

# A Tutorial on Deep Learning for Music Information Retrieval

Keunwoo Choi  
keunwoo.choi@qmul.ac.uk

György Fazekas  
g.fazekas@qmul.ac.uk

Kyunghyun Cho  
kyunghyun.cho@nyu.edu

Mark Sandler  
mark.sandler@qmul.ac.uk

## ABSTRACT

Following their success in Computer Vision and other areas, deep learning techniques have recently become widely adopted in Music Information Retrieval (MIR) research. However, the majority of works aim to adopt and assess methods that have been shown to be effective in other domains, while there is still a great need for more original research focusing on music primarily and utilising musical knowledge and insight. The goal of this paper is to boost the interest of beginners by providing a comprehensive tutorial and reducing the barriers to entry into deep learning for MIR. We lay out the basic principles and review prominent works in this hard to navigate field. We then outline the network structures that have been successful in MIR problems and facilitate the selection of building blocks for the problems at hand. Finally, guidelines for new tasks and some advanced topics in deep learning are discussed to stimulate new research in this fascinating field.

## 1. MOTIVATION

In recent years, deep learning methods have become more popular in the field of music information retrieval (MIR) research. For example, while there were only 2 deep learning articles in 2010 in ISMIR conferences <sup>1</sup> ([30], [38]) and 6 articles in 2015 ([123], [91], [97], [65], [36], [10]), it increases to 16 articles in 2016. This trend is even stronger in other machine learning fields, e.g., computer vision and natural language processing, those with larger communities and more competition. Overall, deep learning methods are probably going to serve an essential role in MIR.

There are many materials that focus on explaining deep learning including a recently released book [34]. Some of the material specialises on certain domains, e.g., natural language processing [33]. In MIR, there was a tutorial session in 2012 ISMIR conference, which is valuable but outdated as of 2017.<sup>2</sup>

Much deep learning research is based on shared modules and methodologies such as dense layers, convolutional layers, recurrent layers, activation functions, loss functions, and backpropagation-based training. This makes the knowledge on deep learning generalisable for problems in different domains, e.g., convolutional neural networks were originally

used for computer vision, but are now used in natural language processing and MIR.

For these aforementioned reasons, this paper aims to provide comprehensive basic knowledge to understand how and why deep learning techniques are designed and used in the context of MIR. We assume readers have a background in MIR. However, we will provide some basics of MIR in the context of deep learning. Since deep learning is a subset of machine learning, we also assume readers have understanding of the basic machine learning concepts, e.g., data splitting, training a classifier, and overfitting.

Section 2 describes some introductory concepts of deep learning. Section 3 defines and categorises the MIR problems from the perspective of deep learning practitioners. In Section 4, three core modules of deep neural networks (DNNs) are described both in general and in the context of solving MIR problems. Section 5 suggests several models that incorporate the modules introduced in Section 4. Section 6 concludes the paper.

## 2. DEEP LEARNING

There were a number of important early works on neural networks that are related to the current deep learning technologies. The error backpropagation [84], which is a way to apply the gradient descent algorithm for deep neural networks, was introduced in the 80s. A convolutional neural network (convnet) was used for handwritten digit recognition in [56]. Later, long short-term memory (LSTM) recurrent unit was introduced [42] for sequence modelling. They still remain the foundation of modern DNN algorithms.

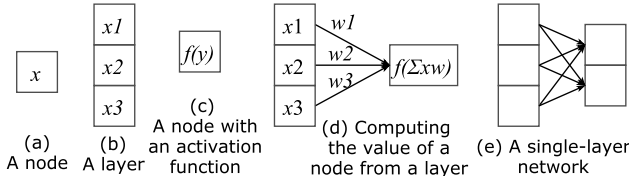
Recently, there have been several advancements that have contributed to the success of the modern deep learning. The most important innovation happened in the optimisation technique. The training speed of DNNs was significantly improved by using rectified linear units (ReLU) instead of sigmoid functions [32] (Section 2.1, Figure 3). This led to innovations in image recognition [53] and speech recognition [22] [122]. Another important change is the advance in hardware. Parallel computing on graphics processing units (GPUs) enabled Krizhevsky et al. to pioneer a large-scale visual image classification in [53].

Let us explain and define several basic concepts of neural networks before we discuss the aspects of deep learning.

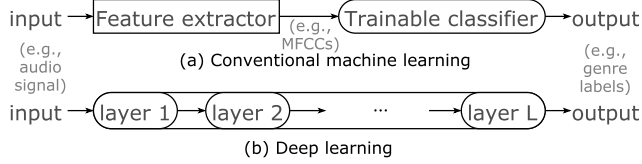
A **node** is analogous to a biological neuron and represents a scalar value as in Figure 1 (a). A **layer** consists of a set of nodes as in Figure 1 (b) and represents a vector. Note that the nodes within a layer are not inter-connected. Usually, an **activation function**  $f()$  is applied to each node as in Figure

<sup>1</sup><http://www.ismir.net>

<sup>2</sup>[http://steinhardt.nyu.edu/marl/research/deep\\_learning\\_in\\_music\\_informatics](http://steinhardt.nyu.edu/marl/research/deep_learning_in_music_informatics)



**Figure 1: Illustrations of (a) a node and (b) a layer. A node often has an nonlinear function called activation function  $f()$  as in (c). As in (d), the value of a node is computed using the previous layer, weights, and an activation function. A single-layer artificial neural network in (e) is an ensemble of (d).**



**Figure 2: Block diagrams of conventional machine learning and deep learning approaches. Trainable modules are in rounded rectangular.**

1 (c). A node may be considered to be activated/deactivated by the output value of an activation function. The value of a node is usually computed as a weighted sum of the input followed by an activation function, i.e.,  $f(w \cdot x)$  where  $w$  is the weights and  $x$  is the inputs as in Figure 1 (d). A simple artificial neural network is illustrated in Figure 1 (e), which is an extension of Figure 1 (d). In a network with multiple layers, there are intermediate layers as well as the input and the output layer which are also called **hidden layers**.

The **depth** of a network is the total number of layers in a network. Similarly, the **width(s)** of a network is the number of nodes in layer(s).

In this paper, deep neural networks (DNNs) indicates neural networks that consist of multiple layers.

## 2.1 Deep learning vs. conventional machine learning

‘Conventional’ machine learning approaches involve hand-designing features and having the machine learn a classifier as illustrated in Figure 2 (a). For example, one can use Mel-frequency cepstrum coefficients (MFCCs), assuming they provide relevant information for the task, then train a classifier (e.g., logistic regression), that maps the MFCCs to the label. Therefore, only a part of the whole procedure (e.g., classifier) is learned while the other (e.g., computing MFCCs) is not data-dependent.

On the contrary, deep learning approaches assume multiple trainable layers, all of which learn from the data, as in Figure 2 (b). Having multiple layers is important because when they are combined with nonlinear activation functions, a network learns complicated relationships between the input and the output, which is not available with a single-layer network. Because of this fully trainable connection, deep learning is also called end-to-end learning. For example, the input and output can be audio signals and genre labels respectively.

## 2.2 Designing and Training Deep Neural Networks

**Designing** a neural network structure involves selecting types and numbers of layers and loss function relevant to the problem. These parameters, that govern the network architecture, are called hyperparameters. Let’s consider neural networks as function approximators  $f : X \rightarrow Y$  with given input  $X$  and output  $Y$  in the data. A network structure constrains the function form; and during designing a network, the constraints are compromised to minimise redundancy in the flexibility while maximising the capacity of the network.

After designing, a network is parametrised by its weights  $w$ . In other words, the output of a network  $\hat{y}$  is a function of input  $x$  and the weights  $w$ . Here, we introduce **loss function**  $J(w)$ , which measures the difference between the predicted output  $\hat{y}$  and the groundtruth output  $y$  with respect to the current weights  $w$ . A loss function is decided so that minimising it would lead to achieving the goal of the task.

**Training** a network is an iterative process of adjusting the weights  $w$  to reduce the loss  $J(w)$ . A loss function provides a way to evaluate the prediction with respect to the true label. A de-facto optimisation method is the gradient descent which iteratively updates the parameters in such a way that the loss decreases most rapidly, as formulated in Eq. 1.

$$w := w - \eta \nabla J(w) \quad (1)$$

where  $\eta$  is the learning rate and  $\nabla J(w)$  is the gradient of  $J(w)$ . As mentioned earlier, in DNNs, gradient descent over multiple layers is called backpropagation [84], which computes the gradient of loss function with respect to the nodes in multiple layers using the chain rule.

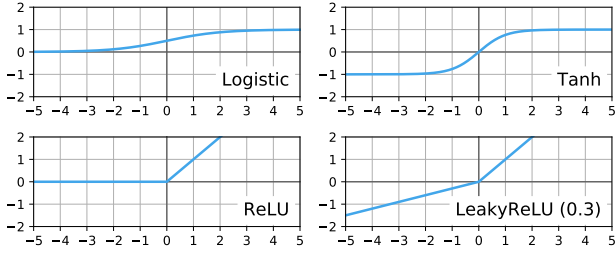
The **training speed** may effect the overall performance because a significant difference on the training speed may result in different convergences of the training. As in Eq. 1, a successful training is a matter of controlling the learning rate  $\eta$  and the gradient  $\nabla J(w)$ .

The **learning rate**  $\eta$  can be adaptively controlled, which is motivated by the intuition that the learning rate should decrease as the loss approaches local minima. As of 2017, ADAM is one of the most popular methods [50]. An overview on adaptive learning rate can be found in an overview article on gradient descent algorithms [83] as well as [34].

The **gradient**  $\nabla J(w)$  has a large effect on the training in DNNs. This is because according to the backpropagation, the gradient at  $l$ -th layer is affected by the gradient at  $l+1$ -th layer, where  $l = 1$  for the input layer and  $l = L$  for the output layer. Therefore, a good *gradient flow* leads to high performance and is the fundamental idea of innovations such as LSTM unit [42], highway networks [101], and deep residual networks [41], all of which are showing the state-of-the-art performances in sequence modelling and visual image recognition.

**Activation functions** play an important role in DNNs. Not only they are analogous to the activation of biological neurons, but also they introduce non-linearity between layers and it enables the whole network to learn more complicated patterns.

Sigmoid functions (e.g., the logistic function,  $f(x) = \frac{1}{1+e^{-x}}$ ) and the hypertangential function,  $f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$ ) were



**Figure 3: Four popular activation functions - a logistic, hypertangential, rectified linear unit (ReLU), and leaky ReLU.**

used in early neural network works until early 2010s. However, a network with sigmoid activation functions may have the ‘vanishing gradient’ problem [5] which impedes training DNNs. The vanishing gradient problem happens when the gradient flow becomes too slow due to a very small gradient,  $\nabla J(w)$  (see [5] for further details).

ReLU was introduced as an alternative to solve the vanishing gradient problem [32]. ReLU has been the first choice in many recent applications in deep learning.

For the output layer, it is recommended to use an activation function that have the same output range to the range of the groundtruth. For example, if the label  $y$  is a probability, Sigmoid function would be most suitable for its output ranges in  $[0, 1]$ . If the task is single-label classification problem, the softmax is preferred because correct gradients can be provided from all the output nodes. If it is a regression problem with an unbounded range, a linear activation function can be used.

### 2.3 Deep or not too deep?

It can be tricky to predict if a deep learning approach would work better than conventional machine learning ones for a given task. This depends on many aspects of the problem such as the dataset size and the complexity of the model.

In general, many deep learning-based MIR researches use datasets that have more than a thousand data samples, e.g., genre classification with Gtzan music genre [106] (1,000 tracks) and music tagging with Million song dataset [7] (million tracks). Korzeniowski et al. [51] used only 383 tracks for chord recognition but the actual number of data samples is much larger than the number of tracks because there are many chord instances in a music track. The minimum amount of data samples also depends on the model complexity, therefore, these numbers only provide rough estimates of the desired dataset size.

When there is not enough data, one can still use DNNs by i) data augmentation, ii) transfer learning, and iii) random weights network (deep network but shallow learning). **Data augmentation** is augmenting the training data by adding some sort of distortion while preserving the core properties, e.g., time stretching and pitch scaling for genre classification, but not for key or tempo detection. **Transfer learning** is reusing a network that is previously trained on a task (as known as source task) for other tasks (as known as target tasks), assuming the source and target tasks are similar so that the trained network can provide relevant representations [18], [111]. In transfer learning, the pre-trained net-

**Table 1: Several MIR problems and their attributes**

Tasks	Subjectivity	Decision time scale
Tempo estimation	Low	Long
Key detection	Low	Long
Onset/offset detection	Low	Short
Beat tracking	Low	Short
Melody extraction	Low	Short
Chord estimation	Low	Short
Structural segmentation	Medium	Short
Music auto-tagging	High	Long
Mood recognition	High	Long

work serves as a feature extractor and a shallow classifier is learned. **Networks without training** (i.e., with randomly initialised weights) have shown that they can provide good representations [43] because the network structure is built based on a strong assumption of the feature hierarchy and therefore the procedure is not completely random but represents some aspects of the input data. As similar to transfer learning, random weights network can serve as a feature extractor, along with a trainable classifier. Features from a convnet with random weights showed reasonable results in many MIR tasks in [18].

## 3. MUSIC INFORMATION RETRIEVAL

MIR is a highly interdisciplinary research field and broadly defined as extracting information from music and its applications [27]. Often music means the audio content, although otherwise its scope extends to other types of musical information e.g., lyrics, music metadata, or user listening history.

### 3.1 MIR problems

This paper discusses how to employ deep learning techniques to solve MIR problems that are related to the audio content. Note that not all the problems can be solved with audio alone. The cultural and social background may be useful to predict some music tags, e.g., era or genre, although the information might be inferred from audio contents. Some tasks are defined completely irrelevant to audio content, e.g., lyrics analysis.

Assuming an audio relevancy, there may be several attributes that specify MIR problems. We focus on two aspects: the subjectivity and the decision time scale, as suggested in Table 1. Many MIR problems are subjective, i.e., a task may be ill-defined and an absolute groundtruth for it may not exist. For example, music genres are rather listeners’ opinions (although they are related to the audio content) and so are music tags, but with a huge vocabulary [54]. Structural segmentation can be done on different hierarchical levels and the groundtruth usually varies by annotators [72]. On the contrary, pitch and tempo are less subjective, although they can be ambiguous, too. Deep learning methods have been achieving good performances for the tasks of both types, and we can explain it from two different perspectives. It is difficult to manually design useful features when we cannot exactly analyse the logic behind subjectivity. In this case, we can exploit the advantage of the data-driven, end-to-end learning to achieve the goal, and furthermore, we can extend our understanding on the domain e.g. analysing relationships of music tags by investigating a trained net-

work [14]. Otherwise, if the logic is well-known, domain knowledge can help effectively structuring the network.

The other property of MIR problems that we focus on in this paper is the decision time scale, which is the unit time length based on which each prediction is made. For example, the tempo and the key are usually static in an excerpt if not in the whole track, which means tempo estimation and key detection are of ‘long’ decision time scale, i.e., time-invariant problems. On the other hand, a melody is often predicted on each time frame which is usually in few tens of millisecond, therefore melody extraction is of ‘short’ decision time scale, i.e., a time-varying problem. Note that this is subject to change depending on the way the problem is formulated. For example, music tagging is usually considered as a time-invariant problem but can be also a time-varying problem [115].

### 3.2 Audio data representations

In this section, we review several audio data representations in the context of using deep learning methods. Majorities of deep learning approaches in MIR take advantage of 2-dimensional representations instead of the original 1-dimensional representation which is the (discrete) audio signal. In many cases, the two dimensions are frequency and time axes.

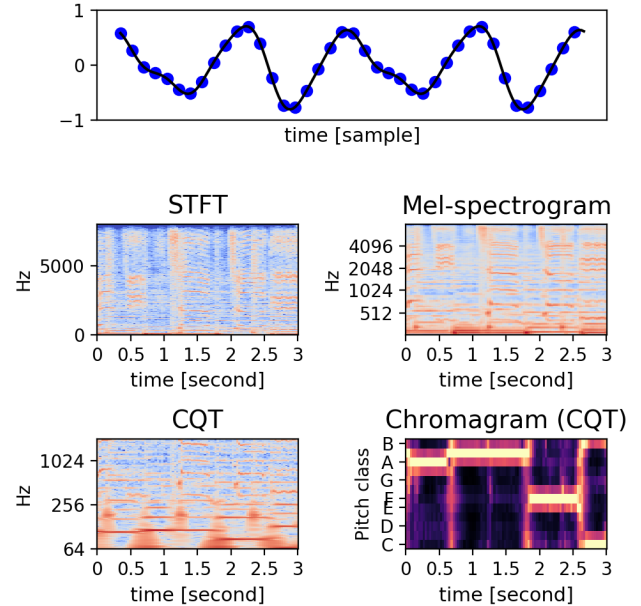
When applying deep learning method to MIR problems, it is particularly important to understand the properties of audio data representations. Training DNNs is computationally intensive, therefore optimisation is necessary for every stage. One of the optimisations is to pre-process the input data so that it represents its information effectively and efficiently – effectively so that the network can easily use it and efficiently so that the memory usage and/or the computation is not too heavy.

In many cases, two-dimensional representations provide audio data in an effective form. By decomposing the signals with kernels of different centre frequencies (e.g., STFT), audio signals are *separated*, i.e., the information of the signal becomes clearer.

Although those 2D representations have been considered as visual images and these approaches have been working well, there are also differences. Visual images are locally correlated; nearby pixels are likely to have similar intensities and colours. In spectrograms, there are often harmonic correlations which are spread along frequency axis while local correlation may be weaker. [67] and [8] focus on the harmonic correlation by modifying existing 2D representations, which is out of the scope in this paper but strongly recommended to read. A scale invariance is expected for visual object recognition but probably not for music/audio-related tasks.

•**Audio signal:** The audio signal is often called *raw* audio, compared to other representations that are transformations based on it. A digital audio signal consists of audio samples that specify the amplitudes at time-steps. In majority of MIR works, researchers assume that the music content is given as a digital audio signal, isolating the task from the effect of acoustic channels. The audio signal has not been the most popular choice; researchers have preferred 2D representations such as STFT and mel-spectrograms because learning a network starting from the audio signal requires even a larger dataset.

Recently, however, one-dimensional convolutions are of-



**Figure 4: Audio content representations.** On the top, a digital audio signal is illustrated with its samples and its continuous waveform part. STFT, mel-spectrogram, CQT, and a chromagram of a music signal are also plotted. Please note the different scales of frequency axes of STFT, melspectrogram, and CQT.

ten used to learn an alternative of existing time-frequency conversions, e.g., in music tagging [26, 59].

•**Short-Time Fourier Transform:** STFT provides a time-frequency representation with linearly-spaced centre frequencies. The computation of STFT is quicker than other time-frequency representations thanks to fast Fourier transform (FFT) which reduces the cost  $O(N^2)$  to  $O(N \log(N))$  with respect to the number of FFT points.

The linear centre frequencies are not always desired in music analysis. They do not match to the frequency resolution of human auditory system, nor musically motivated like the frequencies of CQT. This is why STFT is not the most popular choice in deep learning – it is not efficient in size as melspectrogram and not as raw as audio signals.

One of the merits of STFT is that it is invertible to the audio signal, for which STFT was used in sonification of learned features [16] and source separation [44].

$$S(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]\exp^{-j\omega n} \quad (2)$$

•**Mel-spectrogram:** Mel-spectrogram is a 2D representation that is optimised for human auditory perception. It compresses the STFT in frequency axis and therefore can be more efficient in its size while preserving the most perceptually important information. Mel-spectrogram only provides the magnitude (or energy) of the time-frequency bins, which means it is not invertible to audio signals.

There are other scales that are similar to mel-bands and based on the psychology of hearing – the bark scale, equiv-

alent rectangular bandwidth (ERB), and gammatone filters [75]. They have not been compared in MIR context but in speech research and the result did not show a significant difference on mel/bark/ERB in speech synthesis [117] and mel/bark for speech recognition [94].

There are many suggestions on composing mel-frequencies. [76] suggests the formula as Eq. 3.

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \quad (3)$$

for frequency  $f$  in hertz. Trained kernels of end-to-end configuration have resulted in nonlinear frequencies that are similar to log-scale or mel-scale [26, 59]. Those results agree with the known human perception [75] and indicate that the mel-frequencies are quite suitable for the those tasks, perhaps because tagging is a subjective task. For those empirical and psychological reasons, mel-spectrograms have been popular for tagging [25] [15], [17], boundary detection [108], onset detection [90] and learning latent features of music recommendation [110].

- **Constant-Q Transform (CQT)**: CQT provides a 2D representation with logarithmic-scale centre frequencies. This is well matched to the frequency distribution of the pitch, hence CQT has been predominantly used where the fundamental frequencies of notes should be precisely identified, e.g. chord recognition [45] and transcription [96]. The centre frequencies are computed as Eq. 4.

$$f_c(k_{lf}) = f_{min} \times 2^{k_{lf}/\beta} \quad (4)$$

, where  $f_{min}$ : minimum frequency of the analysis (Hz),  $k_{lf}$ : integer filter index,  $\beta$ : number of bins per octave,  $Z$ : number of octaves.

Note that the computation of a CQT is heavier than that of an STFT or melspectrogram. (As an alternative, log-spectrograms can be used and showed even a better performance than CQT in piano transcription [49].)

- **Chromagram** [4]: The chromagram, also often called the pitch class profile, provides the energy distribution on a set of pitch classes, often with the western music’s 12 pitches.[31] [114]. One can consider a chromagram as a CQT representation folding in the frequency axis. Given a log-frequency spectrum  $X_{lf}$  (e.g., CQT), it is computed as Eq. 5.

$$C_f(b) = \sum_{z=0}^{Z-1} |X_{lf}(b + z\beta)| \quad (5)$$

, where  $z$ =integer, octave index,  $b$ =integer, pitch class index  $\in [0, \beta - 1]$ .

Like MFCCs, chromagram is more ‘processed’ than other representations and can be used as a feature by itself.

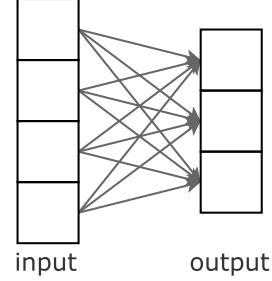
## 4. DEEP NEURAL NETWORKS FOR MIR

In this section, we explain the layers that are frequently used in deep learning. For each type of a layer, a general overview is followed by a further interpretation in the MIR context. Hereafter, we define symbols in layers as in Table 2.

### 4.1 Dense layers

Symbols	Meaning
$N$	Number of channels of a 2D representation
$F$	Frequency-axis length of a 2D representation
$T$	Time-axis length of a 2D representation
$H$	Height of a 2D convolution kernel (frequency-axis)
$W$	Width of a 2D convolution kernel (time-axis)
$V$	Number of hidden nodes of a layer
$L$	Number of layers of a network

**Table 2: Symbols and their meanings defined in this paper. Subscript indicates the layer index, e.g.,  $N_1$  denotes the number of channels (feature maps) in the first convolutional layer.**



**Figure 5: An illustration of a dense layer that has a 4D input and 3D output.**

A dense layer is a basic module of DNNs. Dense layers have many other names - dense layer (because the connection is dense), fully-connected layers (because inputs and outputs are fully-connected), affine transform (because there is  $\mathbf{W} \cdot x + b$  as in Eq. 6), MLP (multi-layer perceptron which is a conventional name of a neural network), and confusingly and unlike in this paper, DNNs (deep neural networks, but to denote deep neural networks only with dense layers). A dense layer is formulated as Eq. 6,

$$y = f(\mathbf{W} \cdot x + b) \quad (6)$$

, where  $x$  and  $b \in \mathbb{R}^{V_{in}}$ ,  $y \in \mathbb{R}^{V_{out}}$ ,  $\mathbf{W} \in \mathbb{R}^{V_{in} \times V_{out}}$ , and each corresponds to input, bias, output, and the weight matrix, respectively.  $f()$  is a nonlinear activation function (Sec 2.2 for details).

The input to a dense layer with  $V$  nodes is transformed into a  $V$ -dimensional vector. In theory, a single node can represent a huge amount of information as long as the numerical resolution allows. In practice, each node (or dimension) often represents a certain semantic aspect. Sometimes, a narrow layer (a layer with a small  $V$ ) can work as a bottleneck of the representation. For networks in many classification and regression problems, the dimension of output is smaller than that of input, and the widths of hidden layers are decided between  $V_{out}$  and  $V_{in}$ , assuming that the representations become more compressed, in higher-levels, and more relevant to the prediction in deeper layers.

### 4.2 Dense layers and music

In MIR, a common usage of a dense layer is to learn a frame-wise mapping as in (d1) and (d2) of Figure 8. Both are non-linear mappings but the input is multiple frames in d2 in order to include contextual information around the

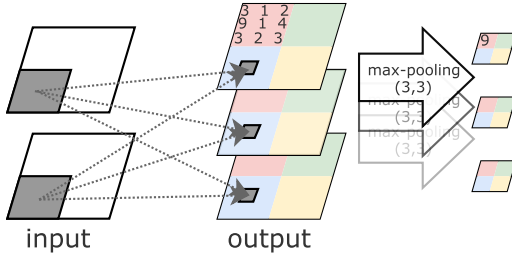


Figure 6: An illustration of a convolutional layer in details, where the numbers of channels of input/output are 2 and 3, respectively. The dotted arrows represent a convolution operation in the region, i.e., a dot product between convolutional kernel and local regions of input.

centre frame. By stacking dense layers on the top of a spectrogram, one can expect that the network will learn how to reshape the frequency responses into vectors in another space where the problem can be solved more easily (the representations becomes *linearly separable*<sup>3</sup>). For example, if the task is pitch recognition, we can expect the first dense layer be trained in such a way that its each output node represents different pitch.<sup>4</sup>

By its definition, a dense layer does not facilitate a shift or scale invariance. For example, if a STFT frame length of 257 is the input of a dense layer, the layer maps vectors from 257-dimensional space to another  $V$ -dimensional space. This means that even a tiny shift in frequency, which we might hope the network be invariant to for certain tasks, is considered to be a totally different representation.

Dense layers are mainly used in early works before convnets and RNNs became popular. It was also when the learning was not less of end-to-end for practical reasons (computation power and dataset size). Instead of the audio data, MFCCs were used as input in genre classification [98] and music similarity [37]. A network with dense layers was trained in [51] to estimate chroma features from log-frequency STFTs. In [107], a dense-layer network was used for source separation. Recently, dense layers are often used in hybrid structures; they were combined with Hidden Markov model for chord recognition [24] and downbeat detection [28], with recurrent layers for singing voice transcription [82], with convnet for piano transcription [49], and on cepstrum and STFT for genre classification [48].

### 4.3 Convolutional layers

The operation in convolution layers can be described as Eq. 7.

$$y^j = f\left(\sum_{k=0}^{K-1} \mathbf{W}^{jk} * x^k + b^j\right) \quad (7)$$

, where all  $y^j$ ,  $\mathbf{W}^{jk}$ ,  $x^k$ , and  $b^j$  are 2-dimensional and the

<sup>3</sup><http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/> for a further explanation and demonstration.

<sup>4</sup>See example 1 on [https://github.com/keunwoochoi/mir-deepnet\\_tutorial](https://github.com/keunwoochoi/mir-deepnet_tutorial), a simple pitch detection task with a dense layer.

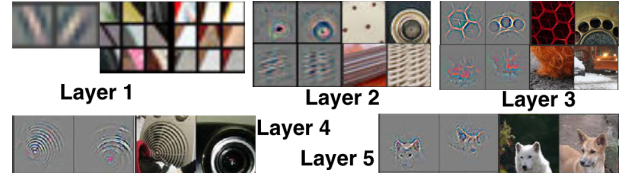


Figure 7: Feature visualisations by deconvolution [121] of a network trained for visual image recognition. For each layer, features are visualised on the left and the corresponding parts of input images that has high activation of the features are on the right.

superscripts denote the channel indices.  $y^j$  is  $j$ -th channel output,  $x^k$  is  $k$ -th channel input,  $*$  is convolution operation,  $\mathbf{W}^{jk} \in \mathbb{R}^{h \times l}$  is the convolution kernel which associates  $k$ -th input channel and  $j$ -th output channel, and  $b^j$  is the bias for  $j$ -th output channel. The total weights ( $\mathbf{W}^{jk}$  for all  $j$  and  $k$ ) are in a 4-dimensional array with a shape of  $(h, l, K, J)$  while  $x$  and  $y$  are 3-dimensional arrays including channel indices (axes for height, width, and channel).

A 2D convolutional kernel ‘sweeps’ over an input and this is often described as ‘weight sharing’, indicating that the same weights (convolutional kernels) are applied to the whole input area. This results in vastly reducing the number of trainable parameters.

For a given input, as a result of the sweeping, the convolutional layers output an representation of local activations of patterns. This representation is called *feature map* ( $y$  in Eq. 7). As a result, unlike dense or recurrent layers, convolutional layers preserves the spatiality of the input.

In essence, the convolution computes a local correlation between the kernel and input. During training, the kernels learn local patterns that are useful to reduce the loss. With many layers, kernels can learn to represent some complex patterns that combines the patterns detected in the previous layer. Figure 7 is a visualisation of a convnet for image classification and presented to help to understand the mechanism of convnets [121]. For each layer, the grey images on the left are one of the 2D kernel ( $\mathbf{W}^{jk}$ ) and the images on the right are corresponding (cropped) input images that show high activations on the feature map. It clearly illustrates the feature hierarchy in the convnet.

#### 4.3.1 Subsampling

Convolutional layers are very often used with pooling layers. A pooling layer reduces the size of feature maps by downsampling them with an operation, usually the *max* function (Figure 6). Using max function assumes that on the feature maps, what really matters is if there exists an activation or not in a local region, which can be captured by the local maximum value. This non-linear subsampling provides distortion and translation invariances because it discards the precise location of the activations.

The average operation is not much used in pooling except in a special case where it is applied *globally* after the last convolutional layer [66]. This global pooling is used to summarise the feature map activation on the whole area of input, which is useful when input size varies, e.g., [2].

### 4.4 Convolutional layers and music

Figure 8 (c1) and (c2) show how 1D and 2D convolutions



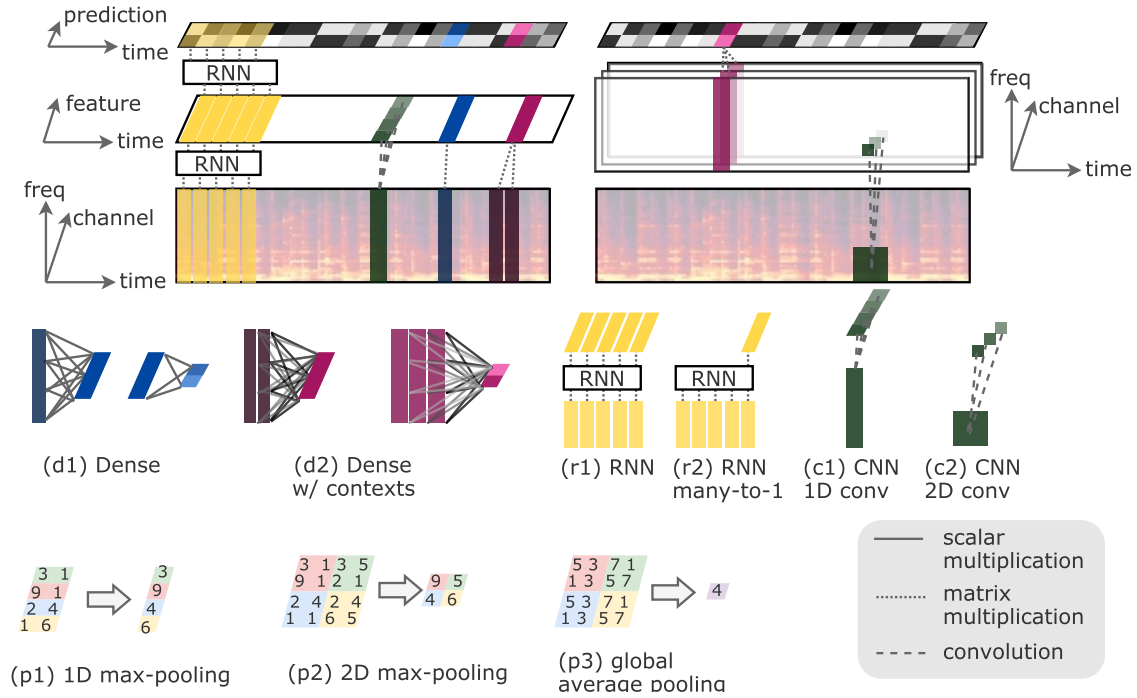


Figure 8: Neural network layers in the context of MIR

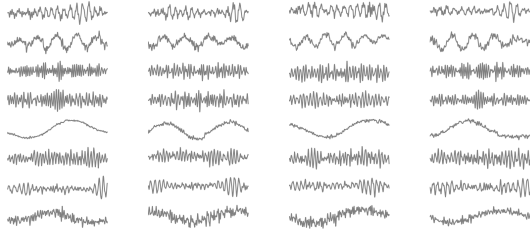


Figure 9: Learned 1D convolutional kernels that are applied to audio samples in [26] for music tagging

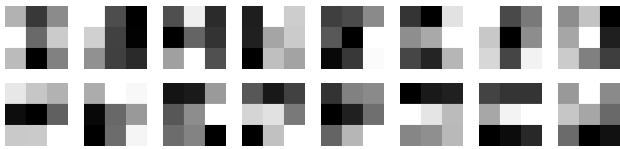


Figure 10: Learned  $3 \times 3$  convolutional kernels at the first layer of a convnet used in [18]. These kernels are applied to log-melspectrograms for music tagging.

are applied to a spectrogram. By stacking convolutional layers, a convnet can learn more complicated pattern from music content [19, 16].

The convolutional layer applied to the input data provides some insights of the mechanism underneath the convnet. Figure 9 illustrates the learned 1D convolutional kernels that are applied to raw audio samples in [26]. The kernels learned various fundamental frequencies. A more common approach is to apply 2D convolutional layers to 2D time-frequency rep-

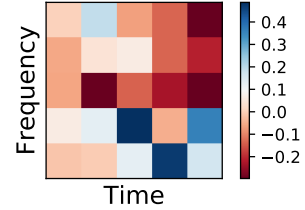


Figure 11: A 2D convolutional kernel in chord recognition network [70]. The right three columns work as a horizontal edge detector, which is explained as a ‘harmonic saliency enhancer’ in [70].

resentations. Figure 10 illustrates a subset of the 2D kernels that are applied to melspectrograms for music tagging problem [15]. Kernels size of 3-by-3 learned vertical edge detectors (on the top row) and horizontal edge detectors (on the lower row). As in the convnets in other domains, these first-layer kernels are combined to create more complex features. Another example is in Figure 11, where a single-channel 2D convolutional layer is applied to the input to enhance the harmonic saliency. The input is 3 bins/note CQT and after an end-to-end training, the kernel learned a horizontal edge detector which can work for ‘thick’ edges, which would exist when the frequency resolution is 3 bins/note.

The kernel size determines the maximum size of a component that the kernel can precisely capture in the layer. How small can a kernel be in solving MIR tasks? The layer would fail to learn a meaningful representation if the kernel is smaller than the target pattern. For example, for a chord recognition task, a kernel should be big enough to capture the difference between major and minor chords. For this

reason, relatively large-sized kernels such as  $17 \times 5$  are used on 36-bins/octave CQT in [45]. A special case is to use different shapes of kernels in the same layer as in the Inception module [104] which is used for hit song prediction in [118].

The second question would be then how big can a kernel be? One should note that a kernel does not allow an invariance within it. Therefore, if a large target pattern may slightly vary inside, it would better be captured with stacked convolutional layers with subsamplings so that small distortions can be allowed. More discussions on the kernel shapes for MIR research are available in [80, 79].

Max-pooling is frequently used in MIR to add time/frequency invariant. Such a subsampling is necessary in the DNNs for time-invariant problems in order to yield a one, single prediction for the whole input. In this case, One can begin with specifying the size of target output, followed by deciding details of poolings (how many and how much in each stage) somehow empirically.

A special use-case of the convolutional layer is to use 1D convolutional layers directly onto an audio signal (which is often referred as a raw input) to learn the time-frequency conversions in [26], and furthermore, [60]. This approach is also proposed in speech/audio and resulted in similar kernels learned of which fundamental frequencies are similar to log- or mel-scale frequencies [85]. Figure 9 illustrates a subset of trained kernel in [26]. Note that unlike STFT kernels, they are not pure sinusoid and include harmonic components.

The convolutional layer has been very popular in MIR. A pioneering convnet research for MIR is convolutional deep belief networks for genre classification [58]. Early works relied on MFCC input to reduce computation [62], [63] for genre classification. Many works have been then introduced based on time-frequency representations e.g., CQT for chord recognition [45], guitar chord recognition [46], genre classification [116], transcription [96], melspectrogram for boundary detection [89], onset detection [90], hit song prediction [118], similarity learning [68], instrument recognition [39], music tagging [26], [15], [17], [59], and STFT for boundary detection [36], vocal separation [100], and vocal detection [88]. One-dimensional CNN for raw audio input is used for music tagging [26], [60], synthesising singing voice [9], polyphonic music [112], and instruments [29].

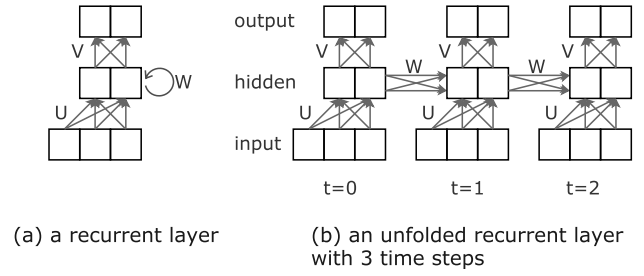
## 4.5 Recurrent layers

A recurrent layer incorporates a recurrent connection and is formulated as Eq. 8.

$$\begin{aligned} y_t &= f_{out}(\mathbf{V}h_t) \\ h_t &= f_h(\mathbf{U}x_t + \mathbf{W}h_{t-1}) \end{aligned} \quad (8)$$

, where  $f_h$  is usually tanh or ReLU,  $f_{out}$  can be softmax/sigmoid/etc.,  $h_t$ : hidden vector of the network that stores the information at time  $t$ , and  $\mathbf{U}, \mathbf{V}, \mathbf{W}$  are matrices which are trainable weights of the recurrent layer. To distinguish the RNN with this formula from other variants of RNNs, it is often specified as vanilla RNN.

An easy way to understand recurrent layers is to build them up from dense layers. If the networks in Figure 12 (b) are isolated by each time step (i.e., if  $\mathbf{W}$  is disconnected), the network becomes a feed-forward network with two dense layers which are parametrised with two matrices,  $\mathbf{V}$  and  $\mathbf{U}$ . Let's further assume that there are three data samples, each of which is a pair ( $x \in \mathbb{R}^3, y \in \mathbb{R}^2$ ) and is fed to the isolated



**Figure 12: Illustrations of a recurrent layer as (a) folded and (b) unfolded, with the dimensionality of input/hidden/output as  $3/2/2$ . Note that  $\mathbf{W} \in \mathbb{R}^{2 \times 2}$  and fully-connects the hidden nodes between time-steps.**

network one by one. In this case, only the relationship between  $x$  and  $y$  for data sample is modelled by  $V$  and  $U$  and one is assuming that there is no relationship between data samples, e.g., input at  $t = 0$  is not related to the outputs at  $t \neq 0$ . By connecting them with the recurrent connection  $\mathbf{W}$ , the relationships between  $x$  at  $t = 0$  and  $y$ 's at  $t = 0, 1, 2$  are modelled by  $U, V, W$ . In other words, the network learns how to update the 'memory' from the previous hidden state ( $h_{t-1}$ ) to the current hidden state ( $h_t$ ) which is then used to make a prediction ( $y_t$ ).

In short, recurrent layer models  $p(y_t|x_{t-k}, \dots, x_t)$ . This is similar to the goal of HMMs (hidden Markov models). Compared to HMMs which consist of 1-of- $n$  states, RNNs are based on hidden states that consist of continuous value and therefore scale with a large amount of information.

In practice, modified recurrent units with gates have been widely used. A **gate** is a vector-multiplication node where the input is multiplied by a same-sized vector to attenuate the amount of input. Gated recurrent networks usually use long short-term memory units (LSTM, [42]) and gated recurrent unit (GRU, [12]). In LSTM units, the gates control how much read/write/forget from the memory  $h$ . Moreover, additive connections between time-steps help gradient flow, remedying the vanishing gradient problem [5]. RNNs with gated recurrent units have been achieving state-of-the-art results in many sequence modelling problems such as machine translation [3] and speech recognition [86] as well as MIR problems, e.g., singing voice detection [61].

Sometimes, an RNN has another set of hidden nodes that are connected by  $W$  but with a reversed direction. This is called a bi-directional RNN [93] and can model the recurrence both from the past and the future, i.e.,  $p(y_t|x_{t-k}, \dots, x_t)$  and  $p(y_t|x_{t+1}, \dots, x_{t+k})$ . As a result, the output is a function of its past and future inputs. An intuitive way of understanding it is to imagine another recurrent layer in parallel that works in the reversed time order.

## 4.6 Recurrent layers and music

Since the input is fully-connected to the hidden layer with  $U$  in a recurrent layer, a recurrent layer can replace with a dense layer with contextual inputs. Figure 8 - r1 is a many-to-many recurrent layer with applied to a spectrogram while r2 is a many-to-one recurrent layer which can be used at the final layer.

Since the shift invariance cannot be incorporated in the computation inside recurrent layers, recurrent layers may be



suitable for the sequences of features. The features can be either known music and audio features such as MFCCs or feature maps from convolutional layers [17].

All the inputs are sequentially transformed and summed to a  $V$ -dimensional vector, therefore it should be capable of containing enough information. The size,  $V$ , is one of the hyperparameter and choosing it involves many trials and comparison. Its initial value can be estimated by considering the dimensionality of input and output. For example,  $V$  can be between their sizes, assuming the network learns to compress the input while preserving the information to model the output.

One may want to control the length of a recurrent layer to optimise the computational cost. For example, on the onset detection problem, probably only a few context frames can be used since onsets can be specified in a very short time, while chord recognition may benefit from longer inputs.

Many time-varying MIR problems are time-aligned, i.e., the groundtruth exists for a regular rate and the problem does not require sequence matching techniques such as dynamic time warping. This formulation makes it suitable to simply apply ‘many-to-many’ recurrent layer (Figure 8 - r1). On the other hand, classification problems such as genre or tag only have one output prediction. For those problems, the ‘many-to-one’ recurrent layer (Figure 8 - r2) can be used to yield only one output prediction at the final time step. More details are in Section 5.2

For many MIR problems, inputs from the future can help the prediction and therefore bi-directional setting is worth trying. For example, onsets can be effectively captured with audio contents before and after the onsets, and so are off-sets/segment boundaries/beats.

So far, recurrent layers have been mainly used for time-varying prediction; for example, singing voice detection [61], singing and instrument transcription [82], [96] [95], and emotion prediction [64]. For time-invariant tasks, a music tagging algorithm used a hybrid structure of convolutional and recurrent layers [17].

## 5. SOLVING MIR PROBLEMS: PRACTICAL ADVICE

In this section, we focus on more practical advice by presenting examples/suggestions of deep neural network models as well as discussing several practical issues.

### 5.1 Data preprocessing

Preprocessing input data is very important to effectively train neural networks. One may argue that the neural networks can learn any types of preprocessing. However, adding trainable parameters always requires more training data, and some preprocessing such as standardisation substantially effects the training speed [57].

Furthermore, audio data requires some designated preprocessing steps. When using the magnitudes of 2D representations, logarithmic mapping of magnitudes ( $\mathbf{X} \rightarrow \log(\mathbf{X} + \epsilon)$ ) is widely used to condition the data distributions and often results in better performance [13]. Besides, preprocessing audio data is an open issue yet. Spectral whitening can be used to compensate the different energy level by frequencies [97]. However, it did not improve the performance on music tagging convnet [13]. With 2D convnet, it seems not helpful to normalise local contrasts in computer vision, which may

be applicable for convnet in MIR as well.

Lastly, one may want to optimise the signal processing parameters such as the numbers of FFT and mel-bins, window and hop sizes, and the sampling rate. As explained in Section 3.2, it is important to minimise the data size for an efficient training. To reduce the data size, audio signals are often downmixed and downsampled to 8-16kHz. After then, one can try real-time preprocessing with utilities such as Kapre [20], Pescador<sup>5</sup>, Fuel[113], Muda [71], otherwise pre-computing and storing can be an issue in practice.

### 5.2 Aggregating information

The time-varying/time-invariant problems need different network structures. As in Table 1, problems with a short decision time scale, or time-varying problems, require a prediction per unit time, often per short time frame. On the contrary, for problems with a long decision time scale, there should be a method implemented to aggregate the features over time. Frequently used methods are *i)* pooling, *ii)* strided convolutions, and *iii)* recurrent layers with many-to-one configuration.

**i) Pooling:** With convolutional layers, a very common method is to use max-pooling layers(s) over time (and often as well as frequency) axis (also discussed in Section 4.3.1). A special case is to use a global pooling after the last convolutional layer [66] as in Figure 8.

**ii) Strided convolutions:** Strided convolutions are the operations of convolutional layers that have strides larger than 1. The effects of this are known to be similar to max-pooling, especially under the generic and simple supervised learning structures that are introduced in this paper. One should be careful to set the strides to be smaller than the convolutional kernel sizes so that all part of the input is convolved.

**iii) Recurrent layers:** Recurrent layers can learn to summarise features in any axis. They involve trainable parameters and therefore take more computation and data to do the job than the previous two approaches. The previous two can reduce the size gradually, but recurrent layers are often set to many-to-one. Therefore, it was used in the last layer rather than intermediate layers in [17].

### 5.3 Depth of networks

In designing a network, one may find it arbitrary and empirical to decide the depth of the network. The network should be deep enough to approximate the relationship between the input and the output. If the relationship can be (roughly) formulated, one can start with a depth with which the network can implement the formula. Fortunately, it is becoming easier to train a very deep network [101, 41, 40].

For convnets, the depth is increasing in MIR as well as other domains. For example, networks for music tagging, boundary detection, and chord recognition in 2014 used 2-layer convnet [26], [90], [46], but recent research often uses 5 or more convolutional layers [15, 68, 52].

The depth of RNNs has been increasing slowly. This is a general trend including MIR and may because *i)* stacking recurrent layers does not incorporate feature hierarchy and *ii)* a recurrent layer already are deep due to the recurrent connection, i.e., it is deep along the time axis, therefore the number of layers is less critical than that of convnets.

<sup>5</sup><http://pescador.readthedocs.io>

## 5.4 First layer to input

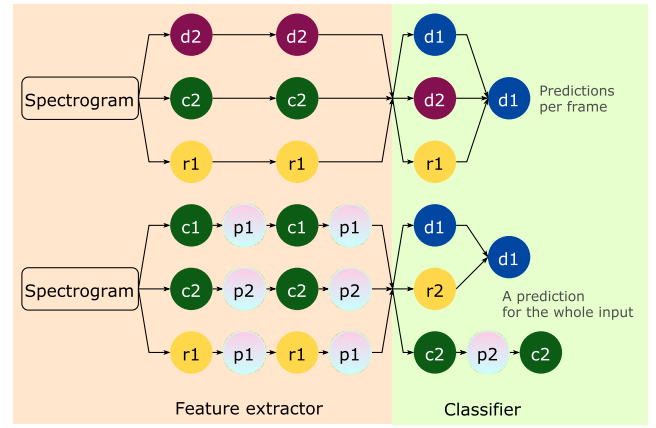
- **d1**: As mentioned earlier, dense layers are not frequency shift invariant. In the output, dense layers remove the spatiality along frequency axis because the whole frequency range is mapped into scalar values.
  - **d2**: The operation, a matrix multiplication to input, is the same as **d1** but it is performed with multiple frames as an input.
  - **c1**: With  $F$ -by-1 convolution kernels, the operation is equivalent to **d1**.  $F$ -by- $W$  kernels with  $W > 1$  are also equivalent to **d2**.
  - **c2**: Many recent DNN structures use a 2-dimensional convolutional layer to the spectrogram inputs. The layer outputs  $N$  feature maps which preserve the spatiality on both axes.
- A special usage of a 2D convolutional layer is to use it as a preprocessing layer, hoping to enhance the saliency of some pattern as in [70], where  $5 \times 5$  kernels compress the transient for a better chord recognition. Depending on the task, it can also enhance the transient [16] or do the both – to approximate a harmonic-percussive separation.
- **r1**: It maps input frames to another dimension, which is same as **d1**. Since recurrent layer can take many nearby frames, the assumption of using **r1** is very similar to that of **d2**.

## 5.5 Intermediate layers

- **d1, d2, c1, r1** : Repeating these layers adds more non-linearity to the model.
- Note that Stacking layers with context enables the network to ‘look’ even wider contexts. For example, imagine dense layers, both are taking adjacent 4 frames (2 from the past and 2 from the future), then  $y_2[t]$ , the output of the second layer, is a function of  $x_2[t - 2 : t + 2] = y_1[t - 2 : t + 2]$ , which is a function of  $x_1[t - 4 : t + 4]$ .
- **c2**: Many structures consist of more than one convolutional layers. As in the case above, stacking convolutional layers also enables the network to look larger range.
  - **p1, p2**: A pooling layer often follows convolutional layers as mentioned in Section 4.3.1.

## 5.6 Output layers

Dense layers are almost always used in the output layer, where the number of node  $V$  is set to the number of classes in classification problem or the dimensionality of the predicted values in the regression problem. For example, if the task is to classify the music into 10 genres, a dense layer with 10 nodes (with a softmax activation function) can be used where each node represents a probability for each genre and the groundtruth is given as one-hot-vector.



**Figure 13: Model suggestions for time-varying and time-invariant problems. Note that **d1** and **d2** can be used interchangeably.**

## 5.7 Model suggestions

In Figure 13, two flowcharts illustrate commonly used structures to solve time-varying (top) and time-invariant (lower) MIR problems. As already mentioned in Section 4, there is no strict boundary between a feature extractor and a classifier.

The structures in Figure 13 only include two layers (pooling layers usually do not count) in the feature extractor stage and one layer in the classifier stage. In reality, the depth can scale depending on the problem and the size of datasets.<sup>6</sup>

### 5.7.1 Time-varying problems

- **DNN**: **d2 - d2 - d2 - d1**: This structure proposed to learn chroma feature in [51], where 15 consecutive frames were input to the 3-layer DNN and output consisted of 12 nodes (for each pitch class). Because the task was to distinguish pitches, the network should not be pitch invariant, and that is why dense layer was used instead of convolutional layers. Since polyphony is assumed, it becomes a multi-label classification problem. Therefore, sigmoid activation function is used at the output layer and each output node represents the chromagram value in  $[0, 1]$ .
- **Conv2D**: **c2 - c2 - p1 - c2 - c2 - p1 - c2 - p1 - d1 - d1 - d1**: For singing voice detection, Schluter adopted 5-layer convnet with 2D kernels as well as following 3 dense layers (we do not consider batch normalization layer [47] here) in [88]. Since the problem is a binary classification (whether there is voice or not), output layer has a single node with sigmoid activation function denoting 1 for True and 0 for False.

The demonstration in the conference (ISMIR 2016) showed that the trained network responds to frequency modulations of voice. The pitch of voice varies, and therefore the pitch invariance of convolutional layers fits well to the problem.

- **Bidirectional RNN**: **r1 - r1 - d1**: Deep bidirectional LSTM was used for singing voice detection in

<sup>6</sup>Implementations of all the models will be online.

[61]. The depth affected the performance and 4-layer network achieved the best performance among [2, 3, 4, 5]-layer architectures. Although [61] did not compare the result without a bi-directional configuration, we can guess it probably helped, because the vocal existence at time  $t$  is probably not independent of the adjacent frames.

In [61], the preprocessing stage includes harmonic-percussive separation [105] to enhance the vocal melody. One can add other preprocessing steps that are relevant to the task, although it is not very common in practice due to a heavy computation.

### 5.7.2 Time-invariant problems

- **Conv1d:** c1 - p1 - c1 - p1 - d1 - d1 - d1

The 1D convolution at the first layer reduces the size drastically ( $F \times T \rightarrow 1 \times T$ , since the height of convolutional kernel is same as  $F$ ). Therefore, this model is computational efficient [17]. It is one of the early structures that were used in MIR, e.g., music tagging [26]. Note that the pooling is performed only along time axis because feature-axis does not imply any spatial meaning.

The limit of this model comes from the large kernel size; since convolutional layers do not allow variances within the kernels, the layer is not able to learn local patterns ('local' in frequency axis).

- **Conv2d:** c2 - p2 - c2 - p2 - c2 - p2 - d2: By stacking 2D convolutions and subsamplings, the network can see the input spectrograms at different scales. The early layers learns relevant local patterns, which is combined in the deeper layers. As a result, the network covers the whole input range with small distortions allowed.

Convnets with 2D convolutions have been used in many classification and regression tasks including music tagging [15], onset detection [90], boundary detection [108], and singing voice detection [88].

- **CRNN:** c2 - p2 - c2 - p2 - r1 - r2 - d1

This structure combines convolutional and recurrent layers [17]. The early convolutional layers capture local patterns and pooling layers reduce the size to some extent. At the end, the recurrent layer summarise the feature maps.

The benchmark in [17] showed that the network benefit from the flexibility of recurrent layers in summarising information along time, achieving the best performance among the structures. CRNN was also used in music emotion recognition and achieved state-of-the-art performance [69].

## 6. CONCLUSIONS

In this paper, we presented a tutorial on deep learning for MIR research. Using deep learning to solve a problem is more than simply loading the data and feed it to the network. Due to heavy computation, exhaustive search for the hyperparameters of a network is not a viable option,

therefore one should carefully decide the structure with understanding both the domain knowledge and deep learning techniques.

We reviewed the basics of deep learning – what is deep learning and designing a structure, how to train it, and when to use deep learning. We also reviewed MIR, focusing on the essential aspects for using deep learning. Then we summarised three popular layers - dense, convolutional, and recurrent layers as well as their interpretations in the MIR context. Based on the understanding of the layers, one can design a new network structure for a specific task. Finally, we summarised more practical aspects of using deep learning for MIR including popular structures with their properties. Although the structures were proposed to a certain problem, they can be broadly used for other problems as well.

Deep learning has achieved many state-of-the-art results in various domains and even surpassing human level in some tasks such as image recognition [41] or playing games [99]. We strongly believe that there still is a great potential for deep learning and there will be even more MIR research relying on deep learning methods.

## Acknowledgements

This work has been part funded by FAST IMPACT EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons (688382). Mark Sandler acknowledges the support of the Royal Society as a recipient of a Wolfson Research Merit Award. Kyunghyun Cho thanks the support by eBay, TenCent, Facebook, Google and NVIDIA.

We appreciate Adib Mehrabi, Beici Liang, Delia Fanoyela, Blair Kaneshiro, and Sertan Şentürk for their helpful comments on writing this paper.

## 7. REFERENCES

- [1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
- [2] Y. Aytaç, C. Vondrick, and A. Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*, pages 892–900, 2016.
- [3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, 2014.
- [4] J. P. Bello. Chroma and tonality. *Music Information Retrieval Course*.
- [5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
- [6] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.
- [7] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596. University of Miami, 2011.
- [8] R. Bittner, B. McFee, J. Salamon, P. Li, and J. Bello. Deep salience representations for  $f_0$  estimation in polyphonic music. In *18th International Society for*

- Music Information Retrieval Conference, ISMIR*, 2017.
- [9] M. Blaauw and J. Bonada. A neural parametric singing synthesizer. *arXiv preprint arXiv:1704.03809*, 2017.
  - [10] S. Böck, F. Krebs, and G. Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *ISMIR*, pages 625–631, 2015.
  - [11] L. Chen, S. Srivastava, Z. Duan, and C. Xu. Deep cross-modal audio-visual generation. *arXiv preprint arXiv:1704.08292*, 2017.
  - [12] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.
  - [13] K. Choi, G. Fazekas, K. Cho, and M. Sandler. A comparison on audio signal preprocessing methods for deep neural networks on music tagging. *arXiv:1709.01922*, 2017.
  - [14] K. Choi, G. Fazekas, K. Cho, and M. Sandler. The effects of noisy labels on deep convolutional neural networks for music classification. *arXiv:1706.02361*, 2017.
  - [15] K. Choi, G. Fazekas, and M. Sandler. Automatic tagging using deep convolutional neural networks. In *The 17th International Society of Music Information Retrieval Conference, New York, USA*. International Society of Music Information Retrieval, 2016.
  - [16] K. Choi, G. Fazekas, and M. Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.
  - [17] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2016.
  - [18] K. Choi, G. Fazekas, M. Sandler, and K. Cho. Transfer learning for music classification and regression tasks. In *The 18th International Society of Music Information Retrieval (ISMIR) Conference 2017, Suzhou, China*. International Society of Music Information Retrieval, 2017.
  - [19] K. Choi, G. Fazekas, M. Sandler, and J. Kim. Auralisation of deep convolutional neural networks: Listening to learned features. *ISMIR late-breaking session*, 2015.
  - [20] K. Choi, D. Joo, and J. Kim. Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras. In *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.
  - [21] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *arXiv preprint arXiv:1603.00982*, 2016.
  - [22] G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.
  - [23] M. Defferrard. Structured auto-encoder with application to music genre recognition. Technical report, 2015.
  - [24] J. Deng and Y.-K. Kwok. Automatic chord estimation on seventhsbass chord vocabulary using deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 261–265. IEEE, 2016.
  - [25] S. Dieleman and B. Schrauwen. Multiscale approaches to music audio feature learning. In *ISMIR*, pages 3–8, 2013.
  - [26] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6964–6968. IEEE, 2014.
  - [27] J. S. Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.
  - [28] S. Durand, J. P. Bello, B. David, and G. Richard. Downbeat tracking with multiple features and deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 409–413. IEEE, 2015.
  - [29] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.
  - [30] F. Eyben, S. Böck, B. W. Schuller, and A. Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *Ismir*, pages 589–594, 2010.
  - [31] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*, pages 464–467, 1999.
  - [32] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.
  - [33] Y. Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.(JAIR)*, 57:345–420, 2016.
  - [34] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
  - [35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
  - [36] T. Grill and J. Schlüter. Music boundary detection using neural networks on spectrograms and self-similarity lag matrices. In *Proceedings of the 23rd European Signal Processing Conference (EUSPICO 2015), Nice, France*, 2015.
  - [37] P. Hamel, M. E. Davies, K. Yoshii, and M. Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. *14th International Conference on Music Information Retrieval*, 2013.
  - [38] P. Hamel and D. Eck. Learning features from music

- audio with deep belief networks. In *ISMIR*, pages 339–344. Utrecht, The Netherlands, 2010.
- [39] Y. Han, J. Kim, and K. Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221, 2017.
- [40] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [42] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [43] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.
- [44] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *ISMIR*, pages 477–482, 2014.
- [45] E. J. Humphrey and J. P. Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications, 11th International Conference on*, volume 2, pages 357–362. IEEE, 2012.
- [46] E. J. Humphrey and J. P. Bello. From music audio to chord tablature: Teaching deep convolutional networks to play guitar. In *Acoustics, Speech and Signal Processing, IEEE International Conference on*, pages 6974–6978. IEEE, 2014.
- [47] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [48] I.-Y. Jeong and K. Lee. Learning temporal features using a deep neural network and its application to music genre classification. *Proc. of the 17th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2016.
- [49] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the potential of simple framewise approaches to piano transcription. In *International Society of Music Information Retrieval (ISMIR), New York, USA*, 2016.
- [50] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [51] F. Korzeniowski and G. Widmer. Feature learning for chord recognition: The deep chroma extractor. *arXiv preprint arXiv:1612.05065*, 2016.
- [52] F. Korzeniowski and G. Widmer. A fully convolutional deep auditory model for musical chord recognition. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*, pages 1–6. IEEE, 2016.
- [53] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [54] P. Lamere. Social tagging and music information retrieval. *Journal of new music research*, 37(2):101–114, 2008.
- [55] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [56] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [57] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
- [58] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.
- [59] J. Lee and J. Nam. Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging. *arXiv preprint arXiv:1703.01793*, 2017.
- [60] J. Lee, J. Park, K. L. Kim, and J. Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *arXiv preprint arXiv:1703.01789*, 2017.
- [61] S. Leglaive, R. Hennequin, and R. Badeau. Singing voice detection with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 121–125. IEEE, 2015.
- [62] L. Li. Audio musical genre classification using convolutional neural networks and pitch and tempo transformations. 2010.
- [63] T. L. Li, A. B. Chan, and A. Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*, 2010.
- [64] X. Li, H. Xianyu, J. Tian, W. Chen, F. Meng, M. Xu, and L. Cai. A deep bidirectional long short-term memory based multi-scale approach for music dynamic emotion prediction. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 544–548. IEEE, 2016.
- [65] D. Liang, M. Zhan, and D. P. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *ISMIR*, pages 295–301, 2015.
- [66] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- [67] V. Lostanlen and C.-E. Cella. Deep convolutional networks on the pitch spiral for musical instrument recognition. *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 2016.
- [68] R. Lu, K. Wu, Z. Duan, and C. Zhang. Deep ranking: triplet matchnet for music metric learning. 2017.
- [69] M. Malik, S. Adavanne, K. Drossos, T. Virtanen, D. Ticha, and R. Jarina. Stacked convolutional and

- recurrent neural networks for music emotion recognition. *arXiv preprint arXiv:1706.02292*, 2017.
- [70] B. McFee and J. Bello. Structured training for large-vocabulary chord recognition. In *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [71] B. McFee, E. J. Humphrey, and J. P. Bello. A software framework for musical data augmentation. In *ISMIR*, pages 248–254, 2015.
- [72] B. McFee, O. Nieto, and J. P. Bello. Hierarchical evaluation of segment boundary detection. In *ISMIR*, 2015.
- [73] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.
- [74] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [75] B. C. Moore. *An introduction to the psychology of hearing*. Brill, 2012.
- [76] D. O’shaughnessy. *Speech communication: human and machine*. Universities press, 1987.
- [77] S. Pascual, A. Bonafonte, and J. Serra. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.
- [78] V. Phongthongloa, S. Kamonsantiroj, and L. Pipanmaekaporn. Learning high-level features for chord recognition using autoencoder. In *First International Workshop on Pattern Recognition*, pages 1001117–1001117. International Society for Optics and Photonics, 2016.
- [79] J. Pons and X. Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.
- [80] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra. Timbre analysis of music audio signals with convolutional neural networks, 2017.
- [81] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [82] F. Rigaud and M. Radenen. Singing voice melody transcription using deep neural networks. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*. New York, pages 737–743, 2016.
- [83] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [84] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [85] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. Learning the speech front-end with raw waveform cldnns. In *Proc. Interspeech*, 2015.
- [86] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- [87] J. Schlüter. Learning binary codes for efficient large-scale music similarity search. In *ISMIR*, pages 581–586, 2013.
- [88] J. Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 44–50, 2016.
- [89] J. Schlüter and S. Böck. Musical onset detection with convolutional neural networks. In *6th International Workshop on Machine Learning and Music (MML)*, Prague, Czech Republic, 2013.
- [90] J. Schluter and S. Bock. Improved musical onset detection with convolutional neural networks. In *Acoustics, Speech and Signal Processing, IEEE International Conference on*. IEEE, 2014.
- [91] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. *ISMIR*, 2015.
- [92] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [93] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- [94] B. J. Shannon and K. K. Paliwal. A comparative study of filter bank spacing for speech recognition. In *Microelectronic engineering research conference*, volume 41, 2003.
- [95] S. Sigtia, E. Benetos, N. Boulanger-Lewandowski, T. Weyde, A. S. d. Garcez, and S. Dixon. A hybrid recurrent neural network for music transcription. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2061–2065. IEEE, 2015.
- [96] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic music transcription. *arXiv preprint arXiv:1508.01774*, 2015.
- [97] S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon. Audio chord recognition with a hybrid recurrent neural network. In *ISMIR*, pages 127–133, 2015.
- [98] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6959–6963. IEEE, 2014.
- [99] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [100] A. J. Simpson, G. Roma, and M. D. Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. *arXiv preprint arXiv:1504.04658*, 2015.
- [101] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.
- [102] N. Steenbergen, T. Gevers, and J. Burgoyne. Chord recognition with stacked denoising autoencoders. *University of Amsterdam, Amsterdam, Netherlands*



(July 2014), 2014.

- [103] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [104] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [105] H. Tachibana, T. Ono, N. Ono, and S. Sagayama. Melody line estimation in homophonic music audio signals based on temporal-variability of melodic source. In *Acoustics speech and signal processing (icassp), 2010 IEEE international conference on*, pages 425–428. IEEE, 2010.
- [106] G. Tzanetakis and P. Cook. Gtzan genre collection. *web resource*, 2001.
- [107] S. Uhlich, F. Giron, and Y. Mitsufuji. Deep neural network based instrument extraction from music. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2135–2139. IEEE, 2015.
- [108] K. Ullrich, J. Schlüter, and T. Grill. Boundary detection in music structure analysis using convolutional neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014), Taipei, Taiwan, 2014*.
- [109] K. Ullrich and E. van der Wel. Music transcription with convolutional sequence-to-sequence models. International Society for Music Information Retrieval (ISMIR), 2017.
- [110] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.
- [111] A. Van Den Oord, S. Dieleman, and B. Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.
- [112] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.
- [113] B. Van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio. Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*, 2015.
- [114] G. H. Wakefield. Mathematical representation of joint time-chroma distributions. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 637–645. International Society for Optics and Photonics, 1999.
- [115] S.-Y. Wang, J.-C. Wang, Y.-H. Yang, and H.-M. Wang. Towards time-varying music auto-tagging based on cal500 expansion. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.
- [116] J. Wülfing and M. Riedmiller. Unsupervised learning of local features for music classification. In *ISMIR*, pages 139–144, 2012.
- [117] J. Yamagishi and S. King. Simple methods for improving speaker-similarity of hmm-based speech synthesis. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4610–4613. IEEE, 2010.
- [118] L.-C. Yang, S.-Y. Chou, J.-Y. Liu, Y.-H. Yang, and Y.-A. Chen. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. *arXiv preprint arXiv:1704.01280*, 2017.
- [119] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions. *arXiv preprint arXiv:1703.10847*, 2017.
- [120] L. Yann. *Modèles connexionnistes de l'apprentissage*. PhD thesis, These de Doctorat, Universite Paris 6, 1987.
- [121] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [122] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013.
- [123] X. Zhou and A. Lerch. Chord detection using deep learning. In *Proceedings of the 16th ISMIR Conference*, volume 53, 2015.

## APPENDIX

### A. FURTHER READING

In this section, we provide some introductory materials for several selected topics.

#### A.1 On the general deep learning

We recommend three materials on general topics of deep learning. LeCun et al. and Schmidhuber wrote overview articles of deep learning [55], [92]. The Deep learning book provides a comprehensive summary of many topics of deep learning [34]<sup>7</sup>.

More recent deep learning researches appear in the Journal of Machine Learning Research (JMLR)<sup>8</sup> and conferences such as the conference on Neural Information Processing Systems (NIPS)<sup>9</sup>, International Conference on Machine Learning (ICML)<sup>10</sup>, and International Conference on Learning Representations (ICLR)<sup>11</sup>. Many novel deep learning researches have been also introduced on arXiv, especially under the categories of computer vision and pattern recognition<sup>12</sup> and artificial intelligence<sup>13</sup>.

#### A.2 Generative adversarial networks

Generative adversarial networks (GANs) are a special type of neural networks that consists of a generator and a discriminator [35]. They are trained for two contradictory goals – the generator is trained to generate fake examples that are similar to the true data to fool the discriminator while the discriminator is trained to distinguish the fake examples from the true ones. As a result, the generator becomes capable of generating realistic examples [81].

In music research, [119] proposed a GANs to generate music on a symbolic domain and [11] used GANs to generate spectrograms of music instruments using visual image inputs. In the signal processing domain, speech denoising was performed using GANs in [77].

The two most influential papers are the first GAN proposal [35] and the first convolutional GANs [81]. Additionally, conditional GANs were introduced to steer the generated output [74]. Recently, many researches have discussed the training of GANs, e.g., [1] proposed better measure of the GAN training loss and [6] introduced a way to balance the training of the generator and the discriminator.

#### A.3 WaveNet models with the raw-audio

Sample-based audio generation is becoming feasible thanks to modern hardware development, aiming a high-quality audio. WaveNet proposed a fully convolutional and residual network for speech and music generation [112] and was used for musical note synthesiser [29] while SampleRNN achieved a competitive audio quality using recurrent neural networks [73].

#### A.4 Auto-encoders

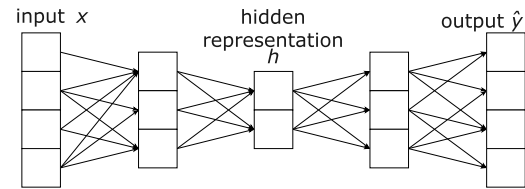


Figure 14: A block diagram of a deep autoencoder where 4D input  $x$  is compressed into a 2D vector ( $h$ ), which is decoded to the prediction  $\hat{y}$ .

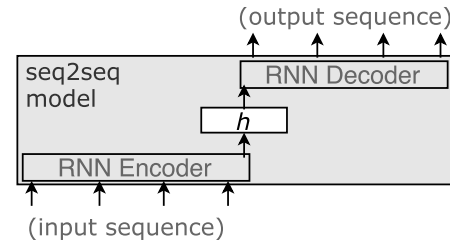


Figure 15: A block diagram of a sequence-to-sequence model that uses two RNNs.

An autoencoder is a type of neural network that learns to set the predicted values to be equal to the inputs [120] as illustrated in Figure 14. An autoencoder often has a bottleneck structure; that *encodes* the input data  $x$  into a vector ( $h$  in Figure 14) which is decoded to itself again. As a result, the network learns how to compress the input data into another vector  $h$  which can be used as a feature of the input.

There are clear pros and cons in autoencoder. It is trained in an unsupervised fashion, therefore a labelled dataset is not required. However, the learned representation by an autoencoder is not task-specific; it only learns to compress the data, which may or may not be relevant to the given task.

Autoencoder-based features are also used in music similarity search [87], music genre classification [23] and chord recognition [78, 102].

#### A.5 Sequence-to-sequence models

A sequence-to-sequence model learns how to encode a sequence into a representation using a DNN, which is decoded by another DNN [12, 103]. Usually, RNN is used for both the encoder and the decoder. It can provide a novel way to deal with audio signals, feature sequences, and symbolic sequences such as notes or chords. In MIR, it was used for transcription [109] and chord recognition [70]. [21] proposed a sequence-to-sequence autoencoder model for unsupervised audio feature learning.

<sup>7</sup>The book is also online for free (<http://www.deeplearningbook.org>)

<sup>8</sup><http://www.jmlr.org>

<sup>9</sup><https://nips.cc>

<sup>10</sup><http://icml.cc>

<sup>11</sup><http://www.iclr.cc>

<sup>12</sup><https://arxiv.org/list/cs.CV/recent>

<sup>13</sup><https://arxiv.org/list/cs.AI/recent>