# Timbre Separation of Monophonic Mixes Using Convolutional Neural Networks

Student: Alexander Albarosa
Supervisor: Claudio Balocco
*Department of Engineering*
*University of Durham*

***Abstract*—This paper demonstrates the capability of convolutional neural networks (CNN) to simultaneously separate and identify pitches of instruments in a mix via timbral discrimination. The separation of an electric guitar and a trumpet was investigated due to their overlapping pitch range and distinct timbres. TensorFlow (TF) was used to construct and train CNN on Musical Instrument Digital Interface (MIDI) synthesised training datasets that covered all the possible monophonic pitch combinations of the two instruments. Two labelling approaches and a variety of CNN structures were tested, furthermore the effect of a novel approach to the data used for training to enhance timbre separation was analysed. The best network was shallow and achieved an averaged F-measure (F) score of 0.928 for the two instruments over three testing datasets, using the metrics defined in Section III-C, suggesting that CNN are capable of timbre separation with careful training. In addition a Jupyter notebook facilitating simple creation of labelled datasets from MIDI files has been made publicly available to aid future research[1].**

## I. Introduction

Timbre separation is one of the key problems that must be solved to enable Automatic Music Transcription (AMT) for audio sources containing multiple instruments. AMT can be defined as "the design of computational algorithms to convert acoustic music signals into some form of music notation" [1]. AMT is a complex problem and has been tackled since the late 1970s, starting with the work in [2]; despite this, current solutions using machine learning are still significantly below the level of a human expert [3]; one particular challenge is that the combinatorially large space of possible musical sounds and notation necessitates vast, labelled datasets, however such datasets are scarce and creating them is a laborious task. The complexity associated with full AMT has led to its reduction into a set of sub-problems which individual pieces of research focus on; in [1] these are recognised as *"(multi-)pitch estimation, onset and offset detection, instrument recognition, beat and rhythm tracking, interpretation of expressive timing and dynamics, and score typesetting"*. Furthermore, there are several key factors that compound the difficulty of AMT, namely the: loss of information due to simultaneous polyphonic sound events from different sources (instruments) [4]; ambiguity of musical notation; violation of statistical independence between source signals due to the metrical structure of music [1] and lack of annotated ground-truth transcriptions for supervised learning and testing (creating such annotations is very time consuming and requires high expertise [1]).

The most obvious use case for AMT is to significantly reduce the time taken for musicians to transcribe music (a frustration noted as early as 1743 by Jean Jacques Rousseau [5]); however, the International Society For Music Information Retrieval (ISMIR) research community has highlighted many other potential use cases of AMT such as: query-by-humming [6], automatic music tutoring [1], plagiarism detection [7] and recreation of improvised performances [7,8]. There is also a clear commercial opportunity for automatic music tutoring, *Yousician* [6] is a popular app that tailors the music tutoring process to an individual depending on their level. An appealing feature of *Yousician* is the ability to play, practice and receive feedback on songs featured in its library. AMT could significantly speed up the addition of songs into this library if it had timbral separation capabilities such that the pitches of a particular instrument, such as a lead guitar, could be automatically extracted from songs.

Recent systems for AMT can be broadly classified into two categories: new approaches using Neural Networks (NN) and traditional, unsupervised approaches not using NN. State-of-the-art results have been obtained with both traditional methods and NN [7] in recent history using the Music Information Retrieval Exchange (MIREX) [8] evaluation metrics, however in the last few years much research has concentrated around NN for their ability to learn high level features on their own [9], [10].

AMT systems typically fall into one of the following four levels [1]:

1) Frame-level: the system outputs the fundamental frequencies present for each time frame for all time frames of the input audio.
2) Note-level: the system outputs frame-level pitch estimates connected over time to produce a piano-roll representation. Each note estimated has a pitch, onset time and offset time.
3) Stream-level: the system outputs note-level transcriptions for each instrument in the mixture by analysing the timbres present in the audio.
4) Notation-level: the system output is a human-readable score, this is the final goal of AMT.

This work falls mostly in the frame-level, however it also incorporates aspects of both the stream and note-levels, a

[1]https://github.com/albarosaali/Timbre-Recognition-Using-CNN/

piano-roll output, see Fig 6, was produced for each separated instrument.

Recent frame-level approaches typically focus on polyphonic pitch estimation (a much harder task than monophonic pitch estimation [11]). Systems have used methods using: traditional signal processing [12], [13], spectral factorisation [14] (e.g. Non-Negative Matrix Factorisation (NMF) and NMF combined with Probabilistic Latent Component Analysis (PLCA) as seen in the work of [15]), and NN [16], [17]. According to [1], traditional signal processing methods are faster but less accurate; the work in [16] reports up to 9% improvement in pitch estimation accuracy using NN over NMF and PLCA methods.

Note-level approaches are currently the main focus of AMT [11] with NN achieving state-of-the-art results, especially when using CNN [3], [7], [18]. The work in [7] using CNN performs better than the best NMF approach in [19] by 1.5% and more traditional methods by approximately 30% in $F$-score (defined in Section. III-C) benchmarks, showing the great promise of NN (and CNN) with regards to note-level transcription.

Little research has been done on stream-level and notation-level AMT tasks. Stream-level AMT is closely related to Audio Source Separation (ASS) tasks [20], and generally relies on recognising timbral differences between instruments. ASS has been researched considerably, though knowledge of it has yet to be widely applied to AMT tasks. The work in [21] is one of the few that has attempted stream-level AMT, though their approach uses NMF and PLCA methods rather than NN. One approach that utilises CNN for a task approaching stream-level transcription is [9], here a CNN is used for frame-level instrument prediction; the task is considered as a multi-label classification problem. The input to the CNN consists of a CQT, a harmonic series feature (HSF) which represents the energy distribution of the harmonics of music notes (the authors believe that this is key in the perception of instrument timbre), and pitch information (they experiment with pitch information from ground-truth labels and from pitch estimations by the CNN in [7]). Another interesting feature is that they use 1D convolutions along time as opposed to 2D convolutions, which according to [22] may capture frequency and timbral information in each time-frame better.

The approach taken in this report specifically focuses on the ability of CNN to separate and identify the pitches of mixes containing a guitar and a trumpet via timbre. Two labelling approaches are tested and a novel approach to the data used for training, detailed in Section. III-A, is analysed to see if timbral information can be successfully identified and used to extract an accurate piano-roll representation of an input mix.

## II. Theory

Following the success of CNN implementations in computer vision applications, e.g. blurring faces in Google StreetView [43] and in radiology to class tumour types [44], the MIR community has attempted to leverage similar methods for AMT. In order to enable the use of CNN for AMT, audio is typically processed into spectrograms (a 2D frequency-time

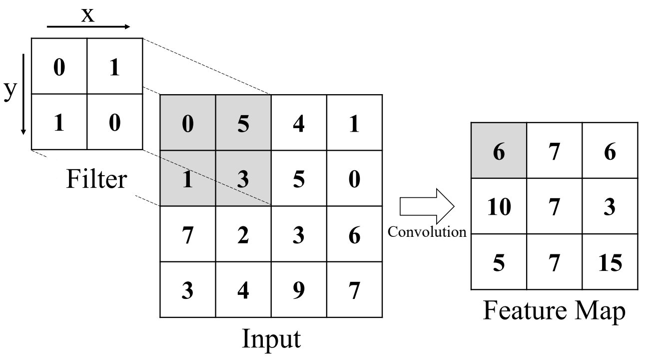



Fig. 1: Visual representation of the calculation of a pixel of a feature map from a simple 2D, size (2x2), filter and input using a stride of 1 in the x and y directions.

representation) and labelled. Where images datasets are readily available online to train computer vision networks, data for AMT applications, especially stream level, is scarce at best. The rest of this section covers the fundamental theory required to understand how CNN work and how they can be used for the separation of instruments by timbre.

### A. Convolutional Neural Networks

A CNN is a NN that contains convolutional layers which output feature maps, using learned weighted filters; each map represents a feature extracted at all locations in the input [23]. Convolutional layers are typically used in conjunction with pooling layers (to save computational resources) and this unit may be repeated many times before the outputs are fed into fully connected layers to enable classification. In supervised training (where the network is trained using labelled inputs): stochastic gradient descent (SGD) is used to minimise the difference between the desired and actual output of the network; the filters in each convolutional layer are updated simultaneously by the learning procedure and back-propagation is used to compute the gradients [23]. For a mathematically rigorous explanation regarding the extraction of feature maps from the input and the learning procedure of CNN the reader is directed to [23]–[25], although a brief explanation of the convolution and back-propagation is provided here. The operation of convolution between an input and a filter can be seen in Fig. 1; the highlighted number on the feature map is simply the sum of the element-wise multiplication of the filter with the highlighted section of the input, the window. Using a stride (the distance which the filter ' "hops" from one convolution to the next), $s = (1, 1)$ in the $x$ and $y$ direction respectively, the filter first moves horizontally across the input, hopping one element further along each convolution; once the filter reaches the end of the row it then, like a type-writer, returns to the first column displaced by one row in the $y$-direction and repeats the process. The result is a feature map of size $(3, 3)$. More generally, for an input of size $(i_x, i_y)$ and a filter of size $(k_x, k_y)$, using a stride $(s_x, s_y)$ the output is a feature map of size $((i_x - k_x)/s_x + 1, (i_y - k_y)/s_y + 1)$.

The second key idea behind CNN, and NN in general, is back-propagation. Back-propagation is the enabling idea

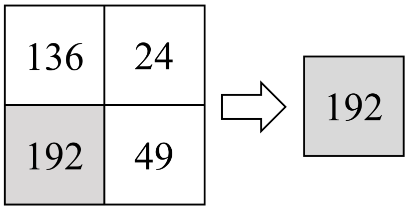



Fig. 2: The operation of a 2D max-pooling layer to reduce the size of the input. The maximum value in a 2x2 region is selected to constitute one pixel in the output.

behind SGD that allows networks to converge to local minima of the cost function, $C$, based on the negative gradient of $C$, or $-\nabla C$. Every time a new batch of outputs is produced by the network, the difference between the produced output and the desired output is calculated using $C$ (the particular $C$ used depends on the nature of the task). The goal of the network is then to minimise $C$ by adjusting the weights and biases which calculate the activation of each neuron corresponding to the outputs. The magnitude by which each weight and bias is changed is determined by $-\nabla C$; this quantity is calculated using back-propagation, which is achieved using a specific case of reverse automatic-differentiation to compute the partial derivatives required to calculate the sensitivity of the output neurons to any given weight or bias [24], [26]. These values allow the network to tune its weights and biases, in a controlled way, to minimise $C$ and thus produce the desired output.

There are certain different training methods in which back-propagation is used, a typical supervised learning approach may use a dense layer as the final layer to classify (or discriminate) the input sample then, using $C$, all the weights in the layers in the network are tuned accordingly, however other methods, such as adversarial, autoencoder and reinforcement learning can also be used. In adversarial learning two separate NN are trained, a generator and a discriminator; the discriminator learns to recognise the difference between the output from the generator and that of a real input that the generator is attempting to mimic, the task of the generator is to generate a "fake" output from input noise such that the discriminator is tricked into thinking that the "fake" input is in fact real at least 50% of the time [27]. Such networks have found use in music synthesis [28] and can be used to improve transcription networks [29].

One of the main advantages of using CNN is that they allow for progressively smaller features to be extracted as more convolutional layers are used, this increases the granularity of the information available for classification of the input. However the use of more layers requires increased computational resources; to offset this effect pooling layers are used to reduce the size of the feature maps between convolutional layers. The operation of a common pooling layer, *MaxPooling2D* [30], can be seen in Fig.2.

A few limitations of CNN are the requirement for large, accurately labelled datasets and the tendency to overfit the training data, though overfitting can be mitigated by the use of dropout layers which force the network to generalise better [17].

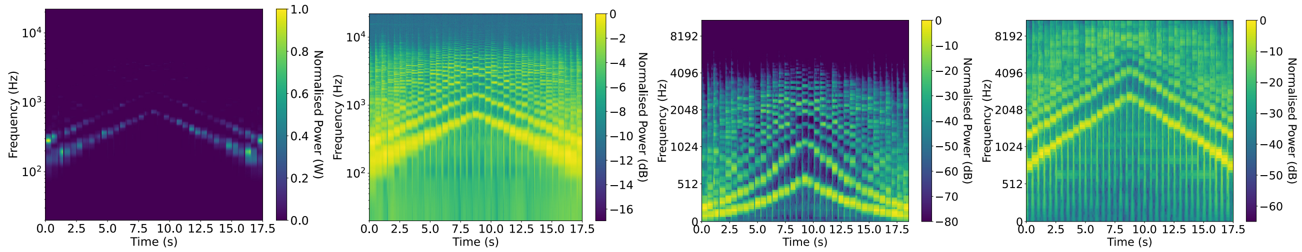
In the context of timbre separation and AMT, spectrograms (2D frequency-time representations of audio) have been used as input to CNN in numerous studies operating at the frame-level of AMT [7], [11], [17]. Spectrograms are in their nature inherently suited for input to CNN: music events are highly correlated over both time and frequency [1] and thus using spectrograms, CNN can learn high-level features and relationships temporally and in frequency [7]. Specifically regarding instrument separation by timbre; it is thought that CNN can be used to learn the unique energy distributions of partials for different instruments from spectrograms [9] . Unfortunately there have not been many studies into frame-level AMT and thus there is a palpable lack of data available for the task [31]. One way around this is to synthesise data using MIDI instruments. Though MIDI instruments can sound different to their real counterparts, they allow for simple generation of audio data through which easy labelling is facilitated by *Python* packages such as `pretty_midi` [32] which make parsing MIDI files into labels very easy. This ease of labelling is extremely important for AMT where making sure that labels are accurately aligned to spectrograms is a key, and time consuming issue to ensure efficient training of CNN [33]. Another advantage of synthesising and labelling one's own data for the task of timbre separation is the ability to accurately control the balance of data for different instruments in the training dataset.

It is important to remember that MIDI synthesised data can only go so far: there can be large differences in timbre between an instrument synthesised digitally using MIDI sound fonts and the same instrument recorded with a microphone in a studio. To enable NN to generalise well to real-world recordings, real instrument data should be used for training. If such data is unavailable then the network should be tested on real instrument recordings to gain valuable insight as to the applicability of the network in real instrument scenarios.

In this study it is proposed that in order to best train a neural network to be able to separate instruments by timbre and estimate their pitch, there are two key considerations to be made with regards to the training data. The data representing each instrument must include: the pitch in frames where it is playing; and a label symbolising a rest in all frames where it is not present, including data concerning all other instruments that could co-exist in a mix with the given instrument to be separated. The latter consideration is proposed as a way of clearly communicating to the NN that one instrument is not another instrument thus forcing the NN to recognise the timbral differences between them, leading to improved separation. With regards to the task of pitch estimation, the problem is considered as a multi-label classification problem with the loss function:

$$C_{cce} = -\sum_{c=1}^{n} b_{o,c} log(p_{o,c}) \tag{1}$$

where $c$ represents the class, $n$ is the number of possible classes, $b$ is a binary indicator as to whether the class classification $c$ is correct for output $o$ and $p$ is the predicted probability that $o$ belongs to $c$ [34]. Treating pitch estimation in this way works well for Western styles of music as each




(a) LS spectrogram (b) LM spectrogram (c) MEL spectrogram (d) CQT spectrogram

Fig. 3: The four commonly found spectrogram representations in AMT systems all displaying an ascending and descending G major scale played on an electric guitar.

class can be defined as corresponding to a specific MIDI pitch from 0 to 127 inclusive, representing the octave range C0 to G9.

### B. Spectrograms

Spectrograms are the key data enabling the use of CNN for AMT tasks, it is important to understand how they are constructed and good practices for their creation. As mentioned in the Section II-A spectrograms are 2-D representations of audio that can be labelled and used as input to a CNN for training. They are produced using the short-time Fourier transform (STFT) and there are four common ways to scale the result as a graph of frequency against time:

1) Logarithmically spaced frequency bins (LS).
2) Logarithmically spaced frequency bins and logarithmically scaled magnitude (LM).
3) Mel-scaled frequency bins (MEL).
4) Constant Q-Transform (CQT).

Fig. 3 shows these four types of spectrograms. Each one was computed at a sample rate of 44.1 kHz with an STFT window length of 1024 samples (giving a frequency bin resolution of 43 Hz) and hop-length of 512. Fig. 3a and Fig. 3b were manually implemented in python with help from the `pyFFTW` package [35], a wrapper around the popular C library `FFTW` that implements fast Fourier transform algorithms. Fig. 3c and Fig. 3d were computed with aid from the `librosa` package [36], a package specifically designed for music and audio analysis for music information retrieval.

Often research papers focus on one type of spectrogram, although some studies, such as [11], [17], show comparisons of results using the different spectrograms for training and testing. In practice, there is little difference between the spectrogram types, all apply some kind of logarithm scaling to the frequency axis, and there seems to be no standard or rule as to which representation to use, though [37] suggests that a CQT spectrogram is better suited for music tasks as the frequency axis is linear in pitch (this also allows for easier data augmentation with regards to pitch shifting samples). Where differences in research approaches can be found is in the window size chosen for the STFT: there is an inverse relationship between time and frequency resolution, i.e. shorter windows (increasing temporal resolution) lead to larger frequency bins (decreasing frequency resolution) and vice versa, thus there is a balance to be struck for AMT systems to be able to extract sufficient timing information for note values and sufficient frequency information for timbre separation and pitch detection. For AMT applications it is common to aim for a temporal resolution (hop-length) in the order of 10ms for a given sampling frequency [1], this is achieved by choosing a small hop-length between frames, however this can lead to large overlap which can cause ambiguity. To reduce computational overhead it is also common to downsample audio files [7], [14], [16], [38], [39], this does however come at the cost of reducing the highest frequency that can be recovered from the signal (due to Nyquist's sampling theorem).

## III. Method

In order to test the hypotheses made in Section. II-A, it was necessary to create tailored training and testing data. To enable easier alignment of labels and spectrogram frames, only 120BPM audio sources were used so that the duration of notes was always of the form $1/(2^n)$ (the duration of a crotchet would have $n = 1$). Furthermore, to simplify training and testing, only monophonic, two-instrument mixes were considered; the focus being a mix containing an electric guitar and a trumpet. This is a good combination to test the ability of a CNN to separate by timbre due to the overlapping pitch range of the instruments and their distinct timbres. The electric guitar features in all mixes as real guitar data could be recorded and tested using the *Native Instruments* Komplete Audio 1 USB Audio Interface. The NN were created and trained with the widely used `TensorFlow` (TF) [40] package.

### A. Dataset Creation and Data Preprocessing

The process of training dataset creation was broken down into five major steps:

1) Creation of a MIDI 'instrument ascension' (see text for definition).
2) Exporting the MIDI file as a `.wav` file.
3) Creation of spectrograms.
4) Spectrogram labelling.
5) Dataset creation.

The first step involved the use of `pretty_midi` to rapidly produce an 'instrument ascension' in the format of a `.mid` file. An instrument ascension was considered to be a sequence of crotchets starting from the lowest and ending at the highest note of an instrument's range. In the case of two instrument mixes, the instrument ascension consisted of every monophonic note combination between the two instruments. Secondly, the MIDI instrument ascension was opened in *Muscore 3*, a popular music-score editing software; soundfonts (a mixture of default and third party [41], [42]) were applied to the ascensions and the MIDI files were exported as `.wav` files (sampled at 44.1kHz). In the third step the `.wav` files were down-sampled to 32.768kHz (so that time covered by each spectrogram was always a multiple of note durations at 120BPM), converted into non-overlapping LM spectrograms covering frequencies from 20Hz to 16kHz and saved as `.png` images of size 128x512x3. This was achieved using *Python* and the packages: `librosa`, `pyfftw` and `matplotlib`. This relatively large image size was chosen so that as much spectrogram information as possible was available to the NN; furthermore, the sampling frequency was kept relatively high to retain the energy distribution of harmonics thought to be key to perceiving instrument timbre [9]. The fourth step involved labelling the pitch of the instrument. Two distinct approaches were taken: one used a "dual-label" containing two one-hot encoded, 129 bit arrays, corresponding to the two separate instruments, and the other simply used one, similarly encoded, 129 bit array to label the pitch of the desired instrument, referred to as "single-label"; Fig. 4 shows the different labelling approaches. Note that this is one bit longer than the number of possible MIDI pitches: this extra bit was used to signify the lack of presence of the instrument in the spectrogram. In the fifth and final step the spectrograms were assigned their labels and were stored in the format of a HDF5 dataset using the `hdf5` [43] package. The HDF5 format was chosen due to its superior performance when reading large amounts of images [44].

The data used for testing underwent the same process, except for the first step. The MIDI files used for the creation of the testing dataset are from a specific subset of the Lakh-MIDI dataset [45], [46] known as the "clean MIDI subset". This subset contains the music from songs of different genres in MIDI format. Eight of these MIDI files were adapted to include just two monophonic melodies, corresponding to the instruments that the network was trained to separate, and set to a tempo of 120BPM before being exported as `.wav` files; furthermore the audio of each of the instruments playing alone was exported in the same way to allow for more rigorous analysis of the performance of the networks. There was a conscious effort made to use varying soundfonts to synthesis the `.wav` files when compared to the training data. As a result there were three testing sets, each containing 35 minutes of music, see Table I, from which spectrograms were made for testing as described previously.

TABLE I: The three dataset types used for testing.

| Dataset | Instrument Mix | |
|---|---|---|
| | Guitar | Trumpet |
| Combined-Test | ✓ | ✓ |
| Guitar-Alone-Test | ✓ | |
| Trumpet-Alone-Test | | ✓ |

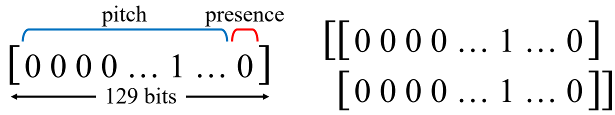



Fig. 4: The labelling methods used. The "single-label" approach can be seen on the left along with the bits allocated for the pitch and whether or not the instrument was present (0 means that the instrument is present). The "dual-label" approach can be seen on the right and simply contains two copies of the "single-label" format, each corresponding to one instrument in the mix.

### *B. Training Process*

The training process for every NN configuration consisted of a K-fold cross validation with K = 5 (implemented using the `scikit` [47] package), with each fold running through 20 epochs using a batch-size of 32 (where memory constraints allowed). K-fold cross validation refers to the process of first shuffling and then splitting the training dataset into K distinct folds each with a training:validation split of (1-1/K:1/K); when K = 5 this corresponds to an 80:20 split. The accuracy over the K-folds is averaged to get a more realistic value and for the purposes of this research, the fold with the best training accuracy was used to assess performance on the testing dataset. Unfortunately this process is time consuming due to the increased number of epochs required, thus to enable faster training, GPU acceleration in TF, with an NVIDIA GTX 1050 GPU with 4GB of video memory, was used.

### *C. Metrics*

The performance of each network was analysed via direct comparison of the piano-roll produced from the raw MIDI file versus the piano-roll extracted from the predictions of the NN (facilitated by the known duration of the spectrograms in time). A piano-roll is a pitch-time representation of audio data at a certain sampling frequency (this was chosen to be 100Hz, corresponding to a time resolution of 10ms, five times stricter than the 50ms used in the MIREX evaluation metrics [8]). Some examples of piano-rolls can be seen in Fig. 6.

The accuracy of predicted piano-rolls were measured by four activation metrics: "correct" (in time and pitch), "wrong" (correct in time but not in pitch), "false" (a note was found when there should have been no note found) and "missed" (there was no note found when one should have been found). From these metrics, the precision ($P$), recall ($R$) and F-measure ($F$) [48]:

$$P = \frac{correct}{correct + wrong + false} \tag{2}$$

$$R = \frac{correct}{correct + missed} \tag{3}$$

$$F = 2\frac{PR}{P + R} \tag{4}$$

were calculated to give a more holistic view of the performance of each network, considering that wrong, false and missed notes are highly undesirable in musical scenarios [1].

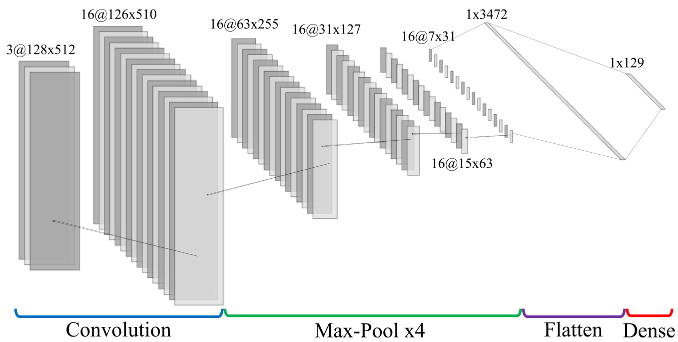



Fig. 5: Architecture of the shallow network used as a baseline for performance. The input is the 128x512x3 spectrogram; the convolutional layer produces 16 feature maps; the 2D max-pooling layers use 2x2 filters to reduce the size of the input to 7x31x16 before flattening; the dense layer is fully connected and has 129 output neurons.

TABLE II: Datasets used for training.

| Dataset | Frame Size | Number of Samples: Guitar | Trumpet | Combined | Total |
|---|---|---|---|---|---|
| Guitar Alone | 8192 | 742 | - | - | 742 |
| Trumpet Alone | 8192 | - | 225 | - | 225 |
| Both Alone | 8192 | 742 | 225 | - | 967 |
| Combined_1 | 8192 | 742 | 225 | 3243 | 4210 |
| Combined_2.0 | 8192 | 2968 | 3150 | 3243 | 9361 |
| Combined_2.1 | 4096 | 5936 | 6300 | 6486 | 18722 |

For every CNN, these metrics were calculated across the three testing datasets (as described in Section III-C) and the results were averaged to obtain the overall performance of any one CNN.

### D. Training and CNN Configurations

Initially, three CNN configurations were trained and tested on three different datasets each. The baseline neural network used with the "single-label" approach was considered to be a shallow network with the structure seen in Fig. 5. The output layer was a fully-connected dense layer, using a *softmax* activation with 129 outputs. The *softmax* activation ensures that the probabilities of all the output neurons sum to 1, this enabled easy extraction of the predicted MIDI pitch of the instrument using the `numpy` [49] package to find the index of the largest probability in the output pitch array. An analogous "dual-label" network was created to serve the same purpose as a baseline network. The two other networks had one and three additional convolutional layers before flattening. The convolution filter size used was 3x3 for all convolutional layers, more details about the structrure of the "dual-label" networks can be found in Table VI. The spectrograms in the datasets used at this stage had a frame-size of 8192 samples (corresponding to a duration of 0.25s) using a hop-length of 512 samples. The datasets used for training are summarised in Table II.

The best performing dataset (within the first four rows of Table II) was taken forward for both the label configurations; the dataset was then balanced resulting in an equal distribution of solo and combined instrument data, see row 5 of Table II. This dataset was then tested again on the three initial NN configurations and the best network and label configuration was noted.

Following this, the effect of using spectrograms with hop-lengths of 32 and 1024, and frame-sizes of 4096, 8192 (corresponding to a duration of 0.125s and 0.25s respectively), were tested on the best network and label configuration to determine whether the hop-length and STFT window used had an effect on the ability of the network to separate by timbre and distinguish pitch. This allowed for the determination of the best representation of data for further experiments with varying NN configurations.

Once the best data configuration was found, experiments concerning the architecture of the CNN, and filter size of the convolutional layers were also conducted. If a certain new configuration performed worse than any previous configuration it was discarded.

## IV. Results

The Dual-Label Shallow (DLS) network (the "dual-label" version of the network seen in Fig. 5) performed the best on average across all the testing datasets out of all the networks tested. Table III summarises the performance of the three best networks. It was found that the "Combined_2.1" dataset, using spectrograms with frame-size and hop-length of 4096 and 512 samples respectively, performed best, furthermore all the best networks utilised the "dual-label" configuration and were shallow using only one convolutional layer for each instrument. The trumpet was more easily separated and identified than the guitar across all three testing datasets, despite variations on NN structure and the filter size used in the convolutional layers. Additionally it was observed that no significant increase in accuracy was obtained by using the "dual-label" over the "single-label" labelling approach, other than reducing the amount of data pre and post-processing, though it should be noted that the latter approach has the potential to be used in a more versatile manner than the former. The best networks were also tested on a real composition (created using a real electric guitar and the Versilian Studios Chamber Orchestra 2 (VSCO2) synthetic trumpet [50]) recorded in *Ableton 10 Live Lite*, by the author, to challenge the networks with unfamiliar data. The results illustrate the problem with depending entirely on MIDI synthesised samples for training, though a solution to generate more realistic sounding data is proposed.

### A. Effect of Input and Training Data

Throughout the testing the significance of training the CNN on the right combination and format of input training data was apparent. The initial networks were tested on the datasets present in the first five rows of Table II, the results can be seen in Table IV. The DLS network performed best when trained on the balanced "Combined_2.0" dataset, however it should be noted that it suffered from reduced guitar precision compared to the "Both Alone" dataset, potentially meaning that the CNN has greater difficulty distinguishing whether the guitar is in fact alone or playing at the same time as the trumpet, than it does under the same conditions for a trumpet using this training

TABLE III: The best performing dataset-network combinations on average across the three testing datasets. All three networks use the "dual-label" approach and are shallow with only one convolutional layer producing 16 feature maps; the stride is (1,1) unless otherwise stated.

| | Size of Filters | | | Hop | Guitar | | Trumpet | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| Network | Guitar | Trumpet | Dataset | Length | P | R | P | R | F | Acc. |
| Dual-Label Shallow (DLS) | (3,3) | (3,3) | Combined_2.1 | 512 | 0.836 | 0.988 | **0.912** | 0.993 | **0.928** | **0.868** |
| Time-Rectangular Filter (T-RF) | (64,16) stride: (32,8) | (3,3) | Combined_2.1 | 512 | 0.819 | 0.990 | 0.903 | **0.996** | 0.921 | 0.856 |
| Frequency-Rectangular Filter (F-RF) | (3,128) | (3,3) | Combined_2.1 | 512 | **0.838** | **0.997** | 0.905 | 0.993 | 0.927 | 0.867 |

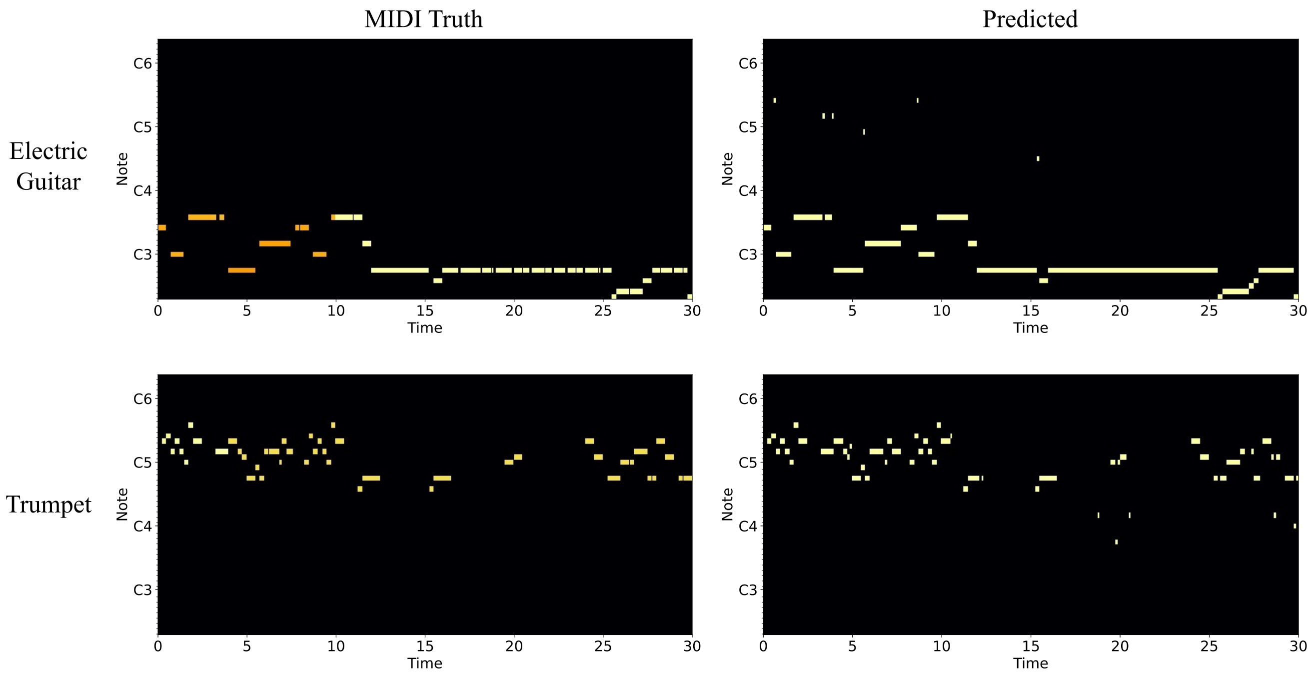



Fig. 6: Comparison between the piano rolls of the MIDI ground truth file (left) and the predicted values by the DLS network (right) on a 30 second excerpt of the song "Suffragette City" by David Bowie in the "Combined-Test" dataset. The spectrograms used had a frame-size and hop-length of 4096 and 512 samples respectively.

TABLE IV: Performance of the DLS network on the testing datasets for the different training dataset types. All datasets used spectrograms with a frame-size and hop-length of 4096 and 512 samples respectively.

| | Guitar | | Trumpet | | Average | |
|---|---|---|---|---|---|---|
| Dataset | P | R | P | R | F | Acc. |
| Guitar Alone | 0.529 | **1.000** | 0.333 | 0.333 | 0.509 | 0.431 |
| Trumpet Alone | 0.333 | 0.333 | 0.436 | **0.999** | 0.454 | 0.385 |
| Both Alone | **0.804** | 0.870 | 0.879 | 0.855 | 0.843 | 0.749 |
| Combined_1 | 0.703 | 0.990 | 0.875 | 0.998 | 0.877 | 0.786 |
| Combined_ 2.0 | 0.746 | 0.991 | **0.889** | 0.995 | **0.894** | **0.813** |

dataset. However the results here are promising as they suggest that the CNN can recognise the difference between the timbres of the guitar and the trumpet when trained in a supervised way. It should also be noted that the DLS network performed the best out of the the initial networks, out-performing those with more convolutional layers in the case of the "Combined" dataset type. When a "Both Alone" type dataset was used, deeper networks appeared to increase the average accuracy. In the case of the "Combined-Test" dataset, the difference in accuracy for the best network using the "Both Alone" type dataset lagged that of the same network using the "Combined" type by approximately 20%.

The DLS network was then tested with a variety of spectrogram parameters as described in Section III-D. It was found that spectrograms with a frame-size and hop-length of 4096 and 512 samples respectively performed best. Table V shows that the average accuracy was improved by approximately 5% by reducing the frame-size from 8192 to 4096 samples, with the gains of 10% seen in the "Combined-Test" and "Guitar-Alone-Test" datasets, however it is likely that the improvement in accuracy is as a result of the increased time resolution obtained by decreasing frame-size, as opposed to the network becoming better at distinguishing timbre. This conclusion is supported by the fact that the percentage of "false" activations remained largely unchanged, in fact the gains in accuracy came from the reduction in the percentage of "wrong" activations; though this illustrates the importance of high temporal resolution in AMT.

Unfortunately the number of "false" activations did not decrease when the hop-length was reduced to 32 samples; the increased overlap was hoped to increase the resolution of timbral information available in the spectrogram by showing a more accurate decay of the power distribution of the note harmonics in time. The failure to capitulate on this extra information may have been down to the use of a shallow

TABLE V: The effect of different frame-sizes and hop-lengths on the performance of the DLS network using the Combined_2 dataset type.

| Frame Size | Hop Length | Guitar P | Guitar R | Trumpet P | Trumpet R | Average F | Average Acc. |
|---|---|---|---|---|---|---|---|
| 8192 | 32 | 0.772 | 0.984 | 0.894 | 0.994 | 0.902 | 0.826 |
| 8192 | 512 | 0.746 | **0.991** | 0.889 | 0.995 | 0.894 | 0.813 |
| 8192 | 1024 | 0.754 | 0.984 | 0.889 | 0.994 | 0.895 | 0.816 |
| 4096 | 32 | 0.810 | 0.989 | 0.908 | 0.993 | 0.919 | 0.853 |
| 4096 | 512 | **0.836** | 0.988 | **0.912** | 0.993 | **0.928** | **0.868** |
| 4096 | 1024 | 0.836 | 0.988 | 0.898 | **0.997** | 0.925 | 0.862 |

network that was unable to extract meaningful relationships from the increased temporal resolution of the spectrograms.

### B. Effect of Network Configurations

The results of testing on the three initial networks (described in Section III-D) are summarised in Table VI and demonstrate that deeper CNN were inferior to shallow networks when trained on the "Combined" type dataset. The biggest performance drop for the deeper networks (DLC-2 and DLC-4) was in the "Trumpet-Alone-Test" dataset, where the DLS network out-performed them by approximately 10-15% in $F$ score averaged across both the instruments; the deeper networks struggled to infer that no guitar was present at all in these recordings, suggesting that the deeper CNN had difficulty learning the differences between the timbres of the guitar and trumpet. To counteract this, networks were created with convolutional layers with long, thin filters, along time or frequency, to better capture the timbral characteristics of the instruments; this choice was motivated by the 1-D convolutions used in [22] and the rectangular filters used in [51]. The T-RF and F-RF networks in Table III show that the $R$ of the guitar was improved using these shapes of filters; it should be noted though that when these two filters were combined in one NN (i.e. a time-filter followed by a frequency-filter) as suggested by [51] the NN were unable to obtain a training accuracy of more than 40% for either instrument. It's clear that for the separation and pitch estimation of two instruments with a "Combined" type dataset, simpler networks are better, this is supported by observations made in the work of [28] that came to a similar conclusion for transcription networks.

On the other hand, it should be noted that when training the three initial networks on the "Both Alone" type of dataset, the number of "correct" activations increased as the networks became deeper across all three testing datasets. If this style of dataset was used, the use of deeper, more complex networks could allow for the separation of instrument mixes without having to train the networks on all the possible note combinations (as was the approach in the "Combined" type dataset); further work here is needed to see if the use of a sufficiently complex CNN can increase the accuracy using a "Both Alone" type dataset to levels comparable to networks trained using the "Combined" type datasets in situations where the two instruments are playing simultaneously.

TABLE VI: The effect of network depth on the performance using the Combined_2.0 dataset using spectrograms with a 512 sample hop-length. The DLS network was used as a starting point from which more convolutional layers were added.

| Network | Convolution Layers (Left to Right) Feature Maps | Filter Sizes | Average F | Average Acc. |
|---|---|---|---|---|
| DLS | 16 | (3,3) | **0.877** | **0.786** |
| DLC-2 | 16, 32 | (3,3), (3,3) | 0.861 | 0.769 |
| DLC-4 | 16, 32, 64, 128 | (3,3), (3,3), (3,3), (3,3) | 0.862 | 0.771 |

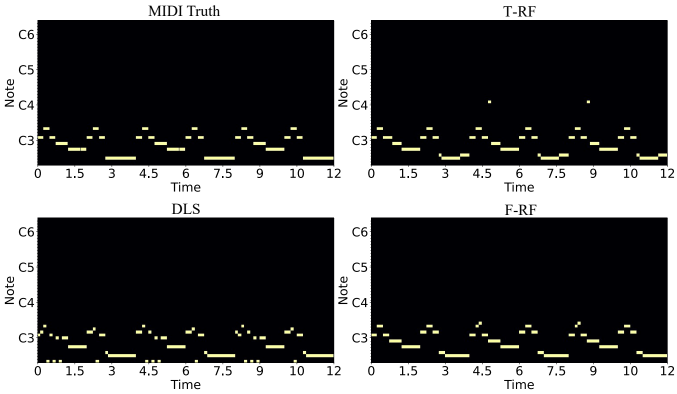



Fig. 7: The predicted piano-rolls for the guitar of the three networks seen in Table III in one of the author's own compositions containing both a real electric guitar and an unfamiliar synthesised trumpet.

### C. Effect of Labelling Approach

The two labelling approaches were found to differ minimally (<1%) in accuracy in all the initial tests. As a result, the "dual-label" approach was used thereafter as it allowed for joint prediction of the guitar and trumpet by one network, reducing the amount of processing required both for dataset creation and for piano-roll extraction. However the "single-label" approach is more versatile: a network could in theory be trained to pick one instrument out of a mix of instruments if a "Combined" type dataset were used; this could allow for a user to specifically select the instruments that they would like to be transcribed out of a mix of instruments.

### D. Testing on Real Data

To see how the networks in Table III performed on real data, a composition was made by the author using a real guitar as described in Section III and the VSCO2 MIDI trumpet that none of the networks had seen before. The guitar was recorded using the "Warm Tweed" amplifier preset in the *Guitar Rig 5* amplifier simulator. Figure 7 shows the predicted guitar of the three networks compared to the MIDI truth (none of the networks were able to distinguish the trumpet in this sample); the T-RF network performs the best out of the three achieving an accuracy of 82.3%, with the F-RF and DLS networks obtaining accuracies of 77.3% and 58.3% respectively; the reverse of the performance order seen in Table III. The much larger convolution filter used in the T-RF network seems to produce a more "stable" and accurate output with regards to pitch than the other networks. The performance of the F-RF network is relatively stable, though it consistently mistakes the C#3 for a D3, suggesting that the filter encompasses too many frequencies in the spectrogram. The DLS network performs by far the worst, the small filter size which worked well for the synthetic MIDI guitar seems too small to reliably capture held notes, often fluctuating within semitones of the actual pitch. There seems to be a balance to be struck with the filter size to recognise the timbre of the guitar in this particular case, when it is too small the network struggles to recognise held notes, and when it is too large, while the held notes are recognised, the pitch can be wrong. On the other hand, it is encouraging to see that the guitar was able to be distinguished from the mix of the two instruments, indicating that the networks had learnt the timbral features of the guitar.

To improve results networks should be trained using a combination of synthetic MIDI data and (elusive) real instrument data. A substitute for real guitar data could be found in passing the synthesised MIDI ascensions (using a clean guitar soundfont) through an amplifier simulator (such as *Guitar Rig 5*) in a digital audio workstation (DAW) in order to generate a new `.wav` file. In this way, a greater variety of guitar sounds could be obtained and easily labelled for use in training. This process is more difficult for the trumpet, though instrument plugins in DAWs, such as the VSCO2 trumpet could be used to a similar effect.

## V. Conclusion

The work in this report demonstrates the ability of CNN to separate and transcribe a two-instrument mix, containing an electric guitar and a trumpet, by recognising the timbre and pitch of each instrument. It has been found that a "Combined" type dataset, using LM spectrograms created from a `.wav` file downsampled to 32.768 kHz with a frame-size and hop-length of 4096 and 512 samples respectively, led to the best average performance across the testing datasets. Furthermore, it was found that simple, shallow networks achieved the best results with this dataset and that filter sizes can be tailored to emphasise the importance of timbral features of a particular instrument. The best average $F$ score, across the three testing datasets and the two instruments, was achieved by the DLS network with a score of 0.928 using the piano-roll metrics defined in Section III-C; it should be noted that this score takes into account both instrument separation as well as pitch recognition. As an aid to further research into AMT, a Jupyter Notebook, facilitating the simple creation of MIDI instrument ascensions and spectrograms for building labelled datasets, has been made publicly available (along with the rest of the code used in this project).

There are some limitations to the approach in this report: the relatively large frame-size of 4096 samples results in a relatively low time resolution of 0.125s (or a semi-quaver at 120BPM) and the labelling scheme is unable to tackle polyphonic sources or note-onsets and offsets. It is hoped that this approach can be combined with other areas of AMT research specifically focused on these tasks. Further work on this project could include: measuring the effect of reducing the frame-size below 4096 samples to see if a greater time resolution could be achieved without sacrificing separation and pitch detection accuracy; creating data using different MIDI instrument note velocities to improve timbre recognition (the timbre of instruments varies with the loudness of the note they produce); utilising the versatility of the "single-label" approach to separate particular instruments from mixes (such as only transcribing a trumpet in a jazz band); and the construction of testing data not constrained to 120BPM. Nonetheless, the work in this report demonstrates that joint timbre separation and pitch detection of an electric guitar and a trumpet is possible using CNN trained only on simple MIDI synthesised instrument ascensions and serves as an encouraging step towards stream-level AMT.

## References

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] M. Piszczalski and B. A. Galler, "Automatic music transcription," *Computer Music Journal*, vol. 1, no. 4, pp. 24–31, 1977.

[3] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "An End-to-end Framework for Audio-to-Score Music Transcription on Monophonic Excerpts," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, (Paris, France), pp. 34–41, ISMIR, Sept. 2018.

[4] M. Roman, A. Pertusa, and J. Calvo-Zaragoza, "A Holistic Approach to Polyphonic Music Transcription with Neural Networks," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, (Delft, The Netherlands), pp. 731–737, ISMIR, Nov. 2019.

[5] J. Rousseau, *Dissertation sur la musique moderne*. HACHETTE LIVRE, 1743.

[6] "Yousician." https://www.yousician.com. (accessed Apr. 19, 2021).

[7] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, 2018*, 2018.

[8] J. S. Downie, "Mirex home." https://www.music-ir.org/mirex/wiki/MIREX_HOME. (accessed Apr. 19, 2021).

[9] Y.-N. Hung and Y.-H. Yang, "Frame-level Instrument Recognition by Timbre and Pitch," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, (Paris, France), pp. 135–142, ISMIR, Sept. 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[11] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "Data representations for audio-to-score monophonic music transcription," *Expert Systems with Applications*, vol. 162, p. 113769, 2020.

[12] 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, 2009.

[13] L. Su and Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, pp. 1600–1612, 2015.

[14] F. Pedersoli, G. Tzanetakis, and K. M. Yi, "Improving music transcription by pre-stacking a u-net," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 506–510, 2020.

[15] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling," in *In Workshop on Advances in Models for Acoustic Processing at NIPS*, 2006.

[16] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[17] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. 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*, pp. 475–481, 2016.

[18] C. Thomé and S. Ahlbäck, "Polyphonic pitch detection with convolutional recurrent neural networks," in *18th International Society for Music Information Retrieval Conference, Suzhou, China*, 2017.

[19] L. Gao, L. Su, Y. Yang, and T. Lee, "Polyphonic piano note transcription with non-negative matrix factorization of differential spectrogram," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 291–295, 2017.

[20] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," *CoRR*, vol. abs/1910.12621, 2019.

[21] V. Arora and L. Behera, "Multiple f0 estimation and source clustering of polyphonic music audio using plca and hmrfs," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 2, pp. 278–287, 2015.

[22] S.-Y. Chou, J.-S. Jang, and Y.-H. Yang, "Learning to recognize transient sound events using attentional supervision," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3336–3342, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[23] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, 2010.

[24] A. G. Baydin, B. A. Pearlmutter, and A. A. Radul, "Automatic differentiation in machine learning: a survey," *CoRR*, vol. abs/1502.05767, 2015.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, p. 2278–2324, 1998.

[26] A. G. Baydin, B. A. Pearlmutter, and A. A. Radul, "Automatic differentiation in machine learning: a survey," *CoRR*, vol. abs/1502.05767, 2015.

[27] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[28] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," 2019.

[29] J. W. Kim and J. P. Bello, "Adversarial learning for improved onsets and frames music transcription," in *20th International Society for Music Information Retrieval Conference, Delft, The Netherlands, 2019*.

[30] Keras, "Keras documentation: Maxpooling2d layer." https://keras.io/api/layers/pooling_layers/max_pooling2d/. (accessed: Apr. 19, 2021.

[31] Y. Hung, Y. Chen, and Y. Yang, "Multitask learning for frame-level instrument recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 381–385, 2019.

[32] C. Raffel and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of midi data with pretty_midi," in *Proceedings of the 15th International Conference on Music Information Retrieval Late Breaking and Demo Papers, 2014*.

[33] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Invariances and data augmentation for supervised music transcription," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2241–2245, 2018.

[34] "Ml glossary documentation: Loss functions." https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. (accessed: Apr. 19, 2021.

[35] H. Gomersall, "pyfftw." https://github.com/pyFFTW/pyFFTW. (accessed: Apr. 19, 2021).

[36] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference* (Kathryn Huff and James Bergstra, eds.), pp. 18 – 24, 2015.

[37] 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.

[38] C. Chafe, D. A. Jaffe, K. Kashima, B. Mont-Reynaud, and J. O. Smith, "Techniques for note identification in polyphonic music," in *Proceddings of the 1985 International Computer Music Conference, Burnaby, B.C., Canada*, International Computer Music Association, 1985.

[39] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2, pp. 357–362, 2012.

[40] M. Abadi, A. Agarwal, and P. Barham, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[41] Z. Demircan, "Flamestudios guitar soundfonts in sf2 format." https://www.hedsound.com/2019/07/flamestudios-guitar-soundfonts-in-sf2.html. (accessed: Apr. 20, 2021).

[42] S. Papel, "Papelmedia final sf2 xxl - trumpet." https://www.polyphone-soundfonts.com/documents/20-brass/199-papelmedia-final-sf2-xxl-trumpet. (accessed: Apr. 20, 2021).

[43] "The hdf5 file library and format." https://www.hdfgroup.org/solutions/hdf5/. (accessed: Apr. 19, 2021).

[44] S.-H. Lim, S. R. Young, and R. M. Patton, "An analysis of image storage systems for scalable training of deep neural networks," in *The seventh workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware (in conjunction with ASPLOS'16), Atlanta, GA, USA*.

[45] C. Raffel, "The lakh midi dataset v0.1." https://colinraffel.com/projects/lmd/. (accessed: Apr. 20, 2021).

[46] C. Raffel, *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[48] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation," in *AI 2006: Advances in Artificial Intelligence* (A. Sattar and B.-h. Kang, eds.), (Berlin, Heidelberg), pp. 1015–1021, Springer Berlin Heidelberg, 2006.

[49] C. R. Harris, K. J. Millman, S. J. van der Walt, and R. Gommers, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[50] V. Studios, "Chamber orchestra 2.1.1: Community edition." https://vis.versilstudios.com/vsco-2.html. (accessed: Apr. 21, 2021).

[51] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *25th European Signal Processing Conference (EUSIPCO)*, (Kos island, Greece), IEEE, 2017.