MusicGAN

A GAN capable of composing
little pieces of music

Dataset

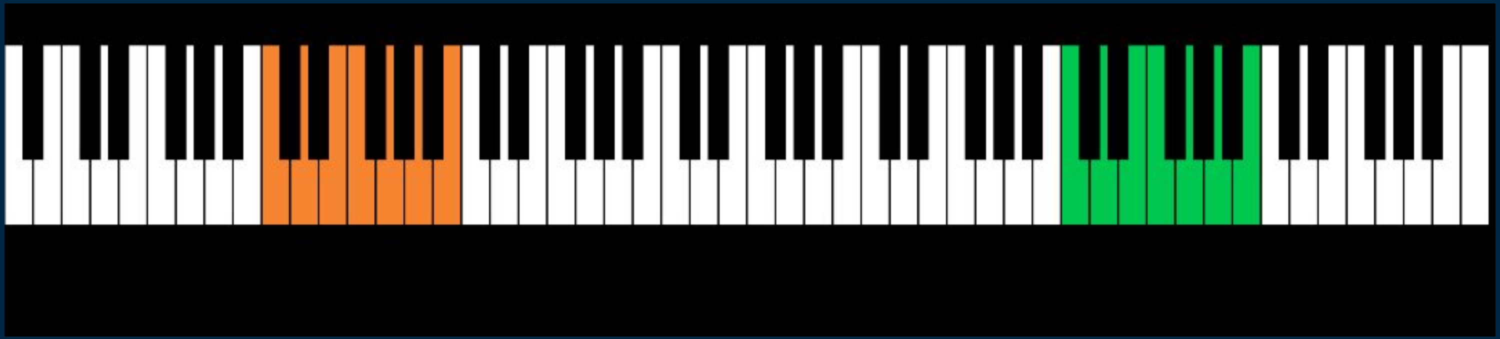
Type of data

- Great collection of piano rolls
- ~20000 different Multitrack objects
- For each beat of the song is known what notes are being played
- Five different instruments

Preprocessing

Reduced note range

- Most notes in a few octaves
- Improvement in memory and time
- Little loss of information

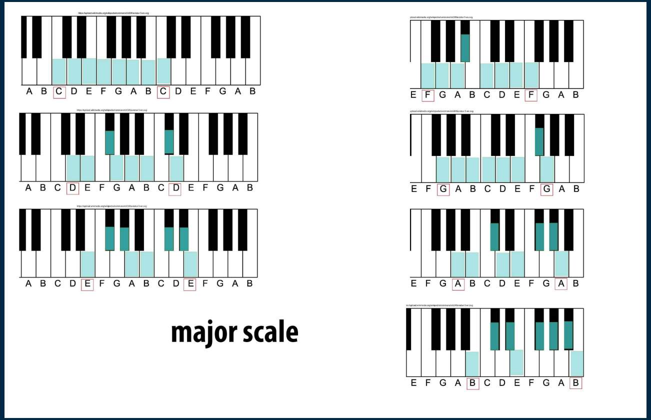


Normalization

- First experiments sounded weird
- Analysis showed different note distribution from dataset
- Train only on a single scale
- Music theory knowledge needed
- Source of claims is a domain expert

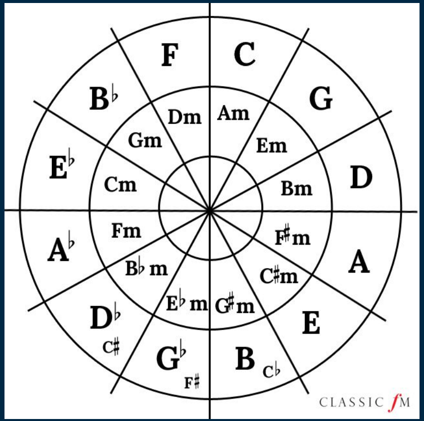
Major scales

- Typical note pattern
- All equal, but translated by a certain offset
- They sound the same to 99.99% of people
- Used in 75% of dataset



Major or natural minor?

- Same notes but with different roles
- Possible to distinguish with clustering
- Natural minor not really used in practice



Other scales

- Melodic/harmonic minor, pentatonic, hexatonic, blues, the list goes on
- Little representation for single type
- Decided to train on major scales

What was done

- Look at note frequency and determine scale
- If major scale, transpose every note, so that it becomes a C major song
- If other type of scale, discard

Approach

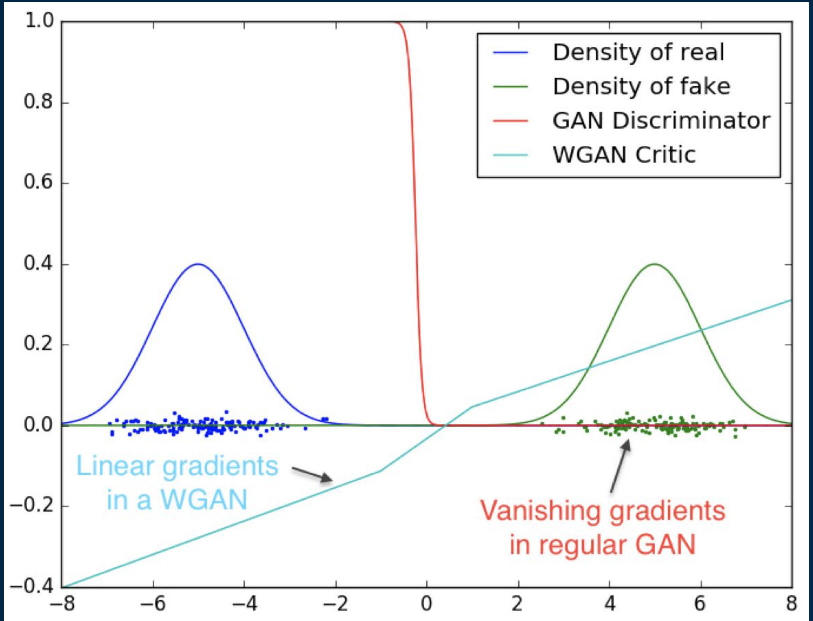
G

C

Wasserstein Metric

$$W(\mathbb{P}_r, \mathbb{P}_g) = \min_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma}\left[\|x - y\|\right]$$

Wasserstein Metric



Reference: Wasserstein GAN, Arjovsky et al.

Algorithm

Require: α, the learning rate. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration. β, the step-size of the gradient penalty.
Inputs: w_0, initial critic parameters. θ_0, initial generator parameters.

while θ has not converged **do**
  for $t = 0, \ldots, n_{critic}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples

$$g_w \leftarrow \nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$$

$$w \leftarrow w + \alpha \cdot RMSProp(w, g_w) - \beta \cdot \nabla_w g_w$$

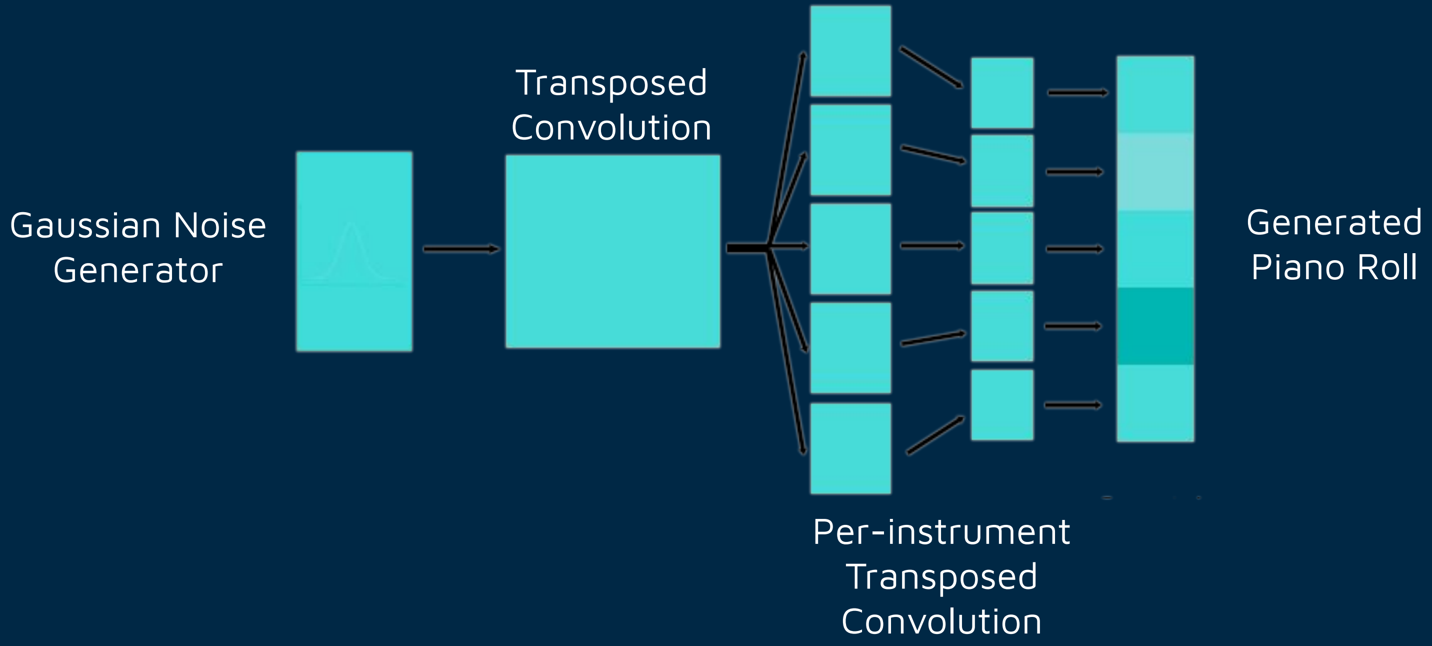
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples

$$g_\theta \leftarrow \nabla_\theta \left[\frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$$

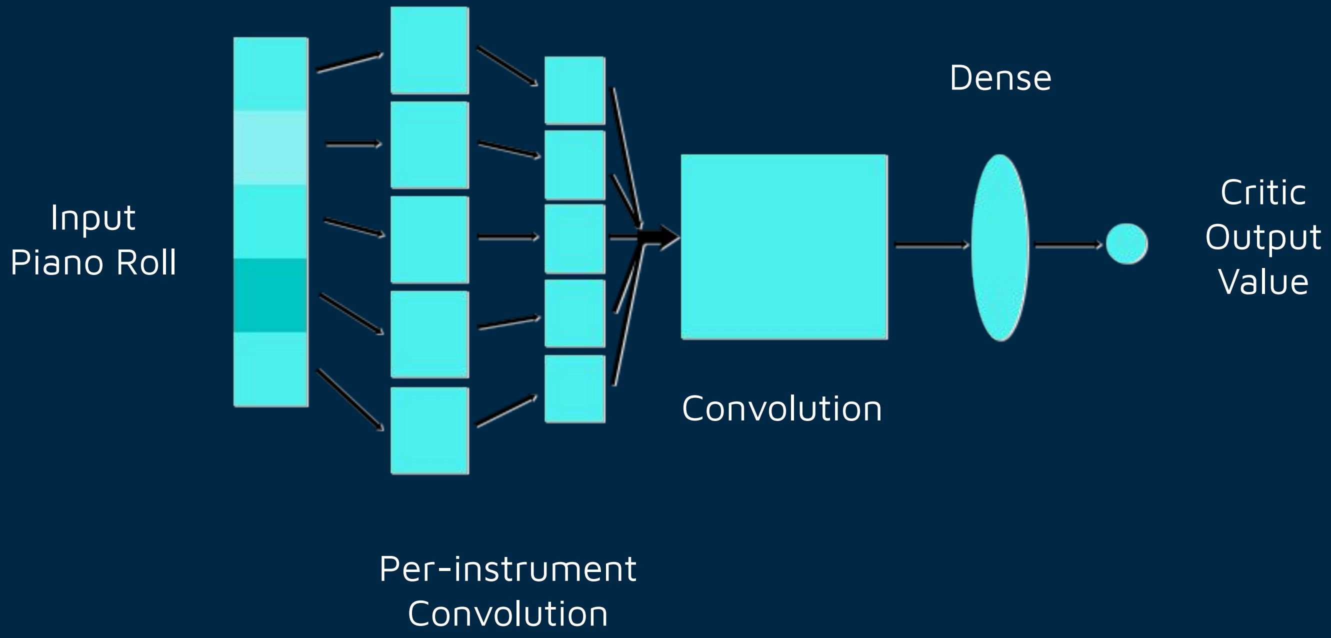
$$\theta \leftarrow \theta + \alpha \cdot RMSProp(\theta, g_\theta)$$

end while

Generator Architecture



Critic Architecture



Training & Results

What we expect:

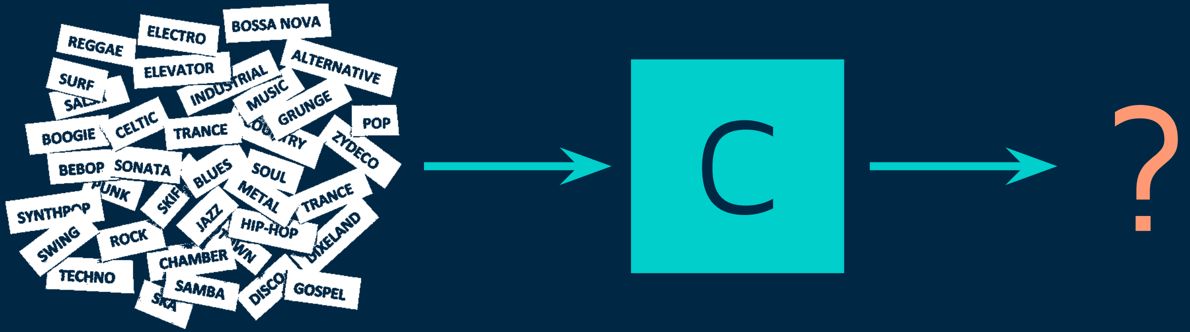
- Not an hit
- A catchy musical motif
- No dissonances and no random notes

Training details:

- “Supervised” training
- Google Colab for memory needs
- Random noise taken from a Gaussian Distribution
- 1:5 ratio of weights update between G and C

First attempt:

Give all the dataset to the network



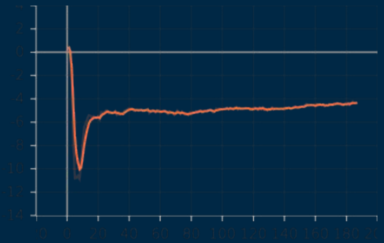
The critic is confused, there is no clarity in the dataset: its value is meaningless

Second attempt:

Helping the networks by training on **one genre**

Increased data **consistency**

The critic is more **reliable**

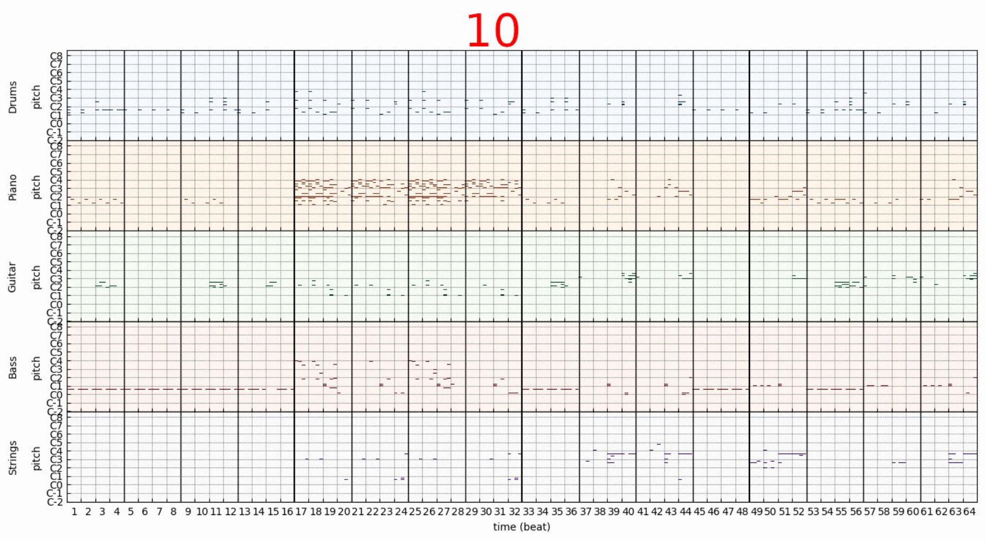


The generator work is facilitated (WGAN)

+

Diversification of the generated songs

Results analysis:



At the minimum of the G loss:

- Notes are less piled up
- Patterns became more definite
- Instruments are synchronized
- Consistent pauses