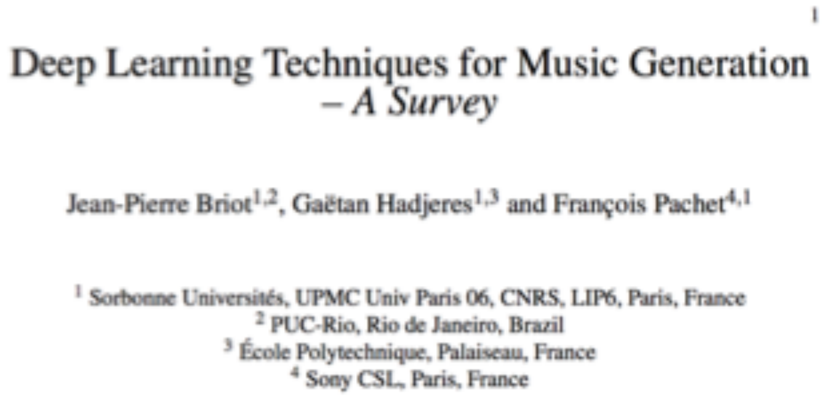
Deep Learning for Music Generation II

William Steimel

Table of Contents

Source



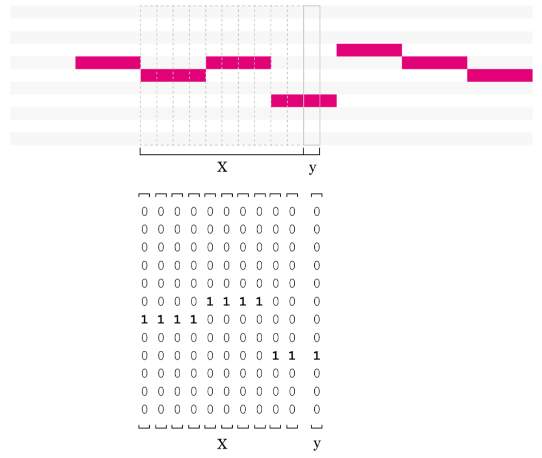
1

Deep Learning Techniques for Music Generation
– *A Survey*

Jean-Pierre Briot[1,2], Gaëtan Hadjeres[1,3] and François Pachet[4,1]

[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France
[2] PUC-Rio, Rio de Janeiro, Brazil
[3] École Polytechnique, Palaiseau, France
[4] Sony CSL, Paris, France

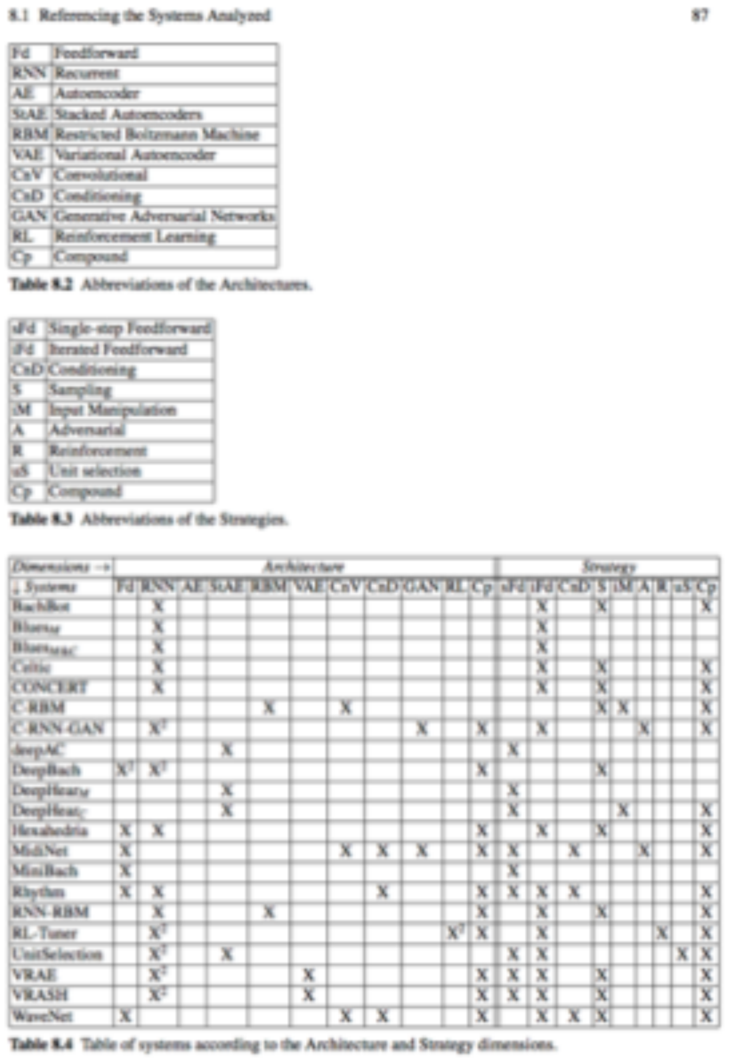
Chapter 5- Architecture



Motivation

- This chapter reviews all of the recent neural network architecture used to generate music
- Although this paper does not cover each algorithm in detail it gives me an idea of which neural networks I will focus on for future research
  - RNN/LSTM
  - Generative Adversarial Network (GAN)
  - Variational Autoencoder (VAE)
  - Restricted Boltzmann Machine (RBM)

Last Presentation



8.1 Referencing the Systems Analyzed 87

Fd	Feedforward
RNN	Recurrent
AE	Autoencoder
StAE	Stacked Autoencoders
RBM	Restricted Boltzmann Machine
VAE	Variational Autoencoder
CnV	Convolutional
CnD	Conditioning
GAN	Generative Adversarial Networks
RL	Reinforcement Learning
Cp	Compound

Table 8.2 Abbreviations of the Architectures.

sFd	Single-step Feedforward
iFd	Iterated Feedforward
CnD	Conditioning
S	Sampling
iM	Input Manipulation
A	Adversarial
R	Reinforcement
uS	Unit selection
Cp	Compound

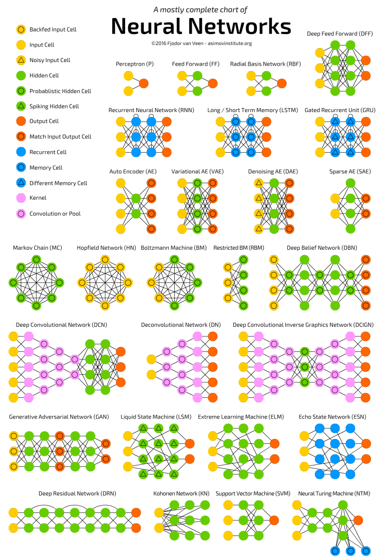
Table 8.3 Abbreviations of the Strategies.

Dimensions →	*Architecture*											*Strategy*								
↓ *Systems*	Fd	RNN	AE	StAE	RBM	VAE	CnV	CnD	GAN	RL	Cp	sFd	iFd	CnD	S	iM	A	R	uS	Cp
BachBot		X											X		X					X
Blues$_M$		X											X							
Blues$_{MMC}$		X											X							
Celtic		X											X		X					X
CONCERT		X											X		X					X
C-RBM					X		X								X	X				X
C-RNN-GAN		X^7							X		X		X				X			X
deepAC				X								X								
DeepBach	X^7	X^7									X				X					
DeepHear$_M$				X								X								
DeepHear$_C$				X								X				X				X
Hexahedria	X	X									X		X		X					X
MidiNet	X						X	X	X		X	X		X			X			X
MiniBach	X											X								
Rhythm	X	X						X			X	X	X	X						X
RNN-RBM		X			X						X		X		X					X
RL-Tuner		X^7								X^7	X		X					X		X
UnitSelection		X^7		X								X	X						X	X
VRAE		X^2				X					X	X	X		X					X
VRASH		X^2				X					X	X	X		X					X
WaveNet	X						X	X			X		X	X	X					X

Table 8.4 Table of systems according to the Architecture and Strategy dimensions.

What is Architecture?

- Architecture can be defined as in the non-technical sense – “the complex or carefully designed structure of something.” – Dictionary.com
- Architectures are the various ways networks or topologies can be structured to address/represent different problems in Deep Learning.
- Today’s presentation will cover the types of Deep Learning Architectures used to generate music


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Source: Asimov Institute

Introduction - Linear Regression

- Linear Regression is an essential foundation of neural networks
- Objective - Fitting a line to model a relationship between a scalar variable y and one or more explanatory variables $x_1, \dots x_n$
  - h is the model (also named hypothesis, as the hypothetical best model to be discovered, i.e. learnt);
  - b the bias (representing the offset, also sometimes noted θ_0);

Linear Regression Formula

$$h = b + \theta_1 x_1 + \dots + \theta_n x_n = b + \sum_{i=1}^{n} \theta_i x_i \qquad (5.1)$$

Training

- What is Training ?
  - Purpose - Learn Optimal Parameters for weights θ_n and bias b that fit the best with the actual data
- Training Algorithm Steps
  - Step 1. Initialize each parameter (weight/bias) to a random or heuristic value
  - Step 2. Compute the value of the model h for all examples
  - Step 3. Compute the cost (loss) – distance between prediction and actual values for all examples
    - EX: could be measured by MSE
  - Step 4. Compute the gradients/partial derivatives to each weight of the cost function
  - Step 5. Search for new value for each parameter (weight and bias) guided by the value of each gradient
  - Step 6. Iterate until each error reaches minimum
- We can then generate predictions based on the best fitting model (Example on the right with one explanatory x)

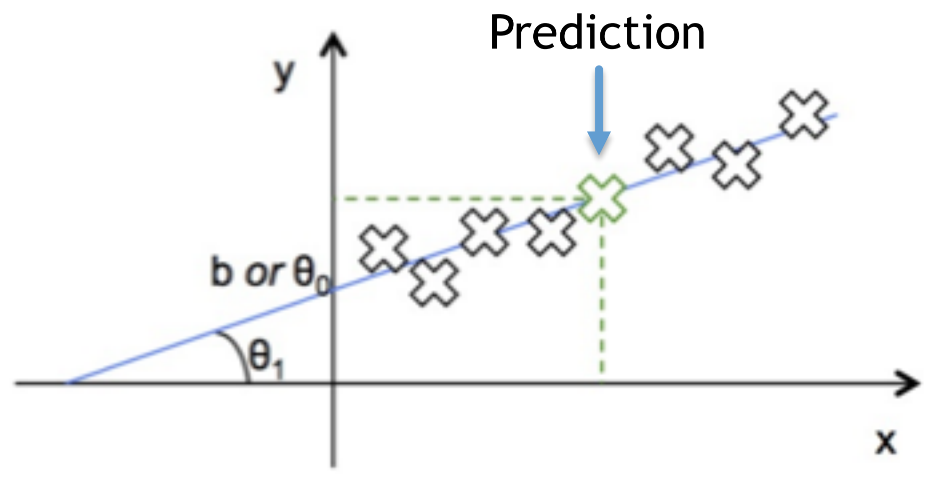
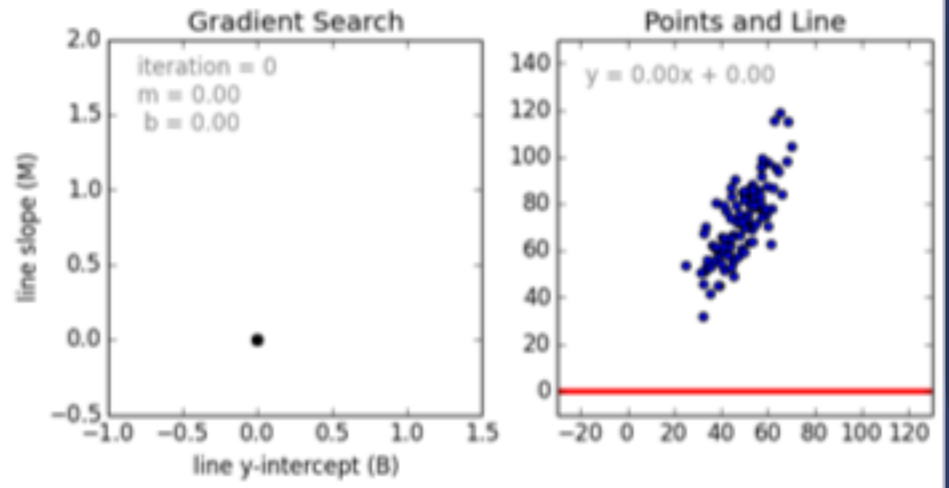


Fig. 5.1 Example of simple linear regression.



Linear Regression Architecture

- As a precursor to Neural Networks the author then introduces an architectural view of Linear Regression
- The weighted sum is represented as a Computational Unit
- Inputs come from the x nodes and bias (circles)
  - In this example – There are 4 explanatory variables and a bias
  - $x_1, x_2, x_3, \ x_4, \ b$
- Linear Regression can also be extended to a multivariate level (Need to predict multiple y values)

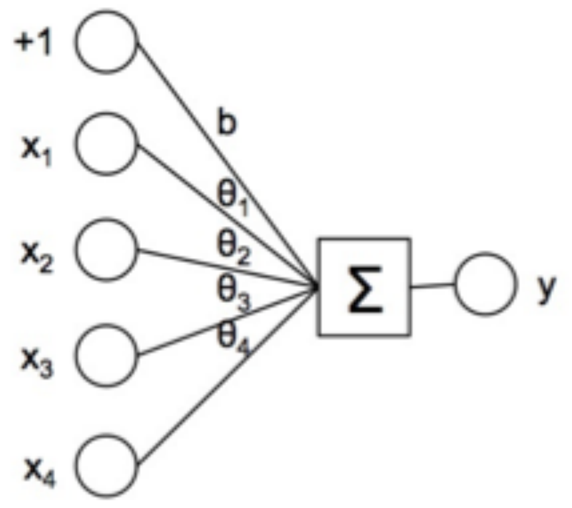


Fig. 5.2 Architectural model of linear regression.

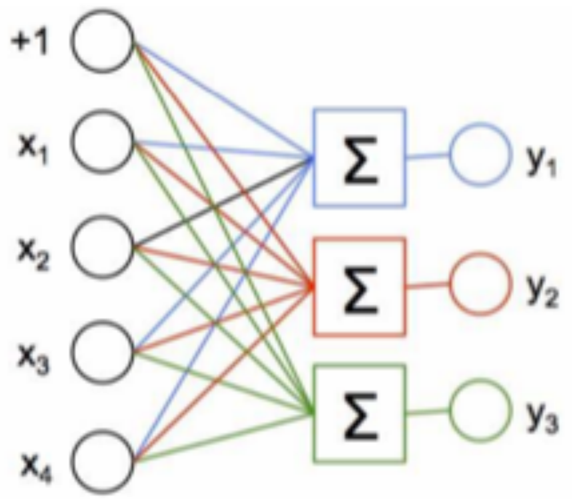
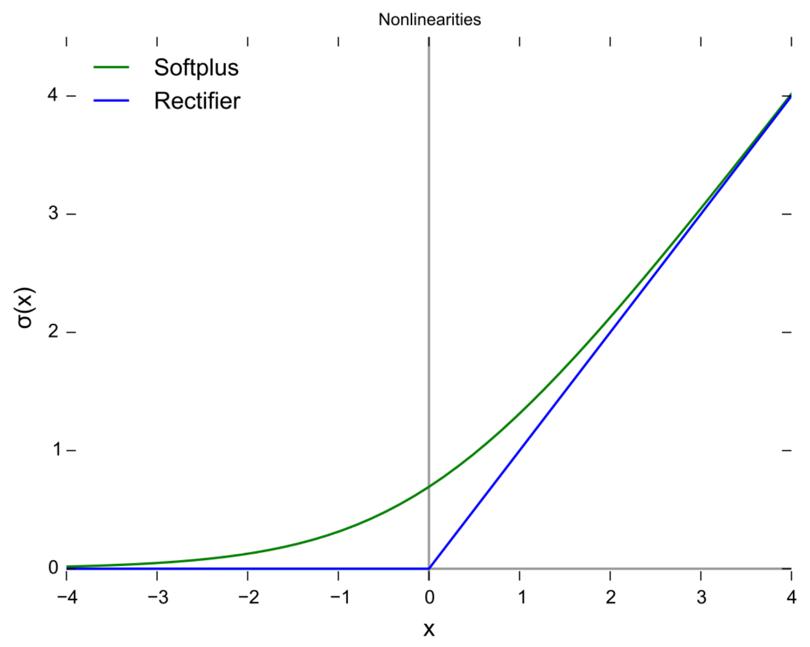


Fig. 5.3 Architectural model of multivariate linear regression.

Activation Functions

- We can then apply an Activation Function (AF) to each weighted sum unit (Computational Unit)
- The activation function allows us to introduce non-linearity in order for the model to be more general.
  - The most common activation function is called RELU which stands for Rectified Linear Unit which is simply a negative values filter
    - $ReLU(x) = \max(0, x)$
  - A smoother version is called Softplus (green line)
- This is a basic building block of neural networks
  - The right shows an example of an AF being applied to all Computational Units



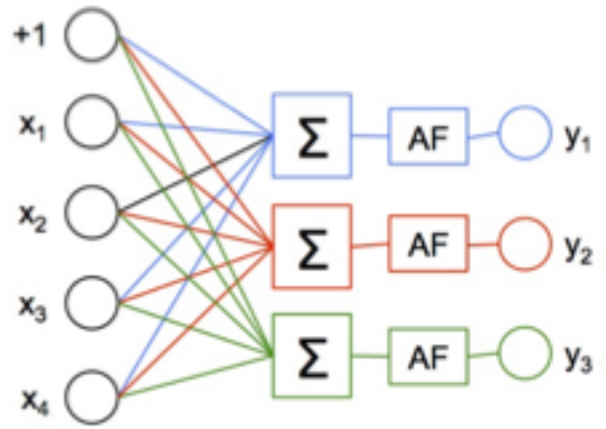


Fig. 5.5 Architectural model of the building block (multivariate linear regression with activation function).

How do we compute the output?

- We just need to ”feedforward” the network –
  - We provide input data (feed in) to the network and compute the output values
- Computation turns out to be simple vector (or Matrix) to matrix multiplication which many computational linear algebra libraries can handle

$$h(X) = AF(b + WX^{T}) \quad (5.2)$$

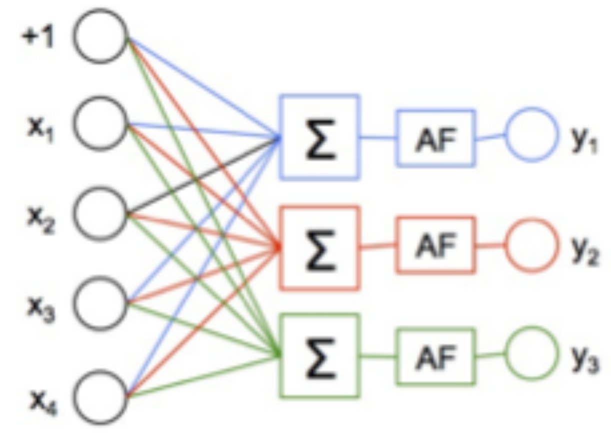


Fig. 5.5 Architectural model of the building block (multivariate linear regression with activation function).

Basic Architectures

- Now that we understand the basic building block of Neural networks this research paper now covers the four basic common neural network architectures used for music generation
  - Neural Networks
  - Auto-Encoders
  - Restricted Boltzmann machines (RBMs)
  - Recurrent Neural Networks (RNN’s)

Multilayer Neural Network aka Feedforward Neural Network

- A multilayer Neural network is essentially a combination of successive layers of the basic building blocks/computational units mentioned in previous slides (Stacking functions)
  - The First Layer- Input Nodes (Input Layer)
  - The Last Layer – Output Nodes (Output Layer)
  - Any Layer in between the First/Last is referred to as a hidden layer
- "The combination of hidden layer and non linear activation function make the neural network an universal approximator, able to overcome the non linear separability limitation"

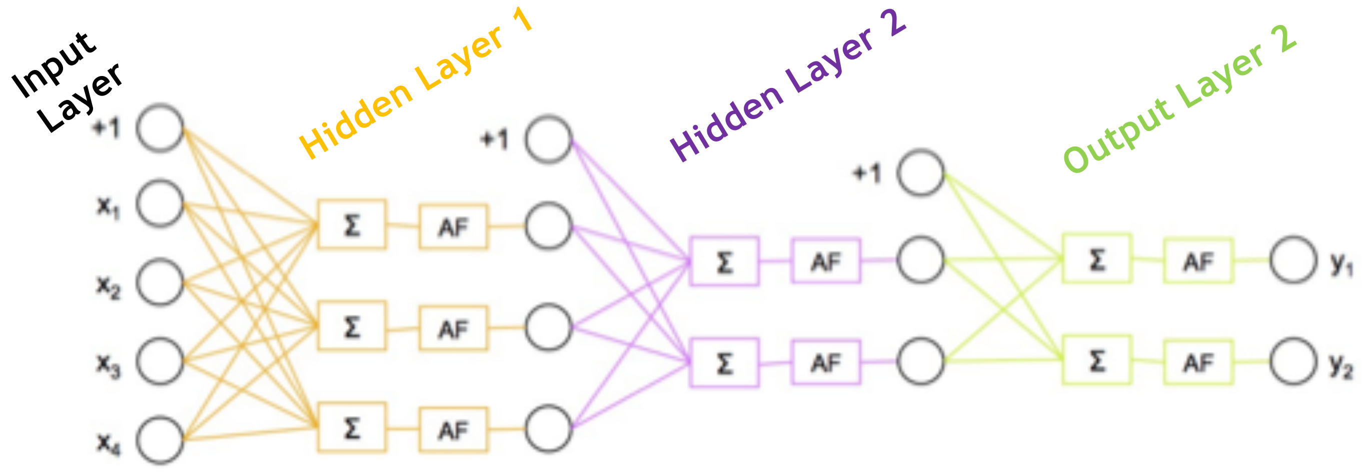


Fig. 5.6 Multilayer neural network.

Multilayer Neural Network aka Feedforward Neural Network

- The first Neural Networks didn’t have many hidden layers but in deep networks the amount of layers can be very large.
  - GoogLeNet – 27 layers
  - ResNet – 152 layers

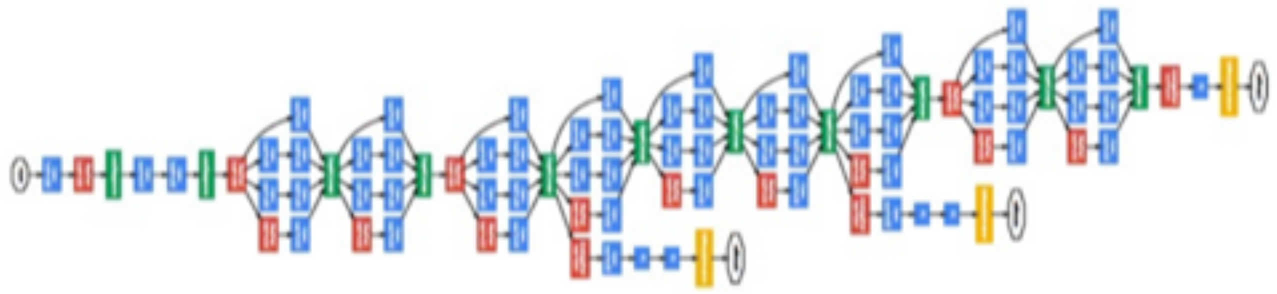


Fig. 5.7 GoogLeNet deep network architecture.

Multilayer Neural Network aka Feedforward Neural Network

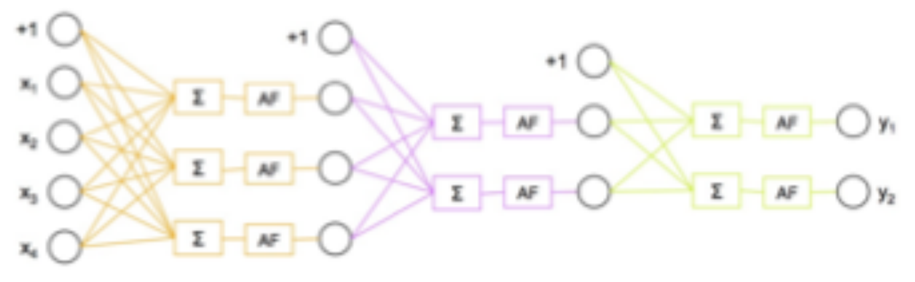


Fig. 5.6 Multilayer neural network.

- We can also further simplify the previous neural network in two ways:
  - 1. Remove the sum and non linear functions
  - 2. Hide the number of nodes

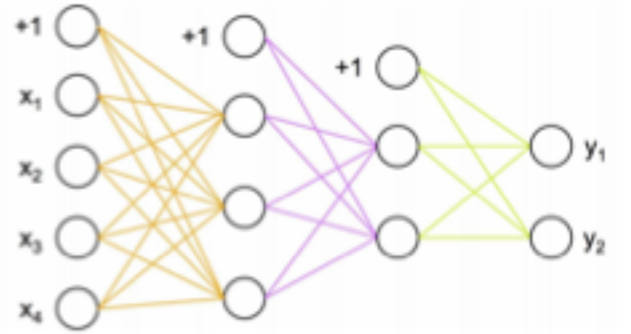


Fig. 5.8 Multilayer neural network – Simplified representation.

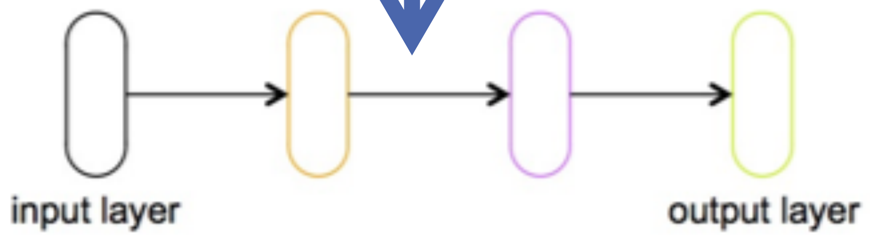


Fig. 5.9 Multilayer neural network – Abstract representation.

Output Activation Functions

- To generate an output an Activation Function is also used:
- Three possible types:
  - **Identity** - For a prediction task where output values are continuous (regression tasks)
  - **Sigmoid** - Used for binary classification (probability)
  - **Softmax-** The most common for a classification task more than two classes (probability)
    - Softmax usually represents a probability distribution of occurrence of each output and ensures the sum of probabilities equal to 1

Sigmoid Function

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{5.3}$$

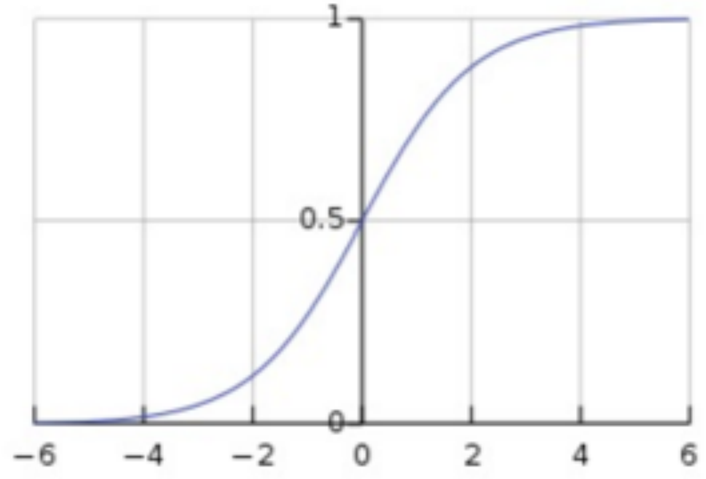


Fig. 5.10 Sigmoid function.

Softmax Function

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{j=1}^{p} e^{z_j}} \tag{5.4}$$

$$\sigma \begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix} \tag{5.5}$$

Cost Function

- Typical Cost functions used in Deep Learning include:
  - Quadratic cost (Mean Squared Error or Maximum Likelihood)
  - Cross-entropy cost
  - Kullback-Leibler divergence (KL-divergence)

- Cost function chosen highly depends on the choice of activation function

Feedforward Propagation

- Feedforward propagation is the method for pushing the input data and propagating the computation through all the layers until producing an output.
- It consists of successive matrix products transformed by the activation function

$$output^{(k)} = AF(b^{(k)} + W^{(k)} output^{(k-1)}) \qquad (5.6)$$

- Neural Networks are considered deterministic – This means that the same input will always produce the same output
  - This is not good in the case of music generation –
  - One method for solving this is to use sampling methods to introduce variability in the generation process
    - Can sample the probability distribution represented by the softmax function instead of just selecting the value with the highest probability

Training a Neural Network

- Backpropagation is the method used to estimate the derivatives (for a multi-layer neural network)
- As the cost function of a multilayer neural network is not convex (many local minima) Gradient Descent does not guarantee reaching global minimum
  - Stochastic Gradient Descent is often employed

- I recommend this video if you are interested in backpropagations details
  - https://www.youtube.com/watch?v=Ilg3gGewQ5U

Recurrent Neural Networks

- “A Recurrent Neural Network (RNN) is a (feedforward) neural network extended to include recurrent connections.”
  - It can learn series (temporal series in the musical context)
  - Recurrent Neural Networks are good at learning sequences and often used for text generation and music generation

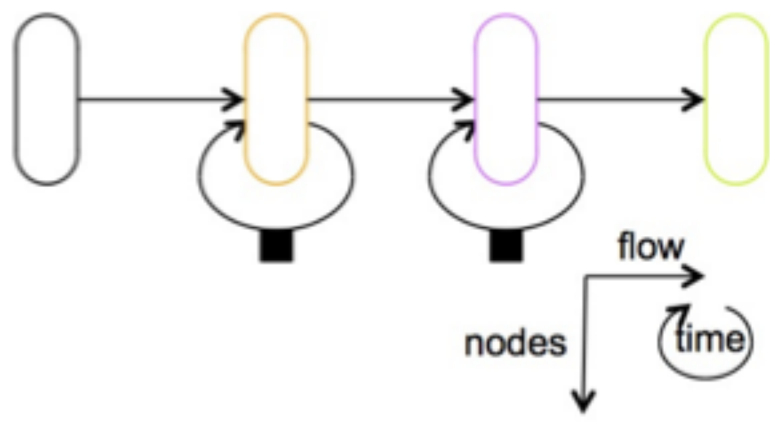


Fig. 5.11 Recurrent neural network (Folded).

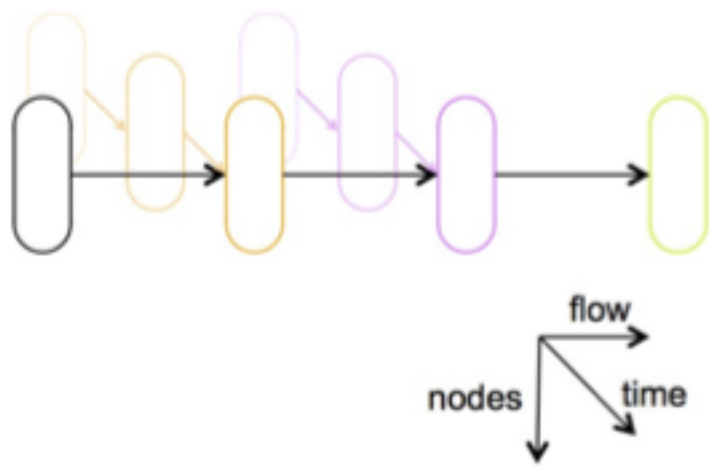


Fig. 5.12 Recurrent neural network (Unfolded).

Recurrent Neural Networks

- Recurrent Neural Networks differ from feed-forward as instead of considering an input x and output y they aim to model a sequence of inputs x_t and outputs y_t indexed by parameter t which represents time
- "The basic idea is that the outputs of a hidden layer (h_t) reenter into itself (with a specific weight matrix, that we note here W_r) as an additional input to compute next values ($h_t + 1$) of the hidden layer. This way, the network can learn, not only based on current data, but also on previous one. "

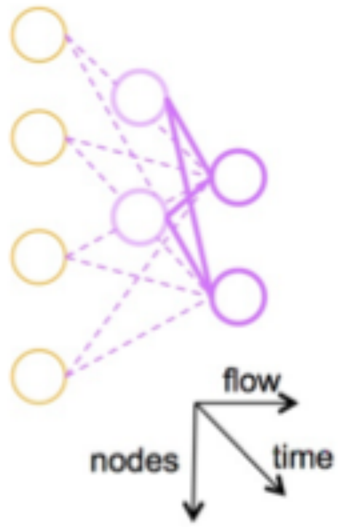
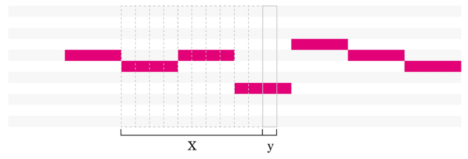


Fig. 5.13 Unfolded Recurrent layer.

Recurrent Neural Network Training

- Recurrent Neural Networks are not trained in the same way as feedforward networks
- A RNN learns the probability distribution over a sequence by being trained to predict the next symbol in the sequence $P(x_t|\, x_t - 1, \ldots, x_1)$
- Present an example element of a sequence as the input x_t (Note for learning melodies) and the actual next element of the sequence as output (x_{t+1})
- This trains the neural network to predict the next element of the sequence



$$\underbrace{\begin{matrix} 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0 \\ 1&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \end{matrix}}_{X} \quad \underbrace{\begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{matrix}}_{y}$$

LSTM

- LSTM is now considered the standard for recurrent networks
- Recurrent networks suffered from a problem called the vanishing or exploding gradient problem – effects were either minimized or amplified
  - Training problem related to the difficulty in estimating gradients when performing Backpropagation through time (BPTT)
- LSTM secures information in memory cells within a block protected from the standard flow of Recurrent networks
  - “Each LSTM block learns how to maintain its memory as a function of input in order to minimize loss”

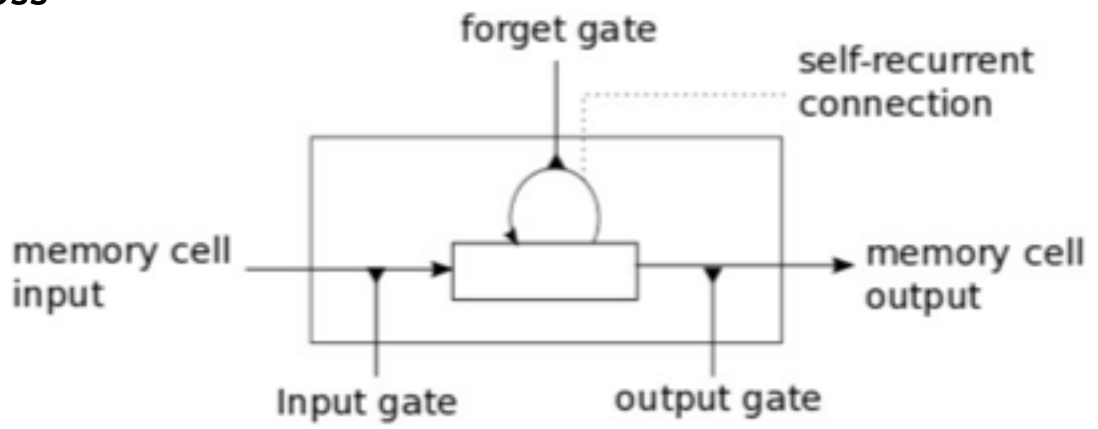


Fig. 5.14 LSTM architecture.

Autoencoder

- A neural network with one hidden layer with an additional constraint:
  - Constraint - Number of output nodes is equal to input nodes
- Autoencoders learn through general supervised learning methods and generally try to learn the identity function
- Hidden layer generally has fewer nodes than the input/output layer
  - Encoder – Compresses Information
  - Decoder – Reconstructs initial information as well as possible
  - Example: Compress the value 2 then allow the decoder to re-represent 2 as best as possible
- This makes it possible for the autoencoder to extract higher level features

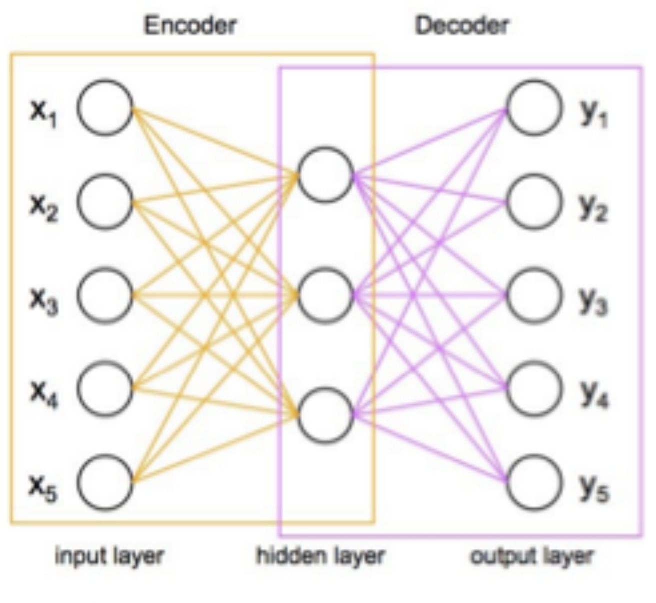
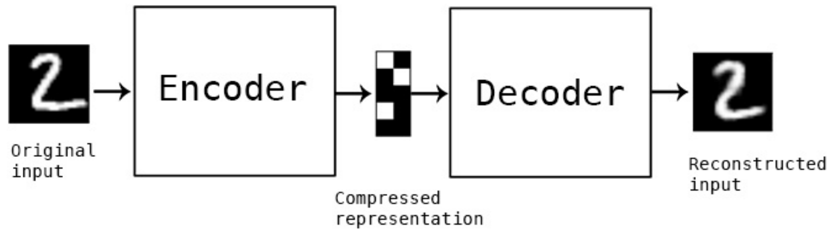


Fig. 5.15 Autoencoder.



Source: Keras

Stacked Autoencoders

- Hierarchical stacking of autoencoders where each successive hidden layer has less units/nodes
- Increases Compression and helps extract higher-level features
  - Often used for feature extraction
- Can also be used for Music Generation (DeepHear System)

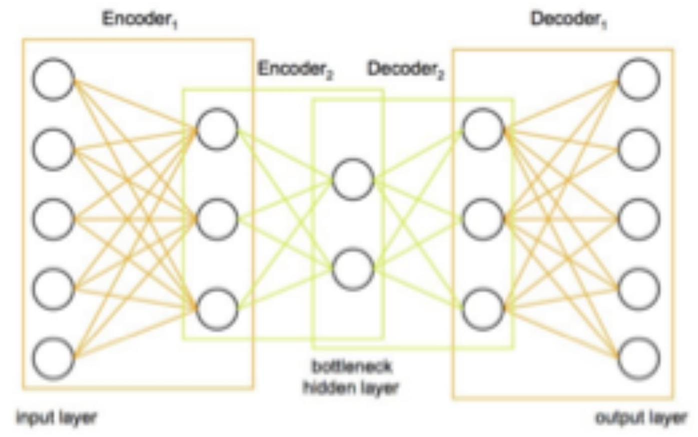


Fig. 5.16 Stacked autoencoders architecture.

Restricted Boltzmann Machine (RBM)

- RBM - “A generative stochastic artificial neural network that can learn a probability distribution over its set of inputs”
  - Named after the Boltzmann distribution in statistical mechanics
- Architectural Restriction for RBM – 2 layers
  - Visible Layer (Serves as input and output layer)
  - Hidden Layer
- Connections cannot exist between nodes in the same layer
- RBM are similar to autoencoders but two key differences
  - RBM has no output (input also acts as output)
  - RBM is stochastic
  - Values manipulated are Booleans

Ludwig Boltzmann

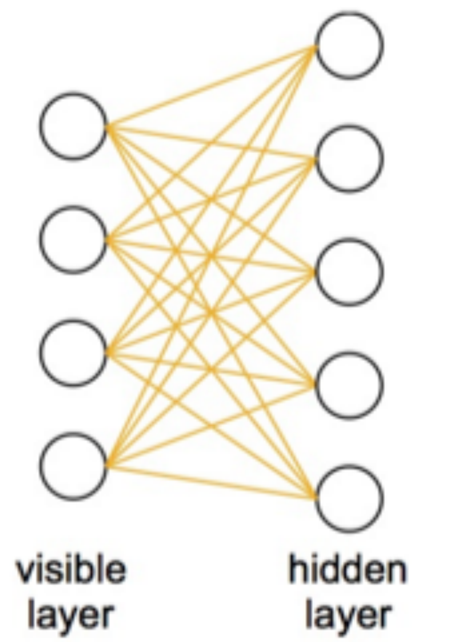


Fig. 5.17 Restricted Boltzmann Machine architecture.

Restricted Boltzmann Machine (RBM)

- RBM is a neural network architecture for learning distributions
  - It can learn efficiently with few examples
- Training an RBM is similar to an auto-encoder (2 repeating steps)
  - Feedforward step – encode – by making predictions about hidden layer node activations
  - Backward step - decode/reconstruct – by making predictions about the visible layer node activations
- This process differs from autoencoders and is called “Generative Learning”

Ludwig Boltzmann

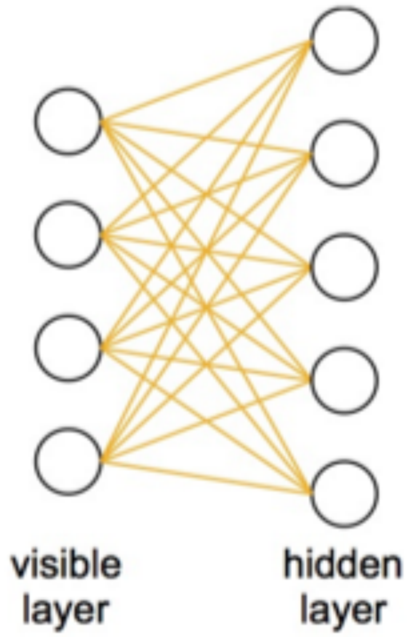
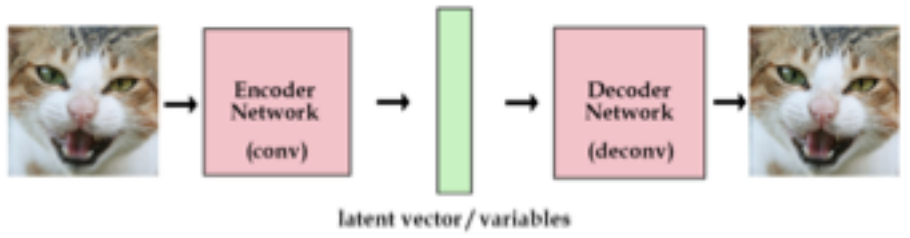


Fig. 5.17 Restricted Boltzmann Machine architecture.

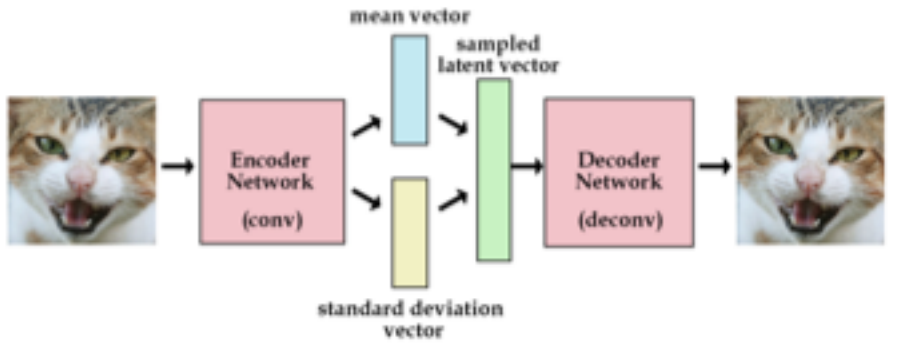
Variational Autoencoder

- If you remember the previous autoencoder – a general autoencoder just generated a lower quality copy of the input image with the higher level features extracted.
  - This is not as desired for music generation
- Variational Autoencoders are a recent approach to deep learning music generation (Generative Model)
- Variational Encoders are complex but to explain the intuition simply.
  - Variational Encoders function similarly to autoencoders but add a constraint that forces it to generate latent vectors that follows a gaussian distribution
  - Noise of output is controlled by a σ hyperparameter which controls the standard deviation of the gaussian distribution
    - A small value makes the learnt examples more similar to the actual examples
  - The decoder learns the relationship between the gaussian distribution of latent variables and the learnt examples
- “Sampling from the VAE is immediate, one just needs to sample a value following a Gaussian distribution for the hidden layer (latent variables), input it into the decoder and feedforward the decoder to generate an output corresponding to the distribution of the examples”
- In this way you can teach it how to draw MNIST numbers that are similar to the MNIST numbers but not exactly the same (in the case of music generate music that is unique)

Autoencoder



Variational Autoencoder



* Diagram Source: http://kvfrans.com/variational-autoencoders-explained/

VAE

Original MNIST

Convolutional Architectural Pattern

- Convolutional Architectural Patterns are common for image applications
  - Convolutions are less common music for applications
- The network learns from individual elements (images – pixels) from an area called a convolution.
- There however have been some recent architectures that have utilized convolutions for music generation

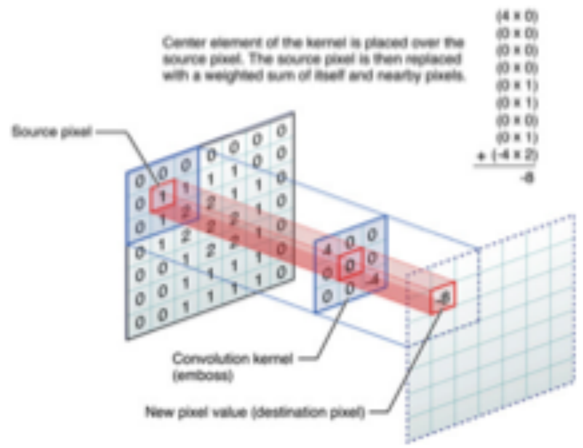

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Fig. 5.18 Convolution.

Conditioning Architectural Pattern

- "The idea of conditioning (sometimes also named conditional architecture) is to condition the architecture on some extra conditioning information, which could be arbitrary, e.g., a class label or data from other modalities."
- Examples:
  - Bass line or beat structure (Rhythm Generation System)
  - Chord progression (MidiNet)
  - Musical genre or an instrument (WaveNet)
- The conditioning information is fed into the architecture as an additional input layer
  - Helps control the generative process (We provide a musical genre and the model generates based on that input)

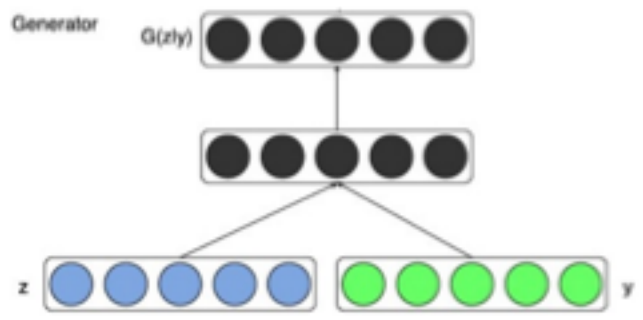


Fig. 5.19 Conditioning architecture

Generative Adversarial Networks (GAN)

- Proposed by Ian Goodfellow- 2014
- Trains two networks simultaneously which are competing **(Adversarial)**
  - Generative Model (G) – objective is to transform random noise vectors into faked samples, which resemble real samples drawn from a distribution of real images (Generates new data)
  - Discriminative Model (D) – Estimates the probability a training sample came from training data rather than G (Evaluates for authenticity)
  - Good Example from deeplearning4j-
    - GAN is like a combination of a counterfeiter of money and cop in a game of cat and mouse.
    - Counterfeiter **(Generator)** is trying to make fake money that the cop cannot detect – Goal is to minimize $V(G, D)$
    - Cop **(Discriminator)** is learning to detect them - Goal is to maximize $V(G, D)$
  - Each side is training and learning to compete with the other side.
  - GAN is used in the MIDINet System

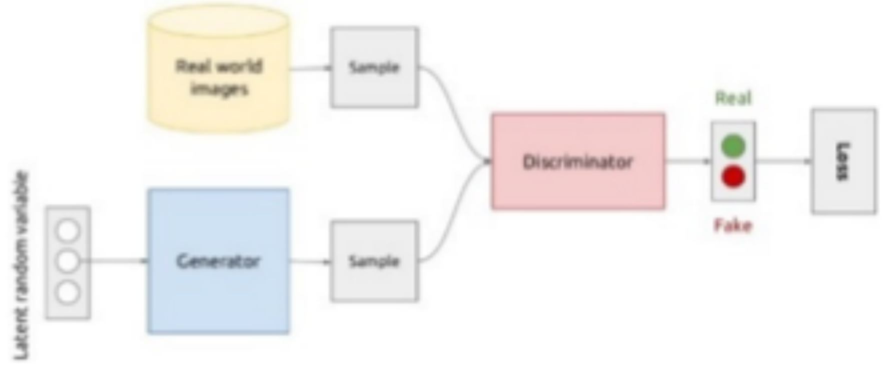
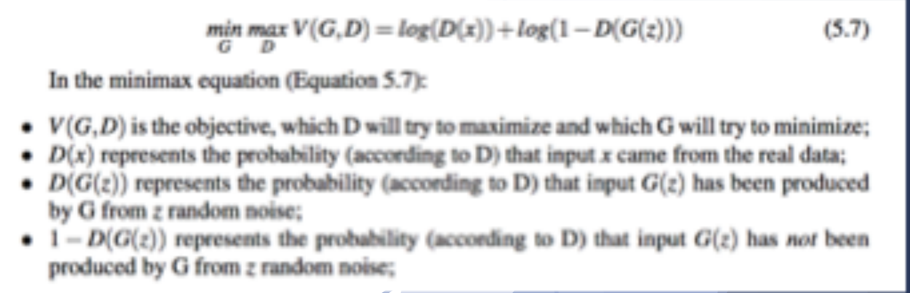


Fig. 5.20 Generative adversarial networks architecture



$$\min_G \max_D V(G,D) = log(D(x)) + log(1 - D(G(z))) \tag{5.7}$$

In the minimax equation (Equation 5.7):

- $V(G,D)$ is the objective, which D will try to maximize and which G will try to minimize;
- $D(x)$ represents the probability (according to D) that input x came from the real data;
- $D(G(z))$ represents the probability (according to D) that input $G(z)$ has been produced by G from z random noise;
- $1 - D(G(z))$ represents the probability (according to D) that input $G(z)$ has *not* been produced by G from z random noise;

* Example Source: https://deeplearning4j.org/generative-adversarial-network

Hyperparameters

- In addition to the parameters (weights/bias) of the model there are also hyperparameters (Two Types)
- **Structural Hyperparameters**
  - Number of layers
  - Number of nodes
  - Non linear activation function used
- **Control Hyperparameters (Concerned with learning process)**
  - Optimization Function
  - Learning Rate
  - Regularization Strategy
  - Regularization Parameters
- Good parameter settings are essential for creating accurate neural networks and in this case the best generating model for music
- There are many methods to hyperparameter optimization including grid search among others.

Compound Architectures

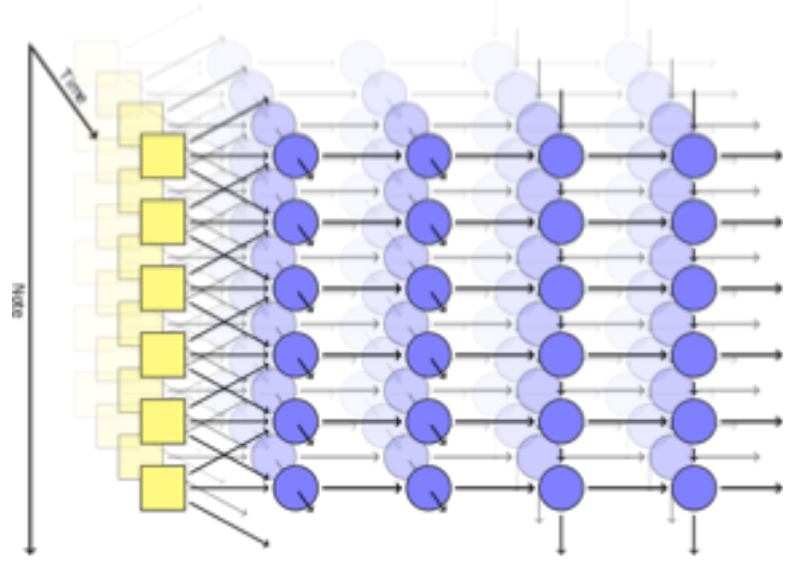
- Compound architectures combine multiple architectures together and there are many classifications that the authors have created based on their observation
  - Composition – Two or more architectures combined of the same type or different types
  - Nesting – One architecture is nested into another (Example cited was stacked autoencoder)
  - Integration – Two types of architecture are merged
  - Pattern instantiation – “An architectural pattern is instantiated onto some given architecture(s).”

Compound Architectures

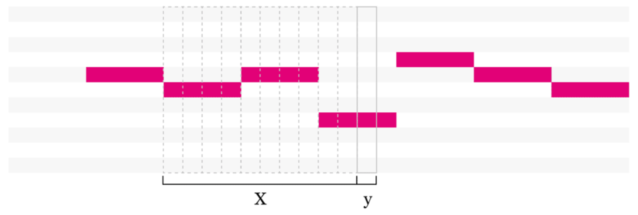
- **Convolutional Generative Adversarial Networks** – Combination of Convolutions with GAN
- **Recurrent Generative Adversarial Networks -** Combination of RNN and GAN
- **RNN-RBM** – Combination of LSTM and Restricted Boltzmann Machines

Next Time

Johnson Hexahedria- RNN Architecture

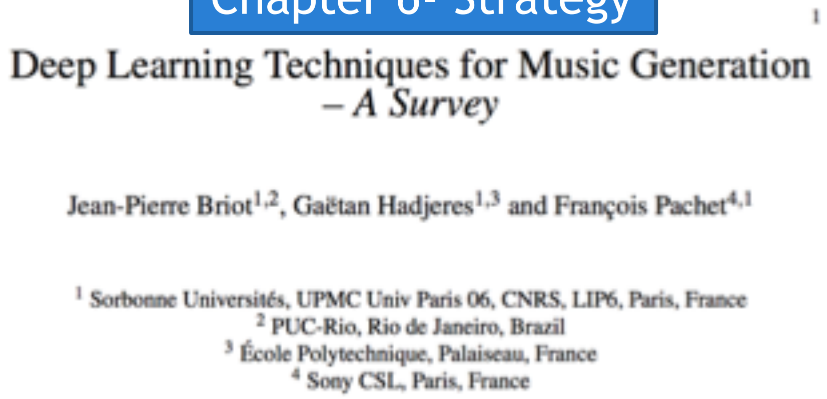


MIDI – RNN Algorithm



$$\begin{array}{cccccccccc|c}
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0\\
\hline
&&&&X&&&&&&y
\end{array}$$

Chapter 6- Strategy



1

Deep Learning Techniques for Music Generation
– *A Survey*

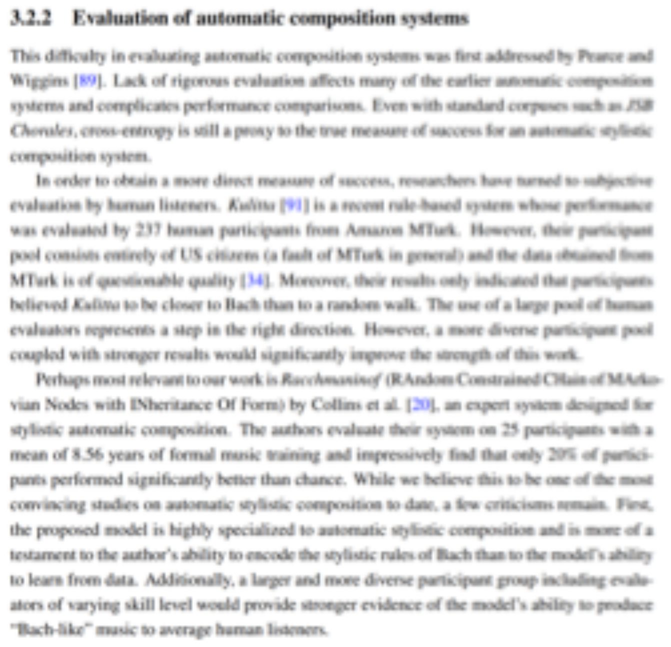
Jean-Pierre Briot[1,2], Gaëtan Hadjeres[1,3] and François Pachet[4,1]

[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France
[2] PUC-Rio, Rio de Janeiro, Brazil
[3] École Polytechnique, Palaiseau, France
[4] Sony CSL, Paris, France

Thesis

Evaluation Methods Compiled

Bachbot



3.2.2 Evaluation of automatic composition systems

This difficulty in evaluating automatic composition systems was first addressed by Pearce and Wiggins [89]. Lack of rigorous evaluation affects many of the earlier automatic composition systems and complicates performance comparisons. Even with standard corpuses such as *JSB Chorales*, cross-entropy is still a proxy to the true measure of success for an automatic stylistic composition system.

In order to obtain a more direct measure of success, researchers have turned to subjective evaluation by human listeners. *Kulitta* [91] is a recent rule-based system whose performance was evaluated by 237 human participants from Amazon MTurk. However, their participant pool consists entirely of US citizens (a fault of MTurk in general) and the data obtained from MTurk is of questionable quality [34]. Moreover, their results only indicated that participants believed *Kulitta* to be closer to Bach than to a random walk. The use of a large pool of human evaluators represents a step in the right direction. However, a more diverse participant pool coupled with stronger results would significantly improve the strength of this work.

Perhaps most relevant to our work is *Racchmaninof* (RAndom Constrained CHain of MArkovian Nodes with INheritance Of Form) by Collins et al. [20], an expert system designed for stylistic automatic composition. The authors evaluate their system on 25 participants with a mean of 8.56 years of formal music training and impressively find that only 20% of participants performed significantly better than chance. While we believe this to be one of the most convincing studies on automatic stylistic composition to date, a few criticisms remain. First, the proposed model is highly specialized to automatic stylistic composition and is more of a testament to the author's ability to encode the stylistic rules of Bach than to the model's ability to learn from data. Additionally, a larger and more diverse participant group including evaluators of varying skill level would provide stronger evidence of the model's ability to produce "Bach-like" music to average human listeners.

http://www.mlsalt.eng.cam.ac.uk/foswiki/pub/Main/ClassOf2016/Feynman_Liang_8224771_assignsubmission_file_LiangFeynmanThesis.pdf

Data Compiled

classtab.org
Classical Guitar Tablature

Collected 1355 Guitar Midi Files from Classtab.org – Database of Classical Guitar Music

- Currently evaluating quality of data

Python Music Analyis Demo

Next Step

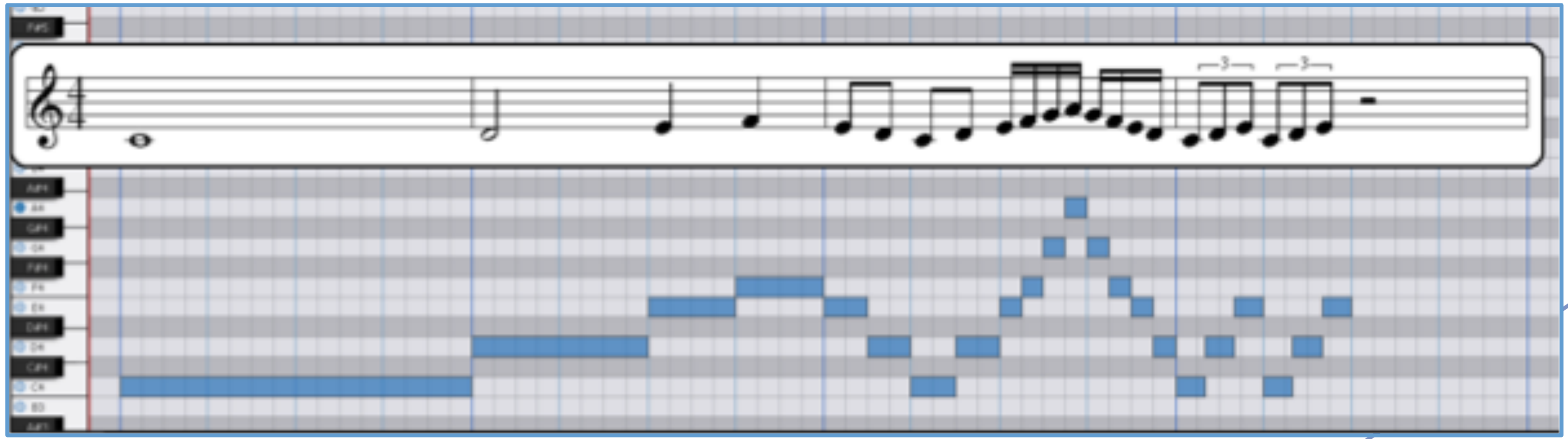
- Need to learn how to one-hot encode musical data

My Interest

- My interest largely lies in Symbolic Music representation
  - MIDI files
  - Piano Roll
- MIDI files are often fundamental data files for composers/film scorers nowadays

Piano Roll vs Traditional Notation

Note/Pitch



Time

Thesis Idea

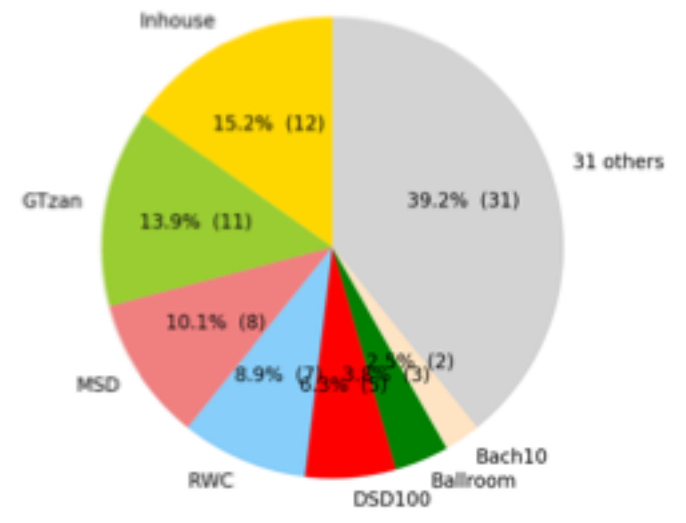
- **Method 1** – Create multiple models based on recently used music generation algorithms and use the same training set (Markov Chain, Restricted Boltzmann Machines, RNN/LSTM)
    - Evaluate Musical Output via survey as music is subjective
    - Determine which model is best in eyes of listeners
- **Method 2** – Create one model based on RNN/LSTM based on one composers data.
    - Evaluate Musical Output via survey as music is subjective
    - Determine whether this music is plausible/pleasant

Steps

- 1. Research History of Algorithmic Music Composition/Deep Learning
- 2. Gather Data Files (Midi Files/Symbolic Representations of Music)
- 3. Clean Data (Format Data and feature engineer so data is proper for NN)
- 4. Train the Model/Perform Experiments (Using NN's etc.)
- 5. Evaluate Model performance

Data Sources

- Touhou:
  - http://easypianoscore.jp/
- Final Fantasy:
  - https://www.thefinalfantasy.com/site/midi-collection.html
- Bach/Chopin
  - https://www.classicalarchives.com/midi.html



Most Common Datasets used in DL Music Research

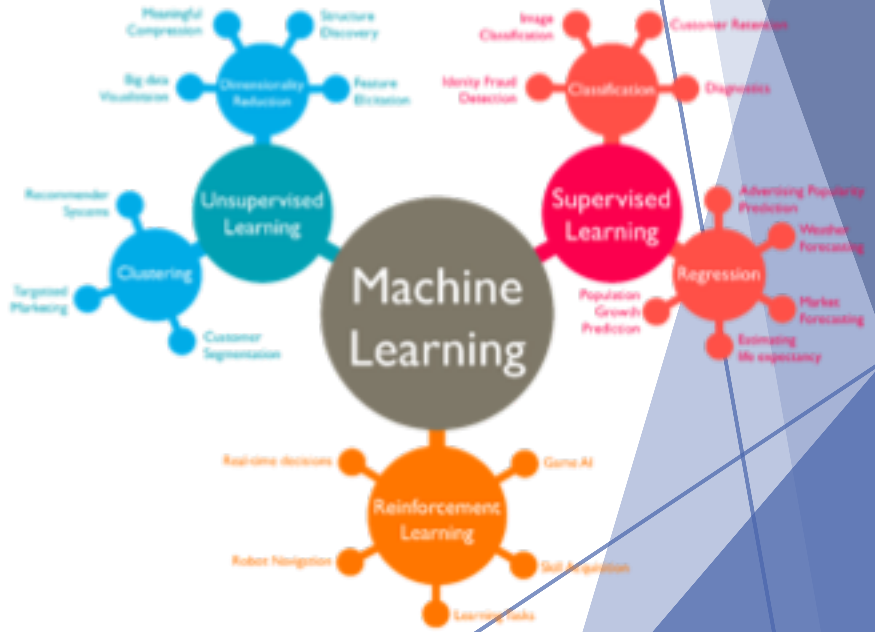
Literature Review

After Performing Research on Deep Learning for Music Generation, I found many resources that I will review in the next 2 years.

- https://medium.com/artists-and-machine-intelligence/neural-nets-for-generating-music-f46dffac21c0 - Neural Nets application to music
- http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/ - Composing with RNN
- http://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf - Composing with LSTM
- https://cs224d.stanford.edu/reports/allenh.pdf - Recent paper on Deep Learning for Music
- https://magenta.tensorflow.org/ - **Magenta** is a research project exploring the role of machine learning in the process of creating art and music.
- https://arxiv.org/pdf/1709.01620.pdf - Deep Learning for Music Generation Survey Paper
- https://github.com/ybayle/awesome-deep-learning-music - Compilation of Deep Learning related Music research

Research Plan

- **Year 1: (General Machine Learning)**
  - **11/9 - Machine Learning - Decision Tree's / Random Forest**
  - **12/7 - Machine Learning Basics / End to End ML / Regression**
  - **1/18 - Machine Learning - Classification (Logistic Regression, LDA, K-NN)**
  - **4/25 - Machine Learning - Recommender Systems**
  - Machine Learning – Naïve Bayes Classifier/ Bayes Theorem
  - Machine Learning – Support Vector Machines
  - Machine Learning - Natural Language Processing
  - Machine Learning - Deep Learning
  - Machine Learning – XGBoost (New)
  - Machine Learning – Feature Engineering (New)
- **Thesis - Deep Learning for Music Generation**
  - Music Composition (Deep Learning)



Github

- https://steimel64.github.io/