

Deep Learning Techniques for Music Generation (6)

Jean-Pierre Briot

`Jean-Pierre.Briot@lip6.fr`

Laboratoire d'Informatique de Paris 6 (LIP6)

Sorbonne Université – CNRS



Programa de Pós-Graduação em Informática (PPGI)

UNIRIO



Architectures

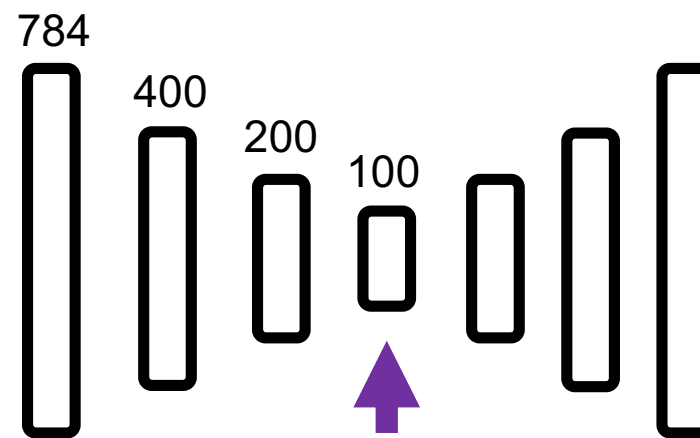
Architectures

- *Feedforward* *mini-bach.py*
- *Autoencoder* *auto-bach.py*
 - *Variational Autoencoder (VAE)* *VRAE*
- *Recurrent (RNN)*
 - *LSTM* *lstm.py, Celtic*
- Generative Adversarial Networks (GAN)
- Restricted Boltzmann Machine (RBM)
- Reinforcement Learning (RL)

Compound Architectures

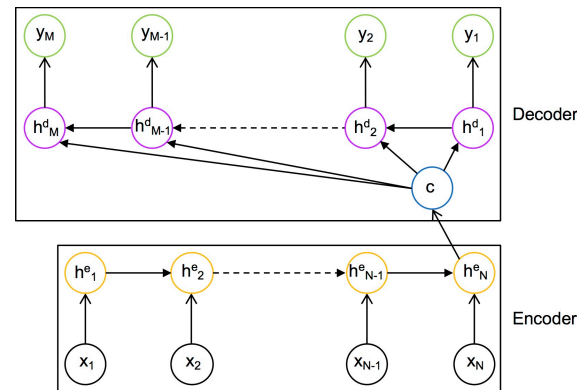
- Autoencoder Stack = Autoencoderⁿ

- DeepHear, auto-bach.py



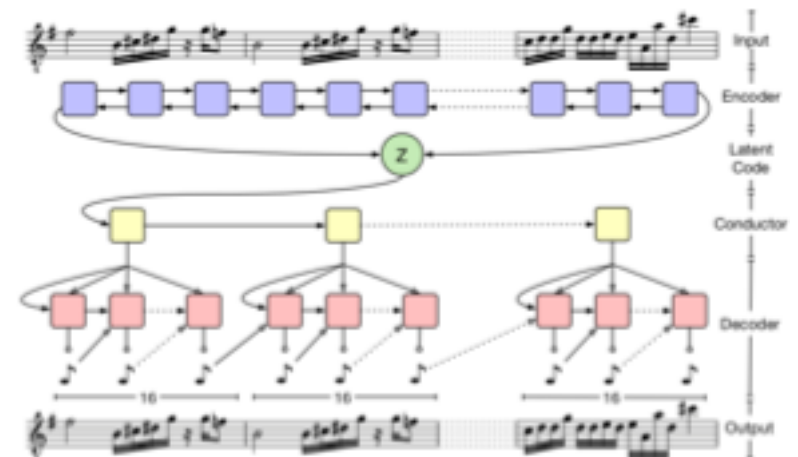
- Autoencoder(RNN, RNN) = RNN Encoder-Decoder

- VRAE



- RNN Variational Encoder-Decoder

- Music-VAE



Generative Adversarial Networks (GAN) [Goodfellow et al., 2014]

- Training Simultaneously 2 Neural Networks
 - Generator
 - » Transforms Random noise Vectors into *Faked* Samples
 - Discriminator
 - » Estimates probability that the Sample came from training data rather than from G
 - Minimax 2-player game

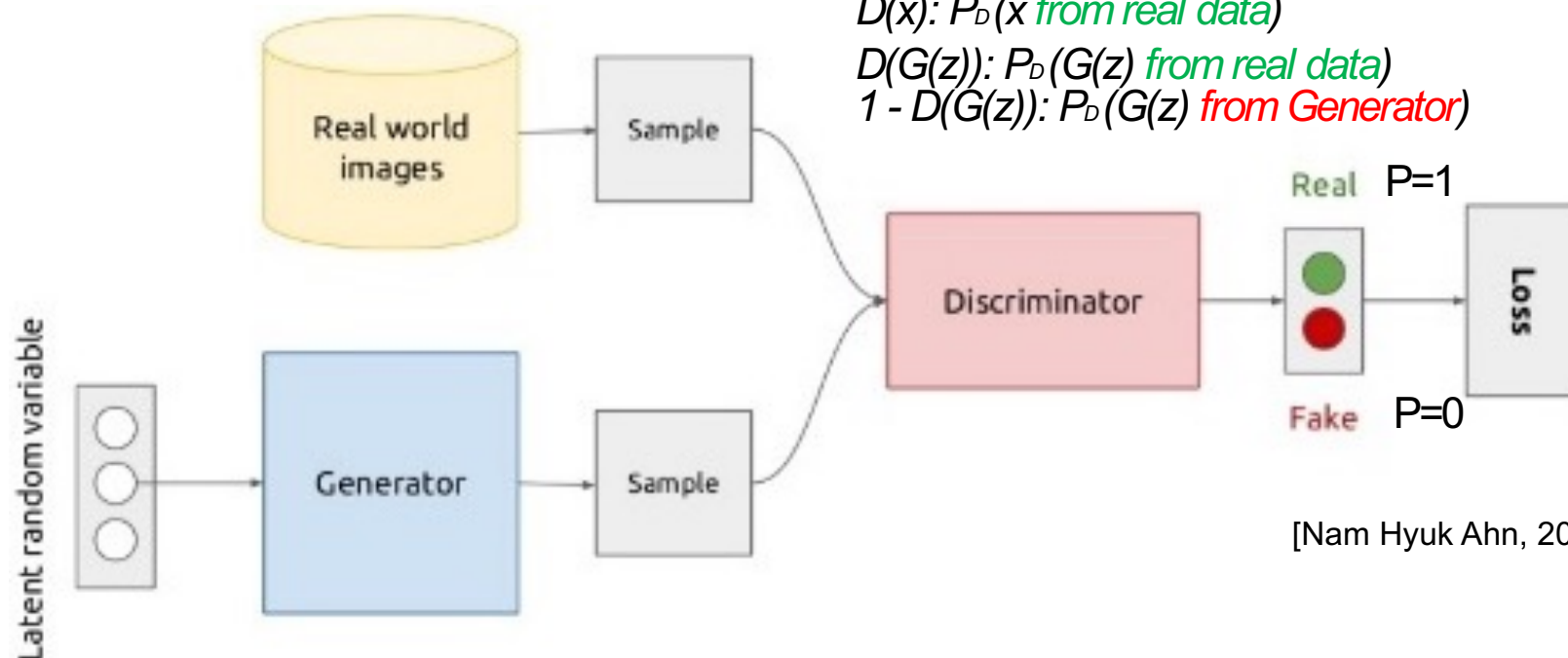
$$\min_G \max_D V(G, D) = \log(D(x)) + \log(1 - D(G(z)))$$

$D(x)$: $P_D(x \text{ from real data})$

$D(G(z))$: $P_D(G(z) \text{ from real data})$

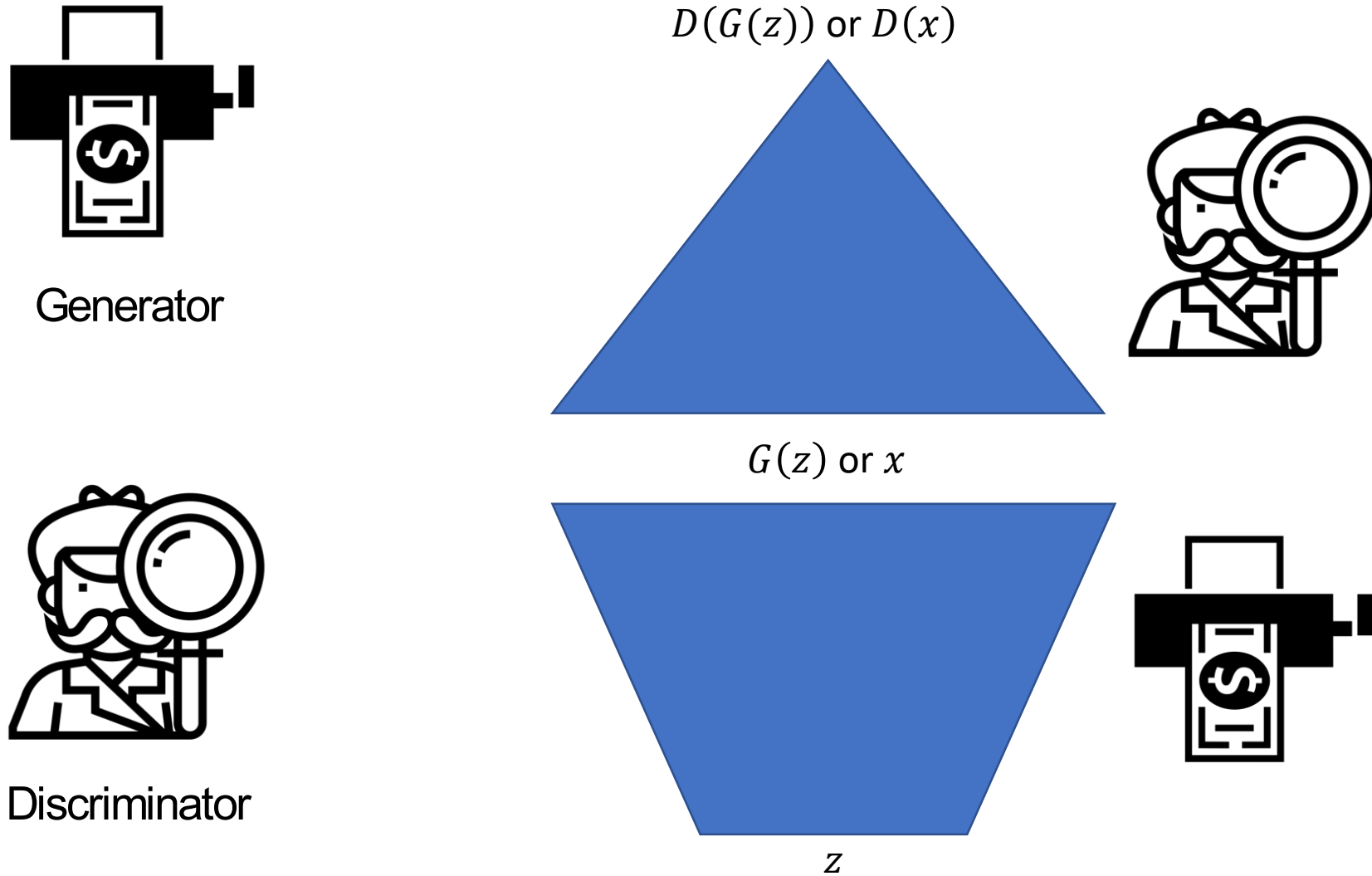
$1 - D(G(z))$: $P_D(G(z) \text{ from Generator})$

Prediction by D
(Correct)
(Incorrect)
(Correct)



[Nam Hyuk Ahn, 2017]

GAN and Turing Test



[Goodfellow, 2016]

GAN Basic Training Algorithm

- Initialize $\theta^{(G)}, \theta^{(D)}$
- For $t = 1:b:T$
 - Initialize $\Delta\theta^{(D)} = 0$
 - For $i = t:t + b - 1$
 - Sample $z_i \sim p(z_i)$
 - Compute $D(G(z_i)), D(x_i)$
 - $\Delta\theta_i^{(D)} \leftarrow$ Compute gradient of **Discriminator loss**, $J^{(D)}(\theta^{(G)}, \theta^{(D)})$
 - $\Delta\theta^{(D)} \leftarrow \Delta\theta^{(D)} + \Delta\theta_i^{(D)}$
 - Update $\theta^{(D)}$
 - Initialize $\Delta\theta^{(G)} = 0$
 - For $j = t:t + b - 1$
 - Sample $z_j \sim p(z_j)$
 - Compute $D(G(z_j)), D(x_j)$
 - $\Delta\theta_j^{(G)} \leftarrow$ Compute gradient of **Generator loss**, $J^{(G)}(\theta^{(G)}, \theta^{(D)})$
 - $\Delta\theta^{(G)} \leftarrow \Delta\theta^{(G)} + \Delta\theta_j^{(G)}$
 - Update $\theta^{(G)}$

Examples of GAN Generated Images



CelebFaces Attributes Dataset (CelebA)
> 200K celebrity images



[Brundage et al., 2018]

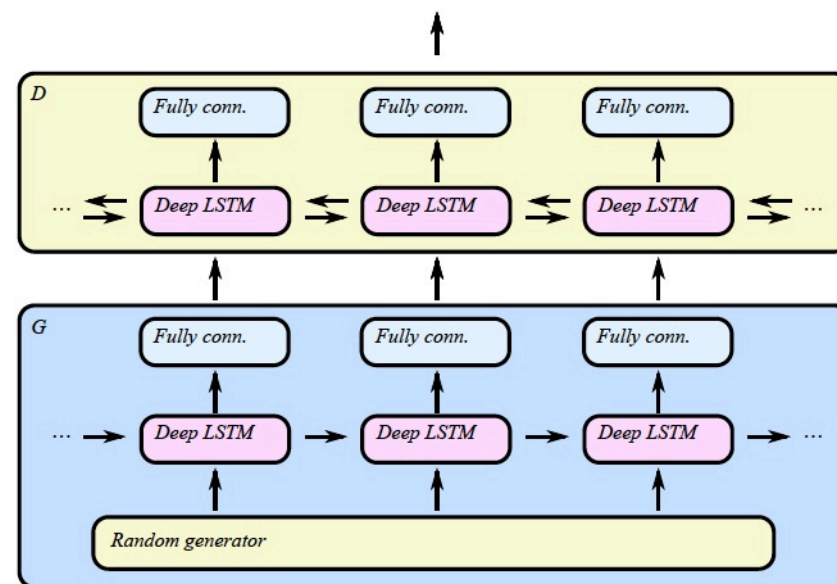
Synthetic (Generated) Celebrity images



C-RNN-GAN [Mogren, 2016]

GAN(Bidirectional-LSTM², LSTM²)

- Discriminator considers the hidden layers (forward and backward) values to be (or not) representative of the Real data
 - Analog to RNN Encoder-Decoder which considers the hidden layer as the summary of a sequence



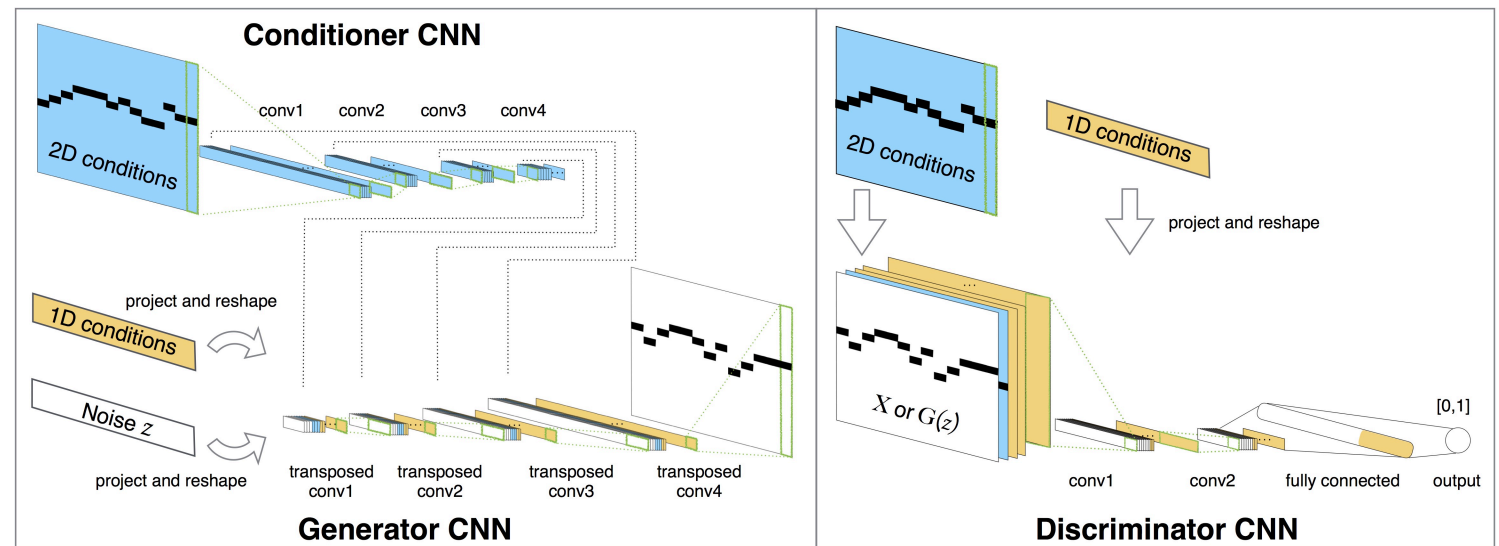
- Classical music Training Dataset



MidiNet [Yang et al., 2017]

GAN(Conditioning(Convolutional(Feedforward),
Convolutional(Feedforward(History, Chord sequence))),
Conditioning(Convolutional(Feedforward), History))

- Convolutional
- Conditioning
 - Previous measure
 - Chord sequence



- Pop music Training Dataset



<https://soundcloud.com/vqtsv6if5fwa/model3>

VAE vs GAN

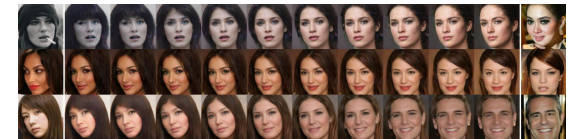
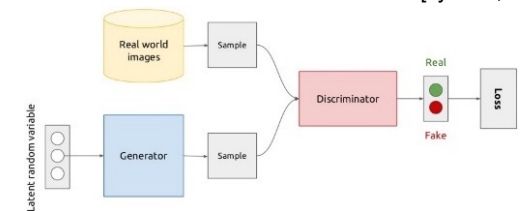
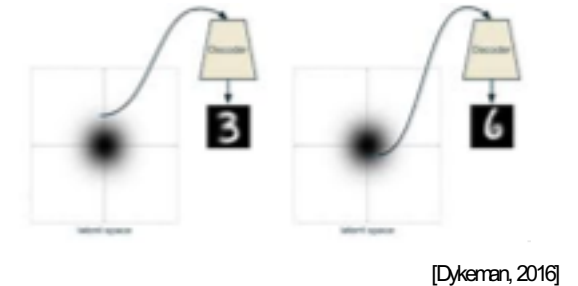
- VAE (Variational Autoencoder) and GAN (Generative Adversarial Networks)

Some Similarities:

- Are both generative architectures
- Generate from random latent variables

Differences:

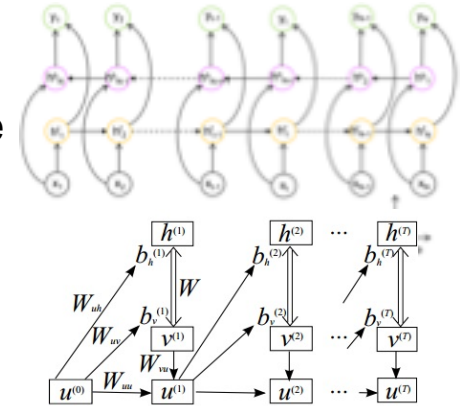
- VAE is representational of the whole training dataset
- GAN is not
- Smooth control interface for exploring latent data space
- GAN has (ex: interpolation) but not as for VAE
- GAN produces better quality content (ex: better resolution images)



Compound Architectures

- Composition

- Bidirectional RNN, combining two RNNs, forward and backward in time
- RNN-RBM [Boulanger-Lewandowski et al., 2012], combining an RNN (horizontal/sequence) and an RBM (vertical/chords)



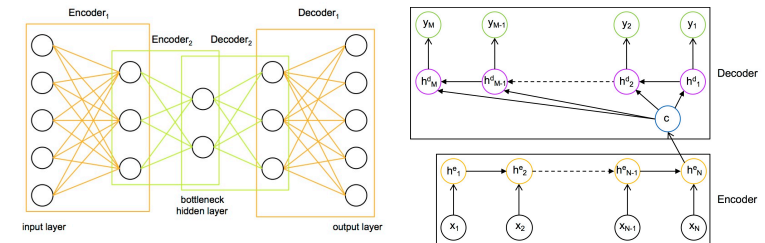
- Refinement

- Sparse autoencoder
- Variational autoencoder (VAE) = Variational(Autoencoder)



- Nested

- Stacked autoencoder = Autoencoderⁿ
- RNN Encoder-Decoder = Autoencoder(RNN, RNN)



- Pattern instantiation

- C-RBM [Lattner et al., 2016] = Convolutional(RBM)
- C-RNN-GAN [Mogren, 2016] = GAN(Bidirectional-LSTM², LSTM²)
- Anticipation-RNN [Hadjeres & Nielsen, 2017] = Conditioning(RNN, RNN)

