpreting the results at test time. These issues with unsupervised spectrogram factorisation methods are addressed by incorporating harmonic constraints in the training algorithm [8], [9]. Spectrogram factorisation based techniques were extended with the introduction of probabilistic latent component analysis (PLCA) [10]. PLCA aims to fit a latent variable probabilistic model to normalised spectrograms. PLCA based models are easy to train with the expectation-maximisation (EM) algorithm and have been extended and applied extensively to AMT problems [11], [3].

As an alternative to spectrogram factorisation techniques, there has been considerable interest in discriminative approaches to AMT. Discriminative approaches aim to directly classify features extracted from frames of audio to the output pitches. This approach has the advantage that instead of constructing instrument specific generative models, complex classifiers can be trained using large amounts of training data to capture the variability in the inputs. When using discriminative approaches, the performance of the classifiers is dependent on the features extracted from the signal. Recently, neural networks have been applied to raw data or low level representations to jointly learn the features and classifiers for a task [12]. Over the years there have been many experiments that evaluate discriminative approaches for AMT. Poliner and Ellis [13] use support vector machines (SVMs) to classify normalised magnitude spectra. Nam et. al. [14] superimpose an SVM on top of a deep belief network (DBN) in order to learn the features for an AMT task. Similarly, a bi-directional recurrent neural network (RNN) is applied to magnitude spectrograms for polyphonic transcription in [15].

Similarly to speech, musical sequences exhibit temporal structure. In addition to an accurate acoustic model, a model that captures the temporal structure of music or a music language model (MLM), can potentially help improve the performance of AMT systems. Unlike speech, language models are not common in most AMT models due to the challenging problem of modelling the combinatorially large output space of polyphonic music. Typically, the outputs of the acoustic models are processed by pitch specific, two-state hidden Markov models (HMMs) that enforce smoothing and duration constraints on the output pitches [3], [13]. However, extending

this to modelling the high-dimensional outputs of a polyphonic AMT system has proved to be challenging, although there are some studies that explore this idea. A dynamic Bayesian network is used in [16], to estimate prior probabilities of note combinations in an NMF based transcription framework. Similarly in [17], a recurrent neural network (RNN) based MLM is used to estimate prior probabilities of note sequences, alongside a PLCA acoustic model. A sequence transduction framework is proposed in [18], where the acoustic and language models are combined in a single RNN.

The ideas presented in this paper are extensions of the preliminary experiments in [19]. We propose an end-to-end architecture for jointly training both the acoustic and the language models for an AMT task. We train neural network acoustic models to identify the pitches in a frame of audio. The discriminative classifiers can be trained on complex mixtures of instrument sources, without having to account for each instrument separately. The neural network classifiers can be directly applied to the time-frequency representation, eliminating the need for a separate feature extraction stage. In addition to the deep feed-forward neural network (DNN) and RNN architectures in [19], we explore using convolutional neural nets (ConvNets) as acoustic models. ConvNets were initially proposed as classifiers for object recognition in computer vision, but have found increasing application in speech recognition [20], [21]. So far, ConvNets have remained unexplored for AMT. We also include comparisons with two state-of-the-art spectrogram factorisation based acoustic models [3], [8]. As mentioned before, the high dimensional outputs of the acoustic model pose a challenging problem for language modelling. We propose using RNNs as an alternative to state space models like factorial HMMs [22] and dynamic Bayesian networks [16], for modeling the temporal structure of notes in music. RNN based language models were first used alongside a PLCA acoustic model in [17]. However, in that setup, the acoustic and language models were trained by optimising different objectives, making the model theoretically unsatisfactory. Acoustic and language model training were combined under a single objective with the hybrid RNN framework in [19]. In the hybrid framework, *approximate* inference over the output variables is performed using beam search. However beam search can be computationally expensive when used to decode long temporal sequences. We apply the efficient hashed beam search algorithm proposed in [23] for inference. The new inference algorithm reduces decoding time by an order of magnitude and makes the proposed model suitable for real-time applications.

The rest of the paper is organised as follows: Section II describes the neural network models used in the experiment, Section III discusses the proposed model and the inference algorithm, Section IV details model evaluation and experimental results. Discussion, future work and conclusions are presented in Section V.

II. BACKGROUND

In this section we describe the neural network models used for the acoustic and language modelling. Although neural networks are an old concept, they have recently been applied to a wide range of machine learning problems with great success [12]. One of the primary reasons for their recent success has been the availability of large datasets and large-scale computing infrastructure [24], which makes it feasible to train networks with millions of parameters. The parameters of any neural network architecture are typically estimated with numerical optimisation techniques. Once a suitable cost function has been defined, the derivatives of the cost with respect to the model parameters are found using the backpropagation algorithm [25] and parameters are updated using stochastic gradient descent (SGD) [26]. SGD has the useful property that the model parameters are iteratively updated using small batches of data. This allows the training algorithm to scale to very large datasets. The layered, hierarchical structure of neural nets makes end-to-end training possible, which implies that the network can be trained to predict outputs from low-level inputs without extracting features. This is in contrast to many other machine learning models whose performance is dependent on the features extracted from the data. Their ability to jointly learn feature transformations and classifiers makes neural networks particularly well suited to problems in MIR [27].

A. Acoustic Models

1) Deep Neural Networks: DNNs are powerful machine learning models that can be used for classification and regression tasks. DNNs are characterised by having one or more layers of non-linear transformations. Formally, one layer of a DNN performs the following transformation:

$$h_{l+1} = f(W_l h_l + b_l). \tag{1}$$

In Equation 1, W_l, b_l are the weight matrix and bias for layer l, $0 \leq l \leq L$ and f is some non-linear function that is applied element-wise. For the first layer, $h_0 = x$, where x is the input. In all our experiments, we fix f to be the sigmoid function ($f(x) = \frac{1}{1+e^{-x}}$). The output of the final layer h_L is transformed according to the given problem to yield a posterior probability distribution over the output variables $P(y|x, \theta)$. The parameters $\theta = \{W_l, b_l\}_0^L$, are numerically estimated with the backpropagation algorithm and SGD. Figure 1a shows a graphical representation of the DNN architecture. For acoustic modelling, the input to the DNN is a frame of features (for example a magnitude spectrogram or the constant Q transform (CQT)) and the DNN is trained to predict the probability of pitches present in the frame $p(y_t|x_t)$ at some time t.

2) Recurrent Neural Networks: DNNs are good classifiers for stationary data, like images. However, they are not designed to account for sequential data. RNNs are natural extensions of DNNs, designed to handle sequential or temporal data. This makes them more suited for AMT tasks, since consecutive frames of audio exhibit both short-term and long-term temporal patterns. RNNs are characterised by recursive connections between the hidden layer activations at some time t and the hidden layer activations at $t-1$, as shown in Figure 1b. Formally, the hidden layer of an RNN at time t performs the following computation:
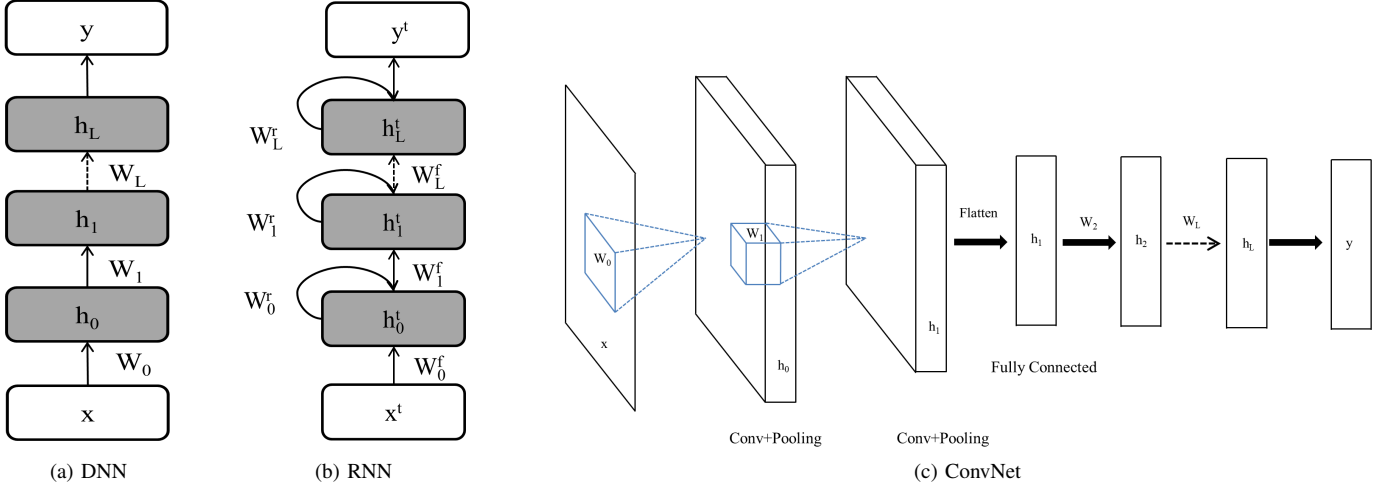
Fig. 1: Neural network architectures for acoustic modelling.

$$h_{l+1}^t = f(W_l^f h_l^t + W_{l+1}^r h_l^{t-1} + b_l). \qquad (2)$$

In Equation 2, W_l^f is the weight matrix from the input to the hidden units, W_l^r is the weight matrix for the recurrent connection and b_l are the biases for layer l. From Equation 2, we can see that the recursive update of the hidden state at time t, implies that h_t is implicitly a function of all the inputs till time t, x_0^t. Similar to DNNs, RNNs are made up of one or more layers of hidden units. The outputs of the final layer are transformed with a suitable function to yield the desired distribution over the ouputs. The RNN parameters $\theta = \left\{ W_l^f, W_l^r, b_l \right\}_0^L$ are calculated using the back propagation through time algorithm (BPTT) [28] and SGD. For acoustic modelling, the RNN acts on a sequence of input features to yield a probability distribution over the outputs $P(y_t | x_0^t)$.

3) Convolutional Networks: ConvNets are neural nets with a unique structure. Convolutional layers are specifically designed to preserve the spatial structure of the inputs. In a convolutional layer, a set of weights act on a local region of the input. These weights are then repeatedly applied to the entire input to produce a *feature map*. These features maps preserve the spatial information present in the data. Convolutional layers are characterised by the sharing of weights across the entire input. As shown in Figure 1c, ConvNets are comprised of alternating convolutional and pooling layers, followed by one or more fully connected layers (same as DNNs). Formally, the repeated application of the shared weights to the input signal constitutes a convolution operation:

$$h_{j,k} = f(\sum_r W_{r,j} x_{r+k-1} + b_j). \qquad (3)$$

The input x is a vector of inputs from different bands (for example, RGB bands for images), $x = \{x_i, \forall i\}$. Each input band x_i has an associated weight matrix. All the weights of a convolutional layer are collectively represented as a four dimensional tensor. Given an $m \times n$ region from a feature map

h, the max pooling function returns the maximum activation in the region. At any time t, the input to the ConvNet is a window of $2k + 1$ feature frames x_{t-k}^{t+k}. The outputs of the final layer yield the posterior distribution distribution $P(y_t | x_{t-k}^{t+k})$.

There are several motivations for using ConvNets for acoustic modelling. There are many experiments in MIR that suggest that rather than classifying a single frame of input, better prediction accuracies can be achieved by incorporating information over several frames of inputs [23], [29], [30]. Typically, this is achieved either by applying a context window around the input frame or by aggregating information over time by calculating statistical moments over a window of frames. Applying a context window around a frame of low level spectral features (like the short time fourier transform (STFT) or CQT) would lead to a very high dimensional input, which is impractical. Secondly, taking mean, standard deviation or other statistical moments makes very simplistic assumptions about the distribution of data over time in neighbouring frames. ConvNets, due to their architecture [12], can be directly applied to several frames of inputs to learn features along both, the time and the frequency axes. Additionally, the weight sharing and pooling architecture leads to a reduction in the number of ConvNet parameters, compared to a fully connected DNN. This is a useful property given that very large quantities of labelled data are difficult to obtain for most MIR problems, including AMT.

B. Music Language Models

Given a sequence $y = y_0^t$, we use the MLM to define a prior probability distribution $P(y)$. y_t is a high-dimensional binary vector that represents the notes being played at t (one time-step of a piano-roll representation). The high dimensional nature of the output space makes modelling y_t a challenging problem. Most post-processing algorithms make the simplifying assumption that all the pitches are independent and model their temporal evolution with independent models [13]. However, for polyphonic music, the pitches that are active concurrently are highly correlated (harmonies, chords). In this section, we

describe the RNN music language models first introduced in [31].

1) Generative RNN: The RNNs defined in the earlier sections were used to map a sequence of inputs x to a sequence of outputs y. At each time-step t, the RNN outputs the conditional distribution $P(y_t|x_0^t)$. However RNNs can be used to define a distribution over some sequence y by connecting the outputs of the RNN at $t-1$ to the inputs of the RNN at t, resulting in a distribution of the form:

$$P(y) = P(y_0) \prod_{t>0} P(y_t|y_0^{t-1}) \quad (4)$$

Although an RNN predicts y_t conditioned on the high dimensional inputs y_0^{t-1}, the individual pitch outputs $y_t(i)$ are independent, where i is the pitch index (Section IV-C). As mentioned earlier, this is not true for polyphonic music. Boulanger-Lewandowski et. al. [31] demonstrate that rather than predicting independent distributions, the parameters of a more complicated parametric output distribution can be conditioned on the RNN hidden state. In our experiments, we use the RNN to output the biases of a neural autoregressive distribution estimator (NADE) [31].

2) Neural Autogressive Distribution Estimator: The NADE is a distribution estimator for high dimensional binary data [32]. The NADE was initially proposed as a tractable alternative to the restricted Boltzmann machine (RBM). The NADE estimates the joint distribution over high dimensional binary variables as follows:

$$P(x) = \prod_i P(x_i|x_{<i})$$

where $x_{<i} = x_0^{i-1}$. The NADE is similar to a fully visible sigmoid belief network [33], since the conditional probability of x_i is a non-linear function of $x_{<i}$. The NADE computes the conditional distributions according to:

$$h_i = \sigma(W_{:,<i} x_{<i} + b_h) \quad (5)$$

$$P(x_i|x_{<i}) = \sigma(V_i h_i + b_v^i) \quad (6)$$

where W, V are weight matrices, $W_{:,<i}$ is a submatrix of W that denotes the first $i-1$ columns and b_h, b_v are the hidden and visible biases, respectively. The gradients of the likelihood function $P(x)$ with respect to the model parameters $\theta = \{W, V, b_h, b_v\}$ can be found exactly, which is not possible with RBMs [32]. This property allows the NADE to be readily combined with other models and the models can be jointly trained with gradient based optimisers.

3) RNN-NADE: In order to learn high dimensional, temporal distributions for the MLM, we combine the NADE and an RNN, as proposed in [31]. The resulting model yields a sequence of NADEs conditioned on an RNN, that describe a distribution over sequences of polyphonic music. The joint model is obtained by letting the parameters of the NADE at each time step, be a function of the RNN hidden state $\theta_{NADE}^t = f(h_t)$. h_t is the hidden state of final layer of the RNN (Equation 2) at time t. In order to limit the number of
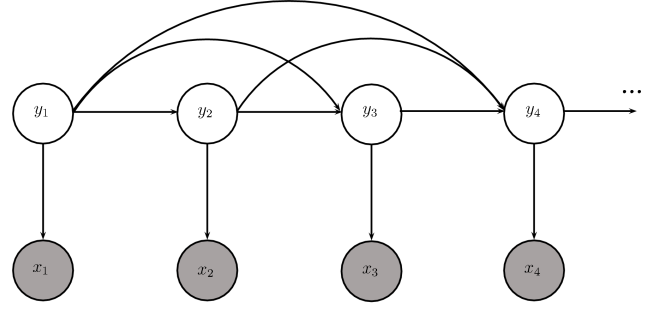

Fig. 2: Graphical Model of the Hybrid Architecture

free parameters in the model, we only allow the NADE biases to be functions of the RNN hidden state, while the remaining parameters (W, V) are held constant over time. We compute the NADE biases as a linear transformation of the RNN hidden state plus an added bias term [31]:

$$b_v^t = b_v + W_1 h_t \quad (7)$$

$$b_h^t = b_h + W_2 h_t \quad (8)$$

W_1 and W_2 are weight matrices from the RNN hidden state to the visible and hidden biases, respectively. The gradients with respect to all the model parameters can be easily computed using the chain rule and the joint model is trained using the BPTT algorithm [31].

III. PROPOSED MODEL

In this section we review the proposed neural network model for polyphonic AMT. As mentioned earlier, the model is comprised of an acoustic model and a music language model. In addition to the acoustic models in [19], we propose the use of ConvNets for identifying pitches present in the input audio signal and compare their performance to various other acoustic models (Section IV-F). The acoustic and language models are combined under a single training objective using a hybrid RNN architecture, yielding an end-to-end model for AMT with unconstrained polyphony. We first describe the hybrid RNN model, followed by a description of the proposed inference algorithm.

A. Hybrid RNN

The hybrid RNN is a graphical model that combines the predictions of any *arbitrary* frame level acoustic model, with an RNN-based language model. Let $x = x_0^T$ be a sequence of inputs and let $y = y_0^T$ be the corresponding transcriptions. The joint probability of y, x can be factorised as follows:

$$P(y, x) = P(y_0 \dots y_T, x_0 \dots x_T) \quad (9)$$
$$= P(y_0)P(x_0|y_0) \prod_{t=1}^{T} P(y_t|y_0^{t-1})P(x_t|y_t).$$

The factorisation in Equation 9 makes the following independence assumptions:

$$P(y_t|y_0^{t-1}, x_0^{t-1}) = P(y_t|y_0^{t-1}) \qquad (10)$$

$$P(x_t|y_0^t, x_0^{t-1}) = P(x_t|y_t) \qquad (11)$$

These independence assumptions are similar to the assumptions made in HMMs [34]. Figure 2 is a graphical representation of the hybrid model. In equation 9, $P(x_t|y_t)$ is the *emission* probability of an input, given output y_t. Using Bayes' rule, the joint distribution can be re-written as follows:

$$P(y, x) = P(y_0)\frac{P(y_0|x_0)}{P(y_0)}\prod_{t=1}^{T} P(y_t|y_0^{t-1})\frac{P(y_t|x_t)}{P(y_t)}\prod_{t=0}^{T} P(x_t) \qquad (12)$$

where $P(y_t|x_t)/P(y_t)$ are scaled likelihoods [13]. With this reformulation of the joint distribution, we observe that the prior distributions $P(y_t|y_0^{t-1})$ can be obtained from an RNN, while the posterior probabilities $P(y_t|x_t)$ can be obtained from any frame level classifier. The hybrid RNN graphical model is a generalisation of the HMM graph, where the state transition probabilities for the HMM $P(y_t|y_{t-1})$ have been generalised to include connections from all previous outputs, resulting in the $P(y_t|y_0^{t-1})$ terms in Equation 12.

For the problem of automatic music transcription, the input time-frequency representation forms the input sequence x, while the output piano-roll sequence y denotes the transcriptions. The priors $P(y_t|y_0^{t-1})$ are obtained from the RNN-NADE MLM, while the posterior distributions $P(y_t|x_t)$ are obtained from the acoustic models. The hybrid model is trained by maximising the joint probability of points x, y in the training set (Section IV-A). From Equation 12:

$$\log P(y, x) = \sum_{t=0}^{1} \log P(y_t|x_t) + \sum_{t=1}^{1} \log P(y_t|y_0^{t-1}) + const. \qquad (13)$$

The terms related to the acoustic and language models can be summed separately and the $\sum_t P(x_t)$ term can be treated as a constant since it does not depend on the model parameters. The derivatives with respect to the individual model parameters are easy to obtain:

$$\frac{\partial \log P(y, x)}{\partial \Theta_a} = \frac{\partial}{\partial \Theta_a}\sum_{t=0}^{T} \log P(y_t|x_t) \qquad (14)$$

$$\frac{\partial \log P(y, x)}{\partial \Theta_l} = \frac{\partial}{\partial \Theta_l}\sum_{t=1}^{T} \log P(y_t|y_0^{t-1}) \qquad (15)$$

where Θ_a, Θ_l are parameters of the acoustic and language models, respectively. The lack of cross terms allow the two models to be trained independently with gradient based optimisers.

B. Inference

At test time, we would like to find the mode of the conditional output distribution:

$$y^* = \underset{y}{\operatorname{argmax}} P(y|x)$$

The conditional probability can be written as:

$$P(y|x) = \frac{P(y, x)}{P(x)} \qquad (16)$$

$$= P(y_0)\frac{P(y_0|x_0)}{P(y_0)}\prod_{t=1}^{T} P(y_t|y_0^{t-1})\frac{P(y_t|x_t)}{P(y_t)}$$

assuming $P(x) = \prod_t P(x_t)$.

From Equations 12 and 16, we observe that the priors $P(y_t|y_0^{t-1})$, tie the predictions of the acoustic model $P(y_t|x_t)$ to all the predictions made till time t. This prior term encourages coherence between predictions over time and allows musicological structure learnt by the language models to influence successive predictions. However, this more general structure leads to a more complex inference (or decoding) procedure at test time. This is due to the fact that at time t, the history y_0^{t-1} has not been optimally determined. Therefore, the optimum choice of y_t depends on *all* the past model predictions. Proceeding greedily in a chronological manner (by selecting y_t that optimises $P(y_t|x_t)$) does not necessarily yield good solutions. We are interested in solutions that globally optimise $p(y|x)$. But exhaustively searching for the best sequence is intractable since the number of possible configurations of y_t is exponential in the number of output pitches (2^n for n pitches).

Beam search is a graph search algorithm that is commonly used to decode the conditional outputs of an RNN [35], [18], [23]. Beam search scales to arbitrarily long sequences and the computational cost versus accuracy trade-off can be controlled via the width of the beam. The inference algorithm is comprised of the following steps: at any time t, the algorithm maintains at most w partial solutions, where w is the beam width or the beam capacity. The solutions in the beam at t correspond to sub-sequences of length t. Next, all possible descendants of the w partial solutions in the beam are enumerated and then sorted in decreasing order of log-likelihood. From these candidate solutions, the top w solutions are retained as beam entries for further search. Beam search can be readily applied to problems where the number of candidate solutions at each step is limited, like speech recognition [36] and audio chord estimation [23]. However, using beam search for decoding sequences with a large output space is prohibitively inefficient.

When the space of candidate solutions is large, the algorithm can be constrained to consider only K new candidates for each partial solution in the beam, where K is known as the *branching factor*. The procedure for selecting the K candidates can be designed according to the given problem. For the hybrid architecture, from Equation 16 we note:

$$P(y_0^t|x_0^t) = P(y_0^{t-1}|x_0^{t-1})P(y_t|y_0^{t-1})P(y_t|x_t) \qquad (17)$$

At time t, the partial solutions in the beam correspond to configurations of y_0^{t-1}. Therefore given $P(y_0^{t-1}|x_0^{t-1})$, the K configurations that maximise $P(y_t|y_0^{t-1})P(y_t|x_t)$ would be a suitable choice of candidates for y_t. However for many families of distributions, it might not be possible to enumerate y_t in decreasing order of likelihood. In [18], the authors propose forming a pool of K candidates by drawing random samples from the conditional output distributions. However, random sampling can be inefficient and obtaining independent samples can be very expensive for many types of distributions. As an alternative, we propose to sample solutions from the posterior distribution of the acoustic model $P(y_t|x_t)$ [19]. There are 2 main motivations for doing this. Firstly, the outputs of the acoustic model are independent class probabilities. Therefore, it is easy to enumerate samples in decreasing order of log-likelihood [18]. Secondly, we avoid the accumulation of errors due to *teacher forcing* in the RNN language model during training [37]. The RNN models are trained to predict y_t, given the *true* outputs y_0^{t-1}. However at test time, outputs sampled from the RNN are fed back as inputs at the next time step. This discrepancy between the training and test objectives can cause prediction errors to accumulate over time.

Although generating candidates from the acoustic model yields good results, it requires the use of large beam widths. This makes the inference procedure computationally slow and unsuitable for real-time applications [19]. In this study, we propose using the *hashed beam search* algorithm proposed in [23]. Beam search is fundamentally limited when decoding long temporal sequences. This is due to the fact that solutions that differ at only a few time-steps, can saturate the beam. This causes the algorithm to search a very limited space of possible solutions. This issue can be solved by efficient pruning. The hashed beam search algorithm improves efficiency by pruning solutions that are *similar* to solutions with a higher likelihood. The metric that determines the similarity of sequences can be chosen in a problem dependent manner and is encoded in the form of a locality sensitive hash function [23]. In Algorithm 1, we outline the beam search algorithm algorithm used for our experiments, while Algorithm 2 describes the hash table beam object. In Algorithms 1 and 2, s is a sequence y_0^t, l is log-likelihood of s, m_a, m_l are acoustic and language model objects and f_h is the hash function.

There are two key differences between Algorithm 1 and the algorithm in [19]. First, the priority queue that stores the beam is replaced by a hash table beam object (see Algorithm 2). Secondly, for each entry in the beam we evaluate K candidate solutions. This is in contrast to the algorithm in [19], where once the beam is full, only w candidate solutions are evaluated per iteration. It might appear that the hashed beam search algorithm might be more expensive, since it evaluates $w * K$ candidates instead of w candidates. However, by efficiently pruning similar solutions, the algorithm yields better results for much smaller values of w, resulting in a significant increase in efficiency (Section IV-F, Figure 3).

Algorithm 2 describes the hash table beam object. The hashed beam search algorithm offers several advantages compared to the standard beam search algorithm. The notion of similarity of solutions can be encoded in the form of hash

---

Algorithm 1 High Dimensional Beam Search

Find the most likely sequence y given x with a beam width w and branching factor K.
$beam \leftarrow$ new beam object
$beam$.insert$(0, \{\})$
for $t = 1$ to T **do**
    $new\_beam \leftarrow$ new beam object
    for l, s, m_a, m_l **in** $beam$ **do**
        for $k = 1$ to K **do**
            $y' = m_a$.next_most_probable()
            $l' = \log P_l(y'|s)P_a(y'|x_t) - \log P(y')$
            $m'_l \leftarrow m_l$ with $y_t := y'$
            $m'_a \leftarrow m_a$ with $x := x_{t+1}$
            new_beam.insert$(l + l', \{s, y'\}, m_a, m_l)$
    $beam \leftarrow new\_beam$
return $beam$.pop()

---

functions. For music transcription, we choose the similarity function to be the last n frames in a sequence s. $n = 1$ corresponds to a dynamic programming like decoding (similar to HMMs) where all sequences with the same final state y_t are considered to be equivalent, and the sequence with the highest log-likelihood is retained. $n = $ len(sequence) corresponds to regular beam search. Additionally, the hash beam search algorithm can maintain ≥ 1 solution per hash key through a process called chaining [38].

Algorithm 2 Description of beam objects given w, f_h, k

Initialise beam object
beam.hashQ = defaultdict of priority queues*
beam.queue = indexed priority queue of length w**
Insert l, s **into beam**
key= $f_h(s)$
queue = beam.queue
hashQ = beam.hashQ[key]
fits_in_queue = **not** queue.full() **or** $l \geq$ queue.min()
fits_in_hashQ = **not** hashQ.full() **or** $l \geq$ hashQ.min()
if fits_in_queue **and** fits_in_hashQ **then**
    hashQ.insert(l, s)
    if hashQ.overfull() **then**
        item = hashQ.del_min()
        queue.remove(item)
    queue.insert(l, s)
    if queue.overfull() **then**
        item = queue.del_min()
        beam.hashQ[f_h(item.s)].remove(item)

* A priority queue of length k maintains the top k entries at all times.
** An *indexed* priority queue allows efficient random access and deletion.

IV. EVALUATION

In this section we describe how the performance of the proposed model is evaluated for a polyphonic transcription task.

A. Dataset

We evaluate the proposed model on the MAPS dataset [39]. The dataset consists of audio and corresponding annotations for isolated sounds, chords and complete pieces of piano music. For our experiments, we use only the full musical pieces for training and testing. The dataset consists of 270 pieces of classical music and MIDI annotations. There are 9 categories of recordings corresponding to different piano types and recording conditions, with 30 recordings per category. 7 categories of audio are produced by software piano synthesisers, while 2 sets of recordings are obtained from a Yamaha Disklavier upright piano. Therefore the dataset consists of 210 synthesised recordings and 60 real recordings.

In preliminary experiments, we observed that the performance of the acoustic models deteriorated when tested on piano types that were unseen at training time. We found it necessary to include examples from all the piano types during training, for the acoustic models to generalise well on unseen test data. We addressed this issue by dividing the entire dataset into 4 different training/test splits. For each split, we randomly select 80% of the data for training (216 musical pieces) and the remaining for testing (54 pieces). From each training split, we hold out 26 tracks as a validation set for selecting the hyperparameters for the training algorithm (Section IV-D). All the reported results are mean values of the evaluation metrics over the 4 splits.

In existing AMT literature, the acoustic models are usually trained on examples from the synthesised pianos and tested on the Disklavier recordings [3], [2], [40]. An issue with dividing the dataset into different train/test splits is that it makes comparisons with other acoustic models difficult. In order to perform fair comparisons, we use two state-of-the-art acoustic models [3], [8], in addition to the neural network models (Section IV-E). These models are tested on the same data as the neural network models and therefore allow direct comparison of results. By combining these acoustic models with the neural network based MLMs in the hybrid architecture, we also investigate the ability of the hybrid architecture to generalise to different acoustic models.

B. Metrics

We use both frame and note based metrics to assess the performance of the proposed system [41]. Frame-based evaluations are made by comparing the transcribed binary output and the MIDI ground truth frame-by-frame. For note-based evaluation, the system returns a list of notes, along with the corresponding pitches, onset and offset time. We use the F-measure, precision, recall and accuracy for both frame and note based evaluation. Formally, the frame-based metrics are defined as:

$$\mathcal{P} = \sum_{t=1}^{T} \frac{TP[t]}{TP[t] + FP[t]}$$

$$\mathcal{R} = \sum_{t=1}^{T} \frac{TP[t]}{TP[t] + FN[t]}$$

$$\mathcal{A} = \sum_{t=1}^{T} \frac{TP[t]}{TP[t] + FP[t] + FN[t]}$$

$$\mathcal{F} = \frac{2 * \mathcal{P} * \mathcal{R}}{\mathcal{P} + \mathcal{R}}$$

where TP[t] is the number of true positives for the event at t, FP is the number of false positives and FN is the number of false negatives. The summation over T is carried out over the entire test data. Similarly, analogous note-based metrics can be defined [41]. A note event is assumed to be correct if its predicted pitch onset is within a $\pm 50ms$ range of the ground truth onset.

C. Preprocessing

We transform the input audio to a time-frequency representation which is then input to the acoustic models. In [19], we used the magnitude short-time Fourier transform (STFT) as input to the acoustic models. However, here we experiment with the constant Q transform (CQT) as the input representation. There are two motivations for this. Firstly, the CQT is fundamentally better suited as a time-frequency representation for music signals, since the frequency axis is linear in pitch [42]. Another advantage of using the CQT is that the resulting representation is much lower dimensional than the STFT. Having a lower dimensional representation is useful when using neural network acoustic models as it reduces the number of parameters in the model.

We downsample the audio to 16 kHz from 44.1 kHz. We then compute CQTs over 7 octaves with 36 bins per octave and a hop size of 512 samples, resulting in a 252 dimensional input vector of real values, with a frame rate of 31.25 frames per second. Additionally, we compute the mean and standard deviation of each dimension over the training set and transform the data by subtracting the mean and diving by the standard deviation. These pre-processed vectors are used as inputs to the acoustic model. For the language model training, we sample the MIDI ground truth transcriptions of the training data at the same rate as the audio (32 ms). We obtain sequences of 88 dimensional binary vectors for training the RNN-NADE language models. The 88 outputs correspond to notes A0-C8 on a piano.

The test audio is sampled at a frame rate of 100 Hz yielding $100 * 30 = 3000$ frames per test file. For 54 test files over 4 splits, we obtain a total of $648,000$ frames at test time[1].

D. Network Training

In this section we describe the details of the training procedure for the various acoustic model architectures and the RNN-NADE language model. All the acoustic models have 88 units in the output layer, corresponding to the 88 output pitches. The outputs of the final layer are transformed by a sigmoid function and yield independent pitch probabilities $P(y_t(i) = 1|x)$. All the models are trained by maximising the log-likelihood over all the examples in the training set.

---

[1]It should be noted that carrying out statistical significance tests on a track level is an over-simplification in the context of multi-pitch detection, as argued in [43].

1) DNN Acoustic Models: For DNN training, we constrain all the hidden layers of the model to have the same number of units to simplify searching for good model architectures. We perform a grid search over the following parameters: number of layers $L \in \{1, 2, 3, 4\}$, number of hidden units $H \in \{25, 50, 100, 150, 200, 250\}$, hidden unit activations $act \in \{ReLU, sigmoid\}$ where ReLU is the rectified linear unit activation function [44]. We found Dropout [45] to be essential for improving generalisation performance. A Dropout rate of 0.3 was used for the input layer and all the hidden layers of the network. Rather than using learning rate and momentum update schedules, we use ADADELTA [46] to adapt the learning over iterations. In addition to Dropout, we use early stopping to minimise overfitting. Training was stopped if the cost over the validation set did not decrease for 20 epochs. We used mini batches of size 100 for the SGD updates.

2) RNN Acoustic Models: For RNN training, we constrain all the hidden layers to have the same number of units. We perform a grid search over the following parameters: $L \in \{1, 2, 3\}$, $H \in \{25, 50, 100, 150, 200, 250\}$. We fix the hidden activations of the recurrent layers to be the hyperbolic tangent function. We found that ADADELTA was not particularly well suited for training RNNs. We use an initial learning rate of 0.001 and linearly decrease it to 0 over 1000 iterations. We use a constant momentum rate of 0.9. The training sequences are further divided into sub-sequences of length 100. The SGD updates are made one sub-sequence at a time, without any mini batching. Similar to the DNNs, we use early stopping and stop training if validation cost does not decrease after 20 iterations. In order to prevent gradient explosion in the early stages of training, we use gradient clipping [47]. We clipped the gradients, when the norm of the gradient was greater than 5.

3) ConvNet Acoustic Models: The input to the ConvNet is a context window of frames and the target is the central frame in the window [23]. The frames at the beginning and end of the audio are zero padded so that a context window can be applied to each frame. Although pooling can be performed along both axes, we only perform pooling over the frequency axis. We performed a grid search over the following parameters: window size $w_s \in \{3, 5, 7, 9\}$ number of convolutional layers $L_c \in \{1, 2, 3, 4\}$, number of filters per layer $n_l \in \{10, 25, 50, 75, 100\}$, number of fully connected layers $L_{fc} \in \{1, 2, 3\}$, number of hidden units in fully connected layers $H \in \{200, 500, 1000\}$. The convolution activation functions were fixed to be the hyperbolic tangent functions, while all the fully connected layer activations were set to the sigmoid function. We tried a large permutation of window shapes for the convolutional layer and the following subset of window shapes yielded good results: $w \in \{(3, 3), (3, 5), (5, 5), (3, 25), (5, 25), (3, 75), (5, 75)\}$. We observed that classification performance deteriorated sharply for longer filters along the frequency axis. 0.5 Dropout was applied to all the fully connected layers. The model parameters were trained with SGD and a batch size of 256. An initial learning rate of 0.01 was linearly decreased to 0 over 1000 iterations. A constant momentum rate 0.9 was used for all the

Model	Architecture
DNN	$L = 3, H = 125$
RNN	$L = 2, H = 200$
ConvNet	$w_s = 7, L_c = 2, L_{fc} = 2, w_1 = (5, 25)$
	$w_2 = (3, 5), n_1 = n_2 = 50, h_1 = 1000, h_2 = 200$
RNN-NADE	$H_{RNN} = 200, H_{NADE} = 150$

TABLE I: Model configurations for the best performing architectures.

updates. We stopped training if the validation error did not decrease after 20 iterations over the entire training set.

4) RNN-NADE Language Models: The RNN-NADE models were trained with SGD and with sequences of length 100. We performed a grid search over the following parameters: number of recurrent units $H_{RNN} \in \{50, 100, 150, 200, 250, 300\}$ and number of hidden units for the NADE $H_{NADE} \in \{50, 100, 150, 200, 250, 300\}$. The model was trained with an initial learning rate of 0.001 which was linearly reduced to 0 over 1000 iterations. A constant momentum rate of 0.9 was applied throughout training.

We selected the model architectures by performing a grid search over the configurations described earlier in the section. The various models were trained on one train/test split and the best performing architecture was then used for all 4 splits in our experiments.

E. Comparative Approaches

For comparative purposes, two state-of-the-art polyphonic music transcription methods were used for experiments [3], [8]. In both cases, the non-binary *pitch activation* output of the aforementioned methods was extracted, for performing an in-depth comparison by combining the acoustic outputs with the proposed MLMs.

The multi-pitch detection method of [8] is based on non-negative matrix factorization (NMF) and operates by decomposing an input time-frequency representation as a series of basis spectra (representing pitches) and component activations (indicating pitch activity across time). This method models each basis spectrum as a weighted sum of narrowband spectra representing a few adjacent harmonic partials, enforcing harmonicity and spectral smoothness. As input time-frequency representation, an Equivalent Rectangular Bandwidth (ERB) filterbank is used.

The multiple-instrument transcription method of [3] is based on shift-invariant PLCA (a convolutive and probabilistic counterpart of NMF). In this model, the input time-frequency representation is decomposed into a series of basis spectra per pitch and instrument source which are shifted across log-frequency, thus supporting tuning changes and frequency modulations. Outputs include the pitch activation distribution and the instrument source contribution per pitch. Contrary to the parametric model of [8], the basis spectra are pre-extracted from isolated musical instrument sounds. As in the proposed method, the input time-frequency representation of [3] is the CQT.

Post Processing	None		Thresholding		HMM		Hybrid Architecture	
Acoustic Model	Frame	Note	Frame	Note	Frame	Note	Frame	Note
Benetos [3]	-	-	63.65	63.56	63.12	64.27	64.82	63.90
Vincent [8]	-	-	62.18	**70.21**	61.95	69.23	63.17	**70.89**
DNN	65.56	56.55	68.55	59.58	68.75	62.50	69.07	63.05
RNN	66.42	62.54	68.73	64.87	67.15	62.87	69.15	65.87
ConvNet	73.18	64.21	**74.03**	65.49	72.04	68.12	**74.76**	70.12

TABLE II: F-measures for multiple pitch detection on the MAPS dataset.

	$\mathcal{P}$		$\mathcal{R}$		$\mathcal{A}$	
Acoustic Model	Frame	Note	Frame	Note	Frame	Note
Benetos [3]	59.46	71.29	71.23	57.89	47.55	46.94
Vincent [8]	57.14	**80.64**	70.61	64.7	42.86	**55.92**
DNN	65.85	66.49	72.61	59.94	52.74	46.08
RNN	67.09	68.06	71.33	63.81	52.84	49.11
ConvNet	**72.7**	74.41	**76.94**	**66.29**	**59.69**	53.93

TABLE III: Precision, Recall and Accuracy for multiple pitch detection on the MAPS dataset using the hybrid architecture
$(w = 10, K = 4, k = 2, f_h(y_0^t) = y_t)$
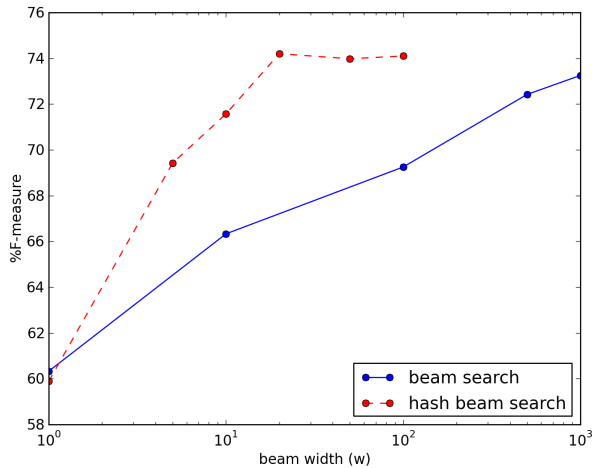
Fig. 3: Effect of beam width (w) on F-measure.
$k = 2, K = 4, f_h = y_t$

F. Results

In this section we present results from the experiments on the MAPS dataset. As mentioned before, all results are the mean values of various metrics computed over the 4 different train/test splits. The acoustic models yield a sequence of probabilities for the individual pitches being active (posteriograms). The post-processing methods are used to transform the posteriograms to a binary piano-roll representation. The various performance metrics (both frame and note based) are then computed by comparing the outputs of the systems to the ground truth.

We consider 4 kinds of post-processing methods. No post processing corresponds to setting the pitch activation to be one if the output probability is greater than 0.5 and 0 otherwise. Rather than setting an arbitrary threshold of 0.5 for each pitch, we also consider learning the thresholds for the individual

pitches. In this method, we select the threshold that maximises the F-measure over the entire training set and use this threshold for testing. We also use individual pitch HMMs for post-processing similar to [13]. The HMM parameters (transition probabilities, pitch marginals) are obtained by counting the frequency of each event over the MIDI ground truth data. The binary pitch outputs are obtained using Viterbi decoding [34], where the scaled likelihoods are used as emission probabilities. Finally, we combine the acoustic model predictions with the RNN-NADE MLMs and obtain binary transcriptions using beam search.

In Table II, we present F-scores (both frame and note based) for all combinations of acoustic models and post-processing methods. The third column of Table II shows results obtained by learning thresholds for post-processing. The performance of the acoustic models can be compared using these results. We note that all the neural network models outperform the PLCA and NMF models in terms of frame-based F-measure by $4\%-10\%$. The DNN and RNN acoustic model performance is comparable, while the ConvNet acoustic model clearly outperforms all the other models. The ConvNets yield an absolute improvement of $\sim 5\%$ over the other neural network models, while outperforming the spectrogram factorisation models by $\sim 10\%$ in frame-wise F-measure. For the note-based F-measure, the RNN and ConvNet models perform better than the DNN acoustic model. This indicates that these acoustic models are better at correctly predicting onsets. This is largely due to the fact that these models include context information in their inputs, which implicitly smooths the output predictions.

We compare the different post-processing methods by observing the rows of Table II. We note that the MLM leads to improved performance on both frame-based and note-based F-measure for all the acoustic models. The performance increase is larger on the note-based F-measure. The relative improvement in performance is maximum for the DNN acoustic model, compared to the RNN and the ConvNet. This
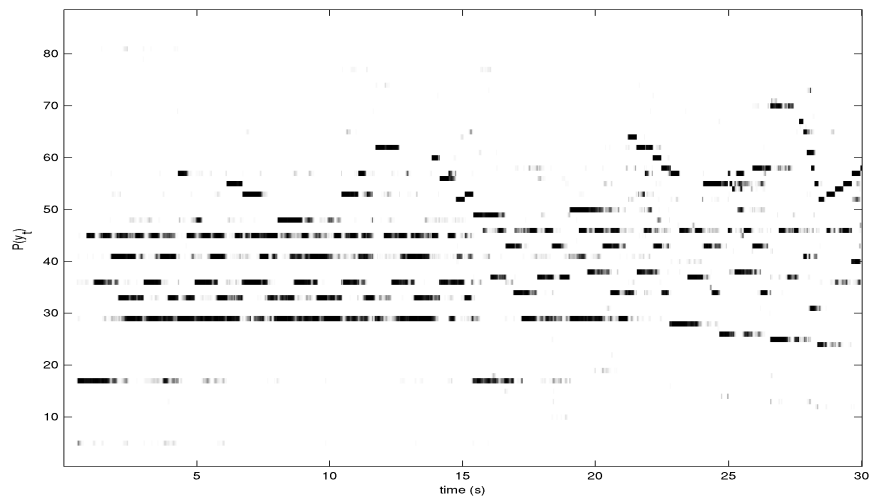
(a) ConvNet Posteriogram
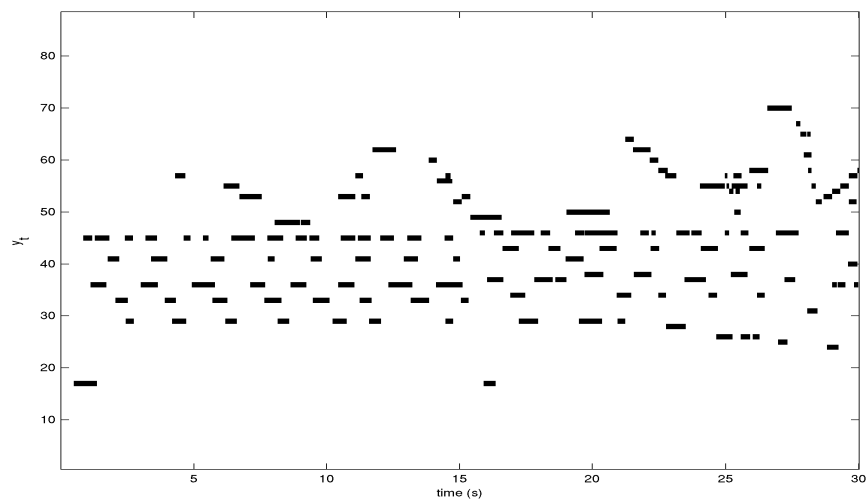
(b) ConvNet Transcription
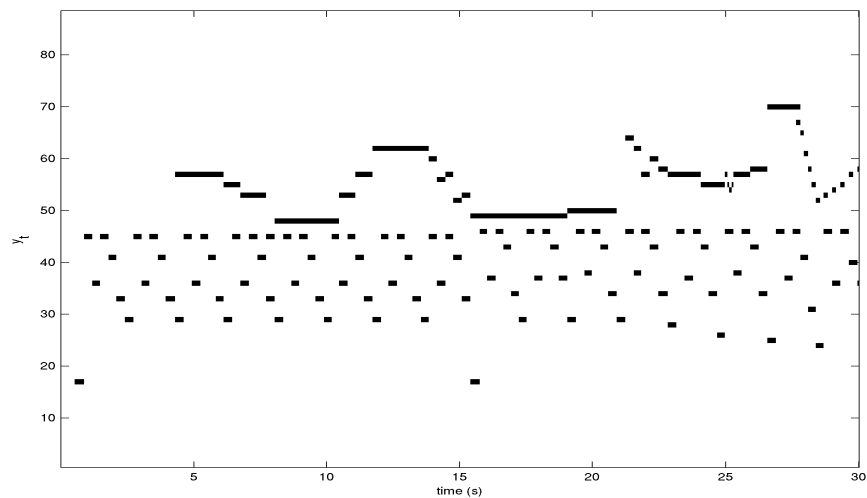
(c) Ground Truth

Fig. 4: a) Pitch-activation (posteriogram) matrix for the first 30 seconds of track MAPS_MUS-chpn_op27_2_AkPnStgb produced by a ConvNet acoustic model. b) Binary piano-roll transcription obtained from posteriogram in a) after post processing with RNN MLM and beam search. c) Corresponding ground truth piano roll representation.

could be due to the fact that the independence assumption in Equation 11 is violated by the RNN and ConvNet, which include context information while making predictions. This leads to some factors being counted twice and we observe a smaller performance improvement in this case. From Rows 1 and 2 of Table II we observe that the RNN-NADE MLM yields a performance increase for the PLCA and NMF acoustic models, though the relative improvement is much smaller as compared to the neural network acoustic models. This might reflect the fact that these models are not trained according to Equation 14, therefore violating the conditions which allow independent training of the acoustic and language models. Another contributing factor is the fact that the PLCA and NMF posteriograms represent the energy distribution over pitches rather than explicit pitch probabilities, which results in many activations being greater than 1. This discrepancy in the *scale* of the acoustic and language predictions leads to an unequal weighting of predictions when used in the hybrid RNN framework. In Table II we observe that the acoustic model in [8] outperforms all other acoustic models on the note-based F-measure, while the frame based F-measure is significantly lower. This can be attributed to the use of an ERB filterbank input representation, which offers improved temporal resolution over the CQT for lower frequencies.

In Table III, we present additional metrics (precision, recall and accuracy) for the all the acoustic models after decoding with an RNN-MLM. We observe that that the NMF and PLCA models have low frame-based precision and high recall and the converse for the note-based precision. For the neural network models, we observe smaller differences between the both frame-based and note-based precision and recall values. Amongst all the neural network models, we observe that the ConvNet outperforms all the other models on all the metrics.

We now discuss details of the inference algorithm. The high dimensional hashed beam search algorithm has the following parameters: the beam width w, the branching factor K, number of entries per hash table entry k and the similarity metric f_h (Algorithm 2). We observed that a value of $K \geq 4$ produced good results. Larger values of K do not yield a significant performance increase and result in much longer run times, therefore we set $K = 4$ for all experiments. We observed that small values of k (number of solutions per hash table entry), $1 \leq k \leq 4$ produced good results. Decoding accuracies deteriorate sharply for large values of k, as observed in [23]. Therefore, we set the number of entries per hash key $k = 2$ for all experiments. We let the similarity metric be the last n emitted symbols, $f_h(y_0^t) = y_{t-n}^t$. We experimented with varying the values of n and observed that we were able to achieve good performance for small n, $1 \leq n \leq 5$. We did not observe any performance improvement for large n, therefore for all experiments we fix $f_h(y_0^t) = y_t$. Figure 3 is a plot showing the effect of beam width w on transcription performance. The results are average values of decoding accuracies over 4 splits. We compare performance of the hashed beam search with the high dimensional beam search in [19]. From Figure 3 we observe that the hashed beam search algorithm is able to achieve performance improvement with significantly smaller beam-widths. For instance, the high

dimensional beam search algorithm takes 20 hours to decode the entire test set with $w = 100$, while the hashed beam search takes 22 minutes, with $w = 10$ and achieves better decoding accuracy.

Figure 4 is a graphical representation of the outputs of a ConvNet acoustic model. From Figure 4 a), b) we observe that the proposed model is able to correctly predict onsets, however it is not able to accurately model the note durations. We observe that some of the longer notes are fragmented and the offsets are estimated incorrectly. One reason for this is that the ground truth offsets don't necessarily correspond to the offset in the acoustic signal, implying noisy offsets in the ground truth. We also observe that the model does not make many harmonic errors in its predictions. Some of the semitone errors are possibly due to the dense polyphony of the test signal, which is a challenging modelling problem for both the acoustic and the language models.

V. CONCLUSIONS AND FUTURE WORK

We presented a hybrid RNN model for polyphonic AMT. The model comprises a neural network acoustic model and an RNN based music language model. We propose using a ConvNet for acoustic modelling, which to the best of the authors' knowledge, has not been attempted before. Our experiments on the MAPS dataset demonstrate that the neural network acoustic models, especially the ConvNet, outperform state-of-the-art acoustic models from the AMT literature. The RNN MLMs consistently improve performance on all evaluation metrics. The proposed inference algorithm with the hash beam search is able to yield good decoding accuracies with significantly shorter run times, making the model suitable for real-time applications.

In the proposed model, the MLM is responsible for modelling correlations between acoustic model predictions over time. Therefore, the MLM is responsible for smoothing the outputs, imposing musicological structure and modelling durations. Although the RNN MLMs help transcription performance, several open questions remain. The MLMs are trained on binary vectors sampled from the MIDI ground truth. Depending on the sampling rate, most note events are repeated many times in this representation. The MLMs are trained to predict the next frame of notes, given an input sequence of binary note combinations. In cases where the same notes are repeated many times, log-likelihood can be trivially maximised by repeating previous inputs. This causes the MLM to act as a smoothing function, rather than imposing any kind of musical structure on the outputs. A potential solution would be to perform beat-aligned language modelling for the training and the test data, rather than sampling the MIDI at some arbitrary sampling rate. Additionally, RNNs can be extended to include duration models for each of their pitch outputs, similar to second order HMMs. However, this is a challenging problem and currently remains unexplored. It would also be interesting to encourage RNNs to learn longer temporal note patterns by interfacing RNN *controllers* with external memory units [48].

The effect of tonality on the performance of the MLMs should be further investigated. The MIDI ground truth can

be easily transposed to any tonality. MLMs can be trained on inputs with transposed tonalities or individual MLMs for each key can be trained. Although the neural network acoustic models perform well, it would be interesting to study the effect of different kinds of input noise on their predictions. The performance of the acoustic models can be further improved by adding noise to training examples to encourage the classifiers to be invariant to commonly encountered input transformations. Additionally, the CQT input representation can be replaced by a representation with higher temporal resolution (like the ERB or a variable-Q transform), to improve performance on note based metrics.

REFERENCES

[1] A. Klapuri and M. Davy, *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[2] T. Berg-Kirkpatrick, J. Andreas, and D. Klein, "Unsupervised transcription of piano music," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 1538–1546.

[3] E. Benetos and S. Dixon, "A shift-invariant latent variable model for automatic music transcription," *Computer Music Journal*, vol. 36, no. 4, pp. 81–94, 2012.

[4] A. P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 804–816, 2003.

[5] V. Emiya, R. Badeau, and B. David, "Automatic transcription of piano music based on hmm tracking of jointly-estimated pitches," in *16th European on Signal Processing Conference*. IEEE, 2008, pp. 1–5.

[6] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2003, pp. 177–180.

[7] S. A. Abdallah and M. D. Plumbley, "Polyphonic music transcription by non-negative sparse coding of power spectra," in *5th International Society for Music Information Retrieval Conference (ISMIR)*, 2004, pp. 318–325.

[8] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Audio, Speech, and Language Processing.*, vol. 18, no. 3, pp. 528–537, 2010.

[9] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Transactions on Audio, Speech, and Language Processing.*, vol. 18, no. 3, pp. 538–549, 2010.

[10] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling." *Advances in models for acoustic processing, NIPS*, vol. 148, 2006.

[11] G. C. Grindlay and D. P. Ellis, "A probabilistic subspace model for multi-instrument polyphonic transcription," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval, 2010, pp. 21–26.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP Journal on Applied Signal Processing*, vol. 2007, no. 1, pp. 154–154, 2007.

[14] J. Nam, J. Ngiam, H. Lee, and M. Slaney, "A classification-based polyphonic piano transcription approach using learned feature representations." in *Proceedings of the 12th Internation Society for Music Information Retrieval Confernece (ISMIR)*, 2011, pp. 175–180.

[15] S. Böck and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 121–124.

[16] S. Raczynski, E. Vincent, S. Sagayama *et al.*, "Dynamic bayesian networks for symbolic polyphonic pitch modeling," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1830–1840, 2013.

[17] S. Sigtia, E. Benetos, S. Cherla, T. Weyde, A. S. d. Garcez, and S. Dixon, "An RNN-based music language model for improving automatic music transcription," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[18] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "High-dimensional sequence transduction," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 3178–3182.

[19] S. Sigtia, E. Benetos, N. Boulanger-Lewandowski, T. Weyde, A. S. d' Avila Garcez, and S. Dixon, "A Hybrid Recurrent Neural Network for Music Transcription," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 2061–2065.

[20] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4277–4280.

[21] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *INTERSPEECH*, 2013, pp. 3366–3370.

[22] E. Vincent and X. Rodet, "Music transcription with isa and hmm," in *Independent Component Analysis and Blind Signal Separation*. Springer, 2004, pp. 1197–1204.

[23] S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon, "Audio chord recognition with a hybrid neural network," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[24] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1223–1231.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, p. 3, 1988.

[26] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[27] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: new directions for music informatics," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 461–481, 2013.

[28] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[29] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks." in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 335–340.

[30] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and adaboost for music classification," *Machine learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[31] 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, pp. 1159–1166.

[32] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 29–37.

[33] R. M. Neal, "Connectionist learning of belief networks," *Artificial intelligence*, vol. 56, no. 1, pp. 71–113, 1992.

[34] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[35] A. Graves, "Sequence transduction with recurrent neural networks," in *Representation Learning Workshop, ICML*, 2012.

[36] N. Boulanger-Lewandowski, J. Droppo, M. Seltzer, and D. Yu, "Phone sequence modeling with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 5417–5421.

[37] B. Pearlmutter, "Dynamic recurrent neural networks," 1990.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms*. MIT press Cambridge, 2001, vol. 2.

[39] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[40] K. O'Hanlon and M. D. Plumbley, "Polyphonic piano transcription using non-negative matrix factorisation with group sparsity," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3112–3116.

[41] M. Bay, A. F. Ehmann, and J. S. Downie, "Evaluation of multiple-F0 estimation and tracking systems." in *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 315–320.

[42] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[43] E. Benetos, "Automatic transcription of polyphonic music exploiting temporal evolution," Ph.D. dissertation, Queen Mary University of London, Dec. 2012.

[44] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[47] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8624–8628.

[48] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom, "Learning to transduce with unbounded memory," *arXiv preprint arXiv:1506.02516*, 2015.