Deep Learning Score Based Generative Models

Stanislas d'Orsetti Jeremy Berloty

May 4, 2022

Abstract

This paper synthesizes the implementation we made of score based generative models, applied to music. We aimed to build a model able to generate a Mozart's sonate, based on techniques developped and presented by [SSDK+21] and techniques of music encoding, with variational autoencoder (VAE) processes, presented by [MEHS21]. We present the method used to achieve our goals, the results and a discussion on the evaluation of the results that can be made, even if a big part of evaluation, when regarding music generation, remains subjective

1 Introduction

Generative models generate new data instances as opposed to discriminative models, that learn to discriminate between different kinds of data instances. Applied to music, the idea is that a discrimative model could tell which composer wrote a certain music, whereas a generative model could generate a music similarly to a given composer. Formally, generative models capture the joint probability of $\mathbb{P}(X, Y)$, with X being the music and Y a composer in our example whereas discriminative models capture the conditional probability $\mathbb{P}(Y|X)$. This models can use deep learning methods, through the combination of generative models and deep neural networks The generative models can be grouped in two categories :

- The likelihood models, that estimate the above joint probability density by maximizing the likelihood
- The implicit generative models, where the main idea is to learn a mapping that generates samples by transforming an easy-to-sample random variable. The most popular example of this class might be the Generative Adversial Networks (GANs), that use adversial trainings, which can be however unstable

Deep generative models are a potential method for developing rich representations of the world from unlabeled data, directly from sensory experience. The human ability to imagine and consider potential future scenarios with rich clarity is a crucial feature of our intelligence, and deep generative models may bring us a small step closer to replicating that ability.

We focused on another class of generative models, which are score-based. The main idea of score-based models is to model a score function instead of the density function, the score function of a probability $p(x)$ being defined by $\nabla_x \log(p(x)$ The main idea is to learn a model $s(x)$ s.t. $s_\theta(x) \approx \nabla_x \log(p(x))$, θ being a learnable parameter parametrizing the probability density function of the data. The advantage of these models for generating data, regarding the likelihood and implicit models presented above are that

- for likelihood models, the gradient of the score-based model allow to get ridd of the normalizing constant when computing $p_\theta(x)$
- regarding GAN's, they achieve better convergence results.

[SSDK+21] showed that with stochastic differential equations, we can inject noise in the data and find a reverse-time equation depending on the above score, which can be estimated with neural networks. They showed that this method encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling and can achieve record-breaking performance for unconditional image generation on CIFAR-10.

Inspired by this framework, we aim to build a model able to generate music. We study how the methods described by the authors can be used for such a problematic, what limitations can be found and how we can improve this framework to meet our needs better by using VAEs, based on the work of [MEHS21] who showed how the sampling methods used in score-based generative models for image are limited but can however achieve good performance when using pretrained VAEs.

We will then show the results we obtained on a dataset containing all Mozart's sonatas and discuss how these results can bee objectively evaluated. We will also propose a framework that tries to unify the two approaches by training the model with a loss combining the VAE training and the score approximation, based on '[VKK21]

2 Score Based Generative Models

This section synthetizes the methods presented by [SSDK+21] and presented in https://yang-song.github.io/blog/2021/score/ in order to better understand how SBGMs work and how they can be applied to music generation

2.1 Sampling from the score function

We presented a general definition of score-based models in the introduction. How can it be applied to deep learning generative models ? Let suppose that we have trained a model $s_\theta(x)$ approximating the score function. There exists sampling methods that allows to sample from a distribution function $p(x)$ based only on the score function. The Langevin Dynamic is a procedure of this type. It is an iterative chain, initialized by a prior π_0 and that builds

$$\forall i = 1, ..., K, x_{i+1} = x_i + \epsilon \nabla_x log(p(x)) \text{ (which is approached by } s_\theta(x)) + \sqrt{2} z_i$$

z_i following a centered normal distribution Now the question is how can we estimate the score function. One of the challenge is that naive methods are innacurate in low density regions.

2.2 Perturbing data in discrete time

To avoid this issue, the authors present a method of multiple noise perturbations which consists in

- Using multiscale pertubations with increasing intensities, indexed by i, that can be represented by σ_i (for example σ_i can be the standard error of a Gaussian noise applied on $p(x)$ such that $p_{\sigma_i} = \int p(t)\mathcal{N}(x, t, \sigma_i^2 \mathcal{I})dt)$
- Training a score model for each i, $s_\theta(x, i)$, for example with a weighted sum of Fisher divergences
- Producing sample with Langevin dynamics for each i, with decreasing intensities σ_i

2.3 Perturbing data in continuous time

Now the idea is to use this method in continuous time, by increasing the number of noise scale to infinity. The noise perturbation procedure becomes a continuous-time stochastic process. Diffusion processes are solutions of stochastic differential equations, whose general form can be described by the following equation:

$$dx = f(x, t)dt + g(t)dw$$

with $f(., t) : \mathbb{R}^d \to \mathbb{R}^d$ the drift coefficient, $g(t)$ the diffusion coefficient and w a brownian motion. A SDE has alwas a reverse corresponding SDE.

$$dx = \left[f(x, t) - g^2(t)\nabla_x \log p_t(x)\right] dt + g(t)dw$$

with $p_t(x)$ being the marginal probability density function of the random variable $x(t)$. It is the equivalent of the p_{σ_i} that we described above in discrete time. So we see that there is exactly the score function in the reverse SDE. So for a continuous collection of random variables $\{x(t)\}_{t\in[0,T]}$, corresponding to increasingly perturbed data, one can rebuild $x(0)$ using reverse SDE, if one knows the terminal distribution $p_T(x)$ and every score functions $\nabla_x \log(p_t(x))$.

- The terminal distribution can be considered as close to a prior distribution
- $\nabla_x \log(p_t(x))$ can be estimated with a training objective defined by a continuous weighted combination of Fisher divergences (and not a sum like in the discrete case)

$$\mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(x)}\left[(t)||\nabla_x \log p_t(x) - s_\theta(x,t)||_2^2\right]$$

Once we estimate $s_\theta(x,t)$ for all t, we can solve the reverse SDE to find, conditional on $s_\theta(x,t)$ being properly estimated an approximate sample of $x(0)$, the original data.

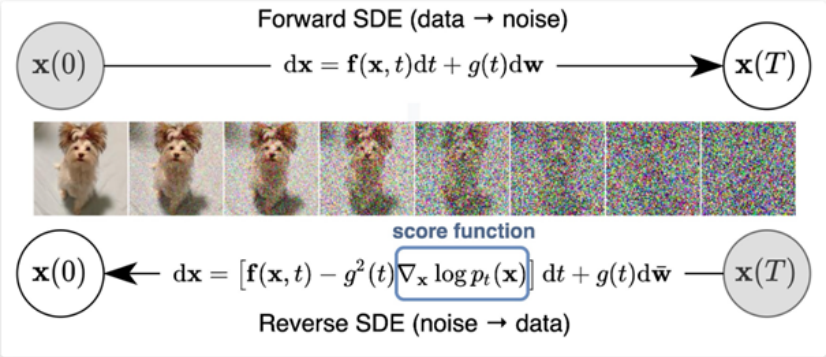


Figure 1: :Description of the processus

2.4 Sampling methods

But we can also use it for sample generation. For this purpose, we need to use numerical SDE solvers. One example of these solvers is the Euler-Maruyama method, which discretizes the SDE using finite time steps and small Gaussian noises. It is similar to the Langevin dynamic in the sense that it builds samples from x using its score function.

The authors propose however to improve the sampling by adding Markov Chain with Monte Carlo (MCMC) tools to the numerical SDE solver. They call them Predictor-Corrector samplers. The general philosophy is simple. First, they use a predictor, which is a numerical SDE solver able to predict $x(t+\Delta t)$ from $x(t)$, $\Delta_t \approx 0$ being a small negative time step, and a corrector, a MCMC procedure relying on the score function, such as Langevin dynamic. In detail, it consists in :

- Use the predictor to predict $x(t+\Delta t)$ with a properly chosen time step Δt
- Improve the sample $x(t+\Delta t)$ with the corrector by using $s_\theta(x, t+\Delta t)$

Finally, the authors propose a last sampling method based on ordinary differential equation (ODEs). Indeed, it is possible to convert any SDE into an ODE without changing the marginal distribution $\{p_t(x)\}_t$. So it is possible to sample from the same distribution as the reverse SDE by solving the ODE. The main advantages of this methods are that when replacing the score function by its approximation, the probability flow ODE (the corresponding ODE of a SDE) becomes a special case of neural ODE, which present a property of exact likelihood computation and therefore compute the data density p_0 from the known prior density p_T with numerical ODE solvers.

Now that we have presented the general framework, we would like to study how it can be applied for generative symbolic music.

3 Application to Symbolic Music

To apply the Score Based Generative Model to discrete symbolic music, we have to first encode the MIDI files into a continuous latent space thanks to a Variational Autoencoder (VAE). Indeed, as pointed by Gauttam Mittal et al (2021) [MEHS21] the diffusion models, as they were inspired by Langevin Dynamics sampling process, are only confined to continuous domains such as images and audio. The VAE therefore enables to model the discrete and sequential aspect of music partitions.

VAE are generative models that define $p(y, z) = p(y|z)p(z)$ where z is a learned latent code for data point y. It is comprised of an encoder $q_\gamma(z|y)$ which models the approximate posterior $p(z|y)$ and a decoder $p_\theta(y|z)$ which models the conditional distribution of data y given latent code z. The training objective is to maximise the log likelihood of $p_\theta = \int p_\theta(y|z)p(z)\,dz$ and we use the following bounding to approximate the posterior density : $\mathbb{E}[log(p_\theta(y|z))] - KL(q_\gamma(z|y)||p(z)) \leq log(p(y))$. The figure below show the overall architecture : for instance the midi partition is divided in 16 bars each one including 16 time stamps (that can be notes or silence). Each bars are projected to an embedding vector by the VAE, which are used to train the SBGM. The reversed process enables to generate new latent vectors that can be decoded by the VAE and then concatenated to output a new music partition. Inspired by https://github.com/Variational-Autoencoder/MusicVAE we adapted the MusicVAE to PyTorch so that it could be put together with the SBGM, also implemented on PyTorch.

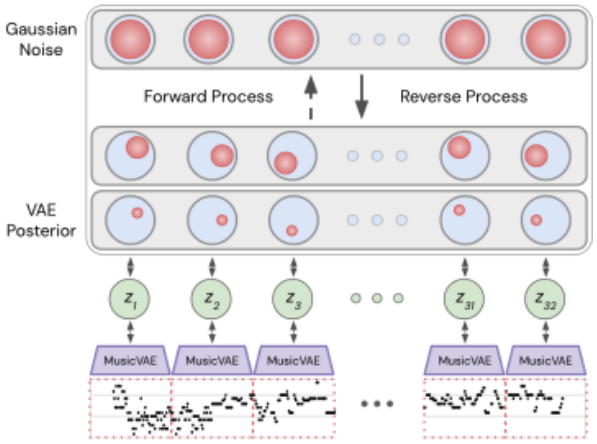


Figure 2: :Overall architecture

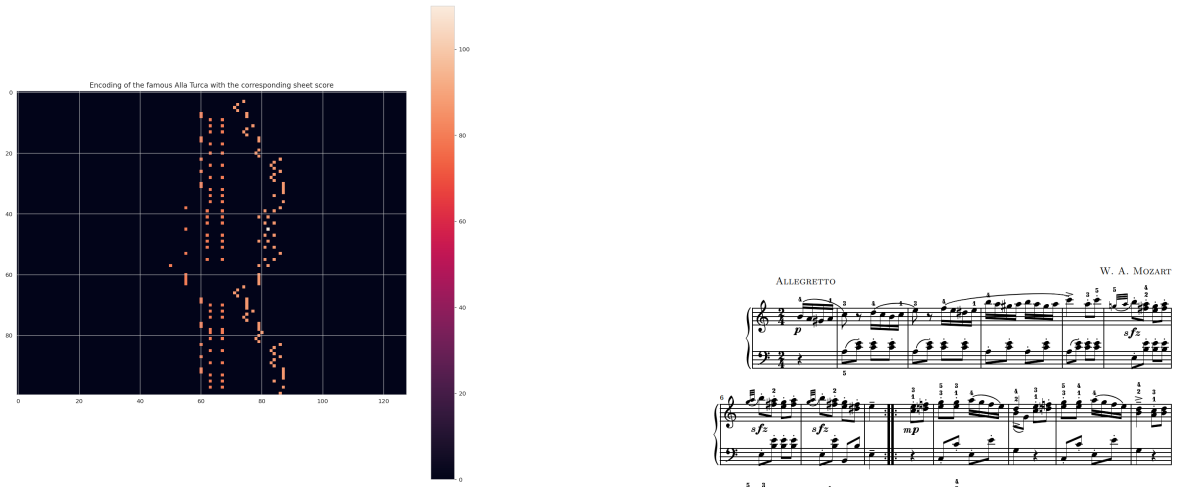
4 Results

4.1 Presentation of the database

The mozart's sonatas are at the number of 19 and were composed between 1774 and 1789, during the composer's most productive era. [Sac98] described them in the following way : "*Elles ont le tort d'être des modèles d'écriture, d'équilibre, de clairvoyance. Elles n'ont pas fourni leur lot d'expériences, comme celles de Haydn, puis celles de Beethoven. De la forme sonate à deux thèmes principaux, du plan en trois mouvements (allegro-andante-allegro), elles se contentent presque immuablement. Un allegro bourré d'idées contrastées, au rebours du monothématisme qui tente Haydn ; un andante de plan ABA ; un nouvel allegro de sonate ou un rondo plus ou moins élaboré ; et la sonate est ficelée, avec une aisance souveraine. Cette absence de conflit avec la matière [...], outre qu'elle leur confère leur transparence particulière, ainsi que cette rondeur, cet agrément sous les doigts devenus motifs à soupçon (comme si seule l'âpreté, la rugosité, voire la maladresse, étaient expressives !), leur donne une évidence qui peut tromper ; on les croit plus simples qu'elles ne sont, à tous les points de vue.*" We see with this description that these are very modelled and formatted pieces of music. Therefore, they happened to us to be interesting to study because we hope that we would be able to recognize pattern between the original music and the generated one, depending of the music being extracted from an Allegro, Andante or a Rondo. We found the midi files on the website kaggle.

4.2 Output

The first step when getting the midi files is to parse them and encode them into numpy arrays. We use the library pretty-midi for this purpose. We extract the piano rolls and obtain arrays with the time frame corresponding to the row dimensions and the keys of a keyboard as columns. The values corresponds to the intensity at which the note was played. Many process functions can be considered starting from there, knowing transposing all notes to the same key, scale all values to the same number,



(a) Encoded alla turca

(b) Alla Turca Sheet music

Figure 3: Encoded and original sheet music of the famous rondo of Mozart's 11th Sonata : Alla Turca

decompose the left hand and the right hand, considering them as different music pieces, or join them together. We tested the different processes to get the best encoding of the midi file. The music is also decomposed into bars, and the bars into notes. During the encoding process, the algorithm iterates on small time steps of the music and check which notes are played at this time. We can then build batches in which small samples of the music are stored, defined as the number of bars considered and the number of notes in each bars.

This encoding can then be projected in the latent space using VAEs as described in 3. The VAE was trained on 100 epochs using an ELBO loss.

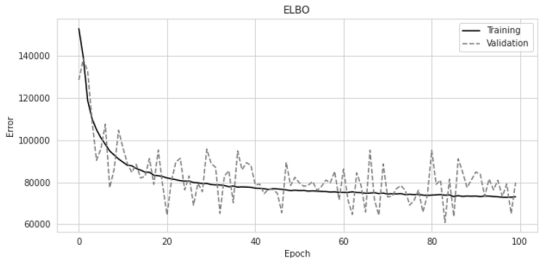


Figure 4: :Elbo Loss on train and validation set of the VAE training

We then train the score model on the projected array, sample from it, by testing the Euler sampler, and decode the output. We get therefore a new array in the same format of the initial arrays. In the resulting array, we see high density zones, were the model generated higher values. We can either choose to keep the highest value, if we want to generate a simple melody with only one note played at once, or define a threshold method to keep several notes, one note, or a silence, depending on the density distribution of the note on one row of the array. To start somewhere, we considered only the notes greater to the variance computed on each time frame.

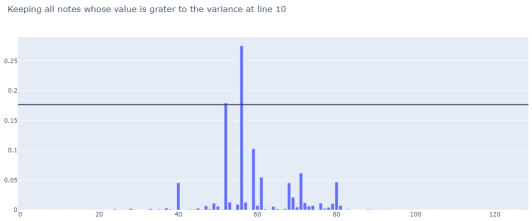


Figure 5: :Decision rule to keep sampled generated notes. The horizontal lines corresponds to the variance, the bars to the intensity on the note, and the x x axis corresponds to all possibles notes, like a keyboard

4.3 Evaluation

For the evaluation of our model we were inspired by [YL20], who defines several features to evaluate a symbolic music generation model :

1. Pitch count (PC): The number of different pitches within a sample. The output is a scalar for each sample.
2. Pitch class histogram (PCH): The pitch class histogram is an octave-independent representation of the pitch content with a dimensionality of 12 for a chromatic scale. In our case, it represents the octave-independent chromatic quantization of the frequency continuum.
3. Pitch class transition matrix (PCTM): The transition of pitch classes contains useful information for tasks such as key detection, chord recognition, or genre pattern recognition. The two-dimensional pitch class transition matrix is a histogram-like representation computed by counting the pitch transitions for each (ordered) pair of notes. The resulting feature dimensionality is 12x12.
4. Pitch range (PR): The pitch range is calculated by subtraction of the highest and lowest used pitch in semitones. The output is a scalar for each sample.
5. Average pitch interval (PI): Average value of the interval between two consecutive pitches in semitones. The output is a scalar for each sample.

We decided to focus on the Pitch Class Histogram and the Pitch Interval Histogram (PI without doing average), on 32 examples of original and generated midi files. The results are plotted below, they show great similarities. We could even ask ourselves if there a re not too similar : indeed the goal a generative model is to generate new datapoints, and in our case to generate new melodies. They are supposed to look like the melodies they were trained on, but they ought to be different, otherwise it would not be interesting.

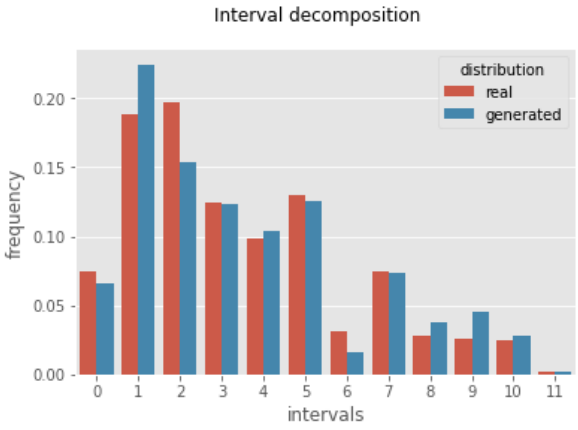


Figure 6: Pitch Interval Histogram

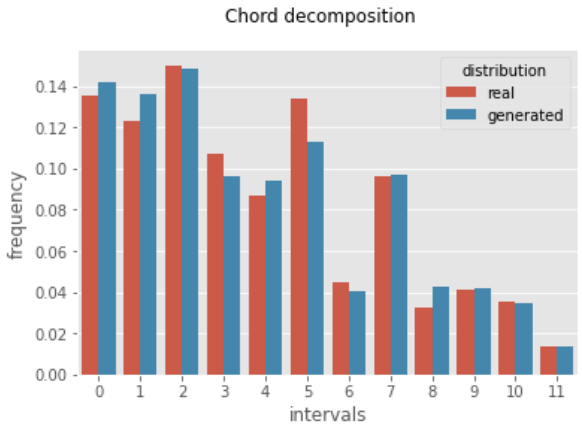


Figure 7: Pitch Class Histogram

5 Developments

As a development, we would have liked to make further tests, particularly to be able to be able to generate not only simple melody but a full music. One other idea that we would like to follow later is the one developed in [VKK21].They propose a Latent Score Based Generative Model, in which a novel approach that trains SGMs in a latent space, relying on the variational autoencoder framework. It seems therefore to be very close from our framework. However, the main difference is that they use the same loss for both the projection in latent space and the SDE-reverse training. More specifically, they

- introduce a new score-matching objective suitable to the LSGM setting
- propose a novel parameterization of the score function that allows SGM to focus on the mismatch of the target distribution with respect to a simple Normal one
- analytically derive multiple techniques for variance reduction of the training objective.

The idea of the paper can be understood with the following figure:

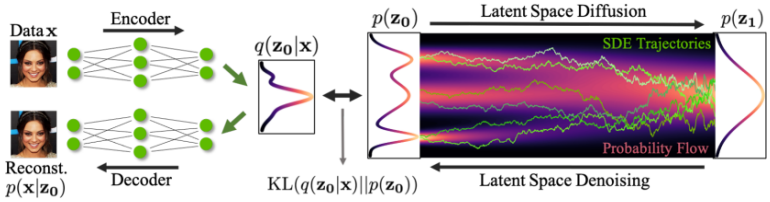


Figure 8: :LSGMs architecture

We see that data are mapped to latent space via an encoder $q(z_0|x)$ and a diffusion process is applied in the latent space ($z_0 \to z_1$). Synthesis starts from the base distribution $p(z_1)$ and generates samples in latent space via denoising ($z_0 \leftarrow z_1$). Then, the samples are mapped from latent to data space using a decoder $p(x|z_0)$. The model is trained end-to-end by minimizing the following loss function :

$$\mathcal{L}(x, \phi, \theta, \psi) = \mathbb{E}_{q_\phi(z_0|x)}[-log(p_\psi(x|z_0))] + KL(q_\phi(z_0|x)||p_\theta(z_0))$$

where $q_\phi(z_0|x)$ is the encoder, $p_\theta(z_0)$ is the SGM prior, and $p_\psi(x|z_0)$ is the decoder. Following the VAE approach, $q_\phi(z_0|x)$ is an approximate of the true posterior $p(z_0|x)$.

We unfortunately did not have the time to implement this model but we think it would have been intersting for our objective.

6 Conclusion

As a conclusion, we focused on score-based generative models to generate music, based on Mozart's sonatas. We tested use this model in a latent space, using a pretrained Variational Autoencoder. Results were subjectivelly satisfying. We discussed also about the possibility to evaluate in a more objective way, with precise criterions, a generated music. We think that our model could be improved by focusing on the following bullet points :

- Test different preprocessing of the midi file
- Try to use this method on other composer musics and evalute those generation with objective criterions.
- Increase the difficulty of the generation, by trying not only to generate "one-note-at-a-time" melodies, but more complex melodies by using rules decisions based on threshold (eventually learned with machine learning algorithms) on the reprojeted generated data to be able to generate silences, chords and both hands playing.
- Modelize an LSGM architecture.

7 Annexes

Please visit our github at https://github.com/sdorsetti/DL_SBGMproject/tree/main
Please visit our shared colab with explications and implementations of the models at https://colab.research.google.com/drive/1UlHGcRHQDY5uSIO50JA-jtRYaueu2KSv?usp=sharing

References

[MEHS21] Gautam Mittal, Jesse H. Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *CoRR*, abs/2103.16091, 2021.

[Sac98] Guy Sacre. La musique de piano. 1998.

[SSDK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. 2021.

[VKK21] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. 2021.

[YL20] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32, 05 2020.