MASTER ATIAM :

PROJET MACHINE LEARNING

Learning controls and interactions for DDSP

Encadrants :
Philippe Esling
Antoine Caillon
Adrien Bitton

Etudiants :
Yu Ji
Jeremy Uzan
Argan Verrier
Georges Le Bellier
Gabriel Dubus

2020-2021

Table des matières

1 Introduction

1.1 Motivations

In a lot of Machine Learning problems, we try to reduce the dimensionality of the data. Variational Autoencoder (VAE) is a very popular method that is used in many recent papers.

Historically, Autoencoders takes an input data (image, vector) with a very high dimensionality and computes it into a neural network which tries to compress the data into a smaller representation. The neural network is composed of two main modules. First, the encoder is made of several layers (which can be fully connected or convolution ones) which takes the input and compresses it down to a smaller representation. This method is named the Bottleneck.

From the bottleneck, the decoder part of the neural network tries to reconstruct the input by using again fully connected or CNN (Convolutional Neural Networks) layers.

The loss function is defined as the distance between the reconstructed version and the input. Through a pixel to pixel comparison, a loss function can be derived. In that way, we are able to create a way to compress the data that minimize this function.

The idea behind the Variational Autoencoder model is that instead of mapping any input to a fixed vector, we want to map our input onto a distribution. The unique difference is that the normal bottleneck z is replaced by two separate vectors. One is the mean of the representation and the second represents the standard deviation of that distribution.

In order to build the reconstruction of the input, we need to sample from the probabilistic latent distribution and then decode from the result. In this case, the loss function is formed from two terms. The first term represents the reconstruction loss (same as in the autoencoder model), and the second part is the Kullback-Liebler (KL) divergence. It ensures that the distribution the model is learning is not too far removed from a Normal distribution. The second term thus forces the latent distribution to be relatively close to a normal distribution with mean of 0 and standard deviation of 1, as well as ensuring a regular enough latent space..

1.2 Literature review

In this section we present the bibliographic study that took place prior to the project, and was based on the provided documents.

Despite years of research, synthesis of realistic sounds (compared to the instrument to mimic) remains a daunting task. In recent years, the development of concatenative synthesis [1] allows one to synthesize relatively realistic sounds by concatenating audio samples extracted from recorded files. However this method implies the access to a large number of audio files and the results might sound unrealistic compared to real instruments.

Recently, the DDSP library [2] has provided high quality results in term of realism. This deep learning approach allows one to include classical DSP modules such as additive synthesizer, filters or delay, in machine learning models. However one remaining issue is the expressiveness and the controls the user has over the out-coming sound. Indeed, in order to produce sound DDSP needs continuous flow or control signal, which is incompatible with lower resolution controls such as midi signals.

This issue is addressed by the authors of [3] by introducing a second model which learns to generate envelopes of pitch and velocity given a sequence of midi notes, which are then used as input to DDSP. The two models are trained separately : the envelope generating model is trained with midi score obtained from audio recording, and the DDSP model is trained based on the envelopes extracted. The addition of a learning model for the envelope generation allows for more flexibility, moreover it achieves highly realistic results with only few minutes of audio. Another approach is presented in [4]. Here the authors propose a generative model which learns to generate pitch and loudness contours, based on individual midi notes extracted from audio files *via* CREPE algorithm. The results were found realistic and expressive, though some artifacts may appear as well as some off-pitch notes and loop-like effect during hold notes.

2 Getting started with VAEs : MNIST - FashionMNIST

In this section we present the work done prior to the development of the envelope generating VAE. In order to get started and used to VAEs, we firstly developed a model trained to encode and decode pictures from both MNIST and FashionMNIST. Firstly, we detail the implemented models, then we present the results and latent spaces we obtained.

2.1 Training and Test

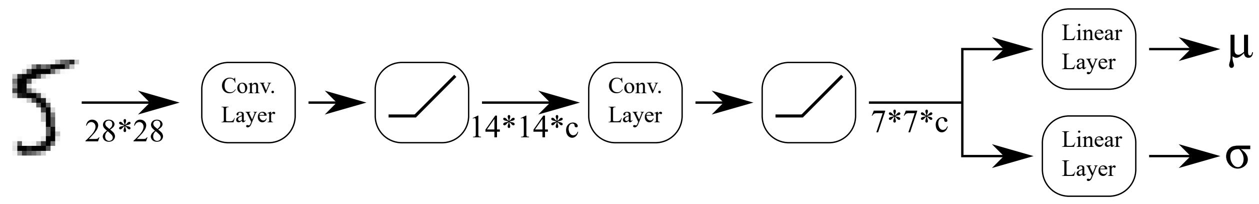
The MNIST database consists of labelled handwritten digits. It contains 70 000 training 28×28 pictures, 60 000 of which were used as training samples. The remaining ones were used for the test.

On the other hand, FashionMNIST is made of 28×28 pictures of articles from Zalando. The training and validation samples are in the same proportion as in the MNIST case.

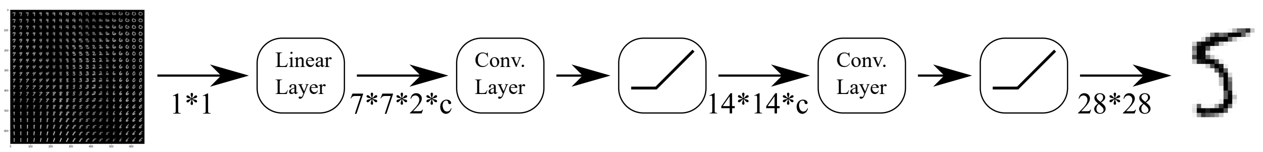
The implemented VAEs can be found in two notebooks entitled `MNIST_VAE_Conv2d.ipynb` and `FashionMNIST_VAE_Conv2d.ipynb`.

The encoder was implemented by two convolutional layers with rectified linear activation functions, and two fully connected layers, as presented in figure 1.

Some results are presented in figure 2 for both datasets. Although the generated pictures may seem blurry, which is likely due to the model being a VAE, it appears our model achieve good reconstructions of the validation samples.

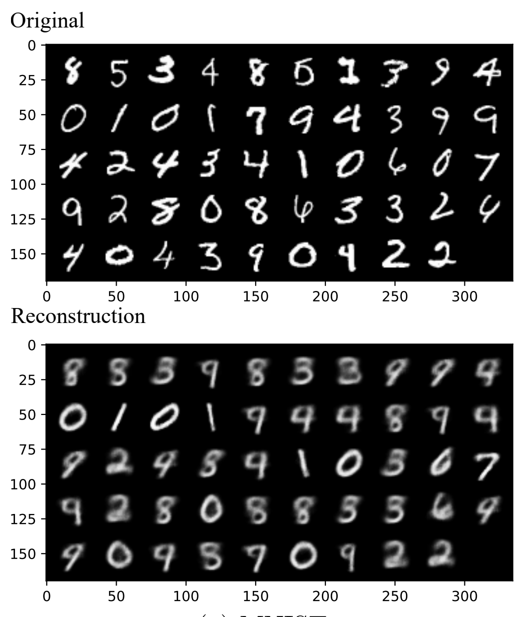


(a) Encoder

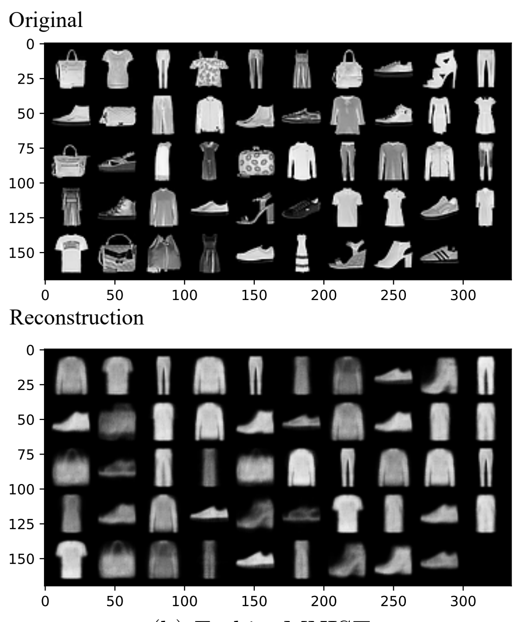


(b) Decoder

FIGURE 1 – Implementations of both encoder (a) and decoder (b).μ and σ are respectively the mean and standard deviation of the latent distributions. The annotations indicate the dimensions at the current stage.



(a) MNIST



(b) FashionMNIST

FIGURE 2 – Comparison between target samples (top) and generated samples (bottom), both for MNIST (a) and FashionMNIST (b). The *capacity* c was set to 64.

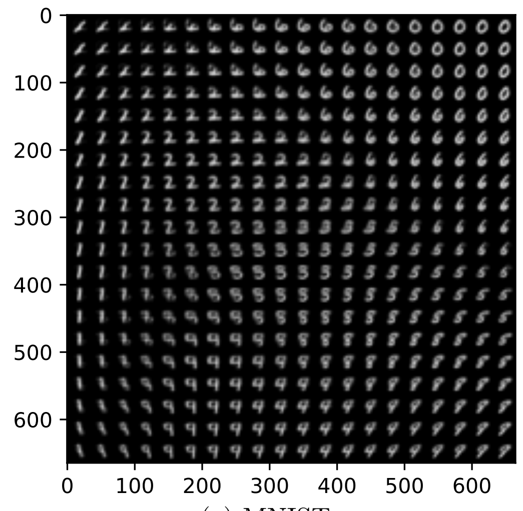
The more we use dimension in the latent representation, the sharper and clearer will be the reconstructed digits. The latent space size of the above trained network was only 2.

2.2 Problems and improvements

Firstly, we did not obtain an accurate reconstruction of our input picture and our final loss didn't diverge gently. We decided to draw the curve of our losses separately during the training. We observed that our KL divergence loss decreases quickly but the reconstruction loss does not, which is because the reconstruction loss is "too strong". As a result, we added a "variational β" as a parameter that adjusts this strength as 0.01 multiply with the term of the KL divergence. Secondly, we changed the function $softplus$ to linear for the log variance computation in the decoder because log variance σ can be negative. Also, we deleted the $ReLu$ function from the average μ as it can be positive or negative in the value of a Gaussian contribution. Finally, after trying with different parameters like : learning rate, batch size, we obtained our final result.

2.3 Navigating the latent space

The encoder has been trained to compress the images (clothes, digits). A 2-dimension representation of the latent spaces obtained for both datasets are presented figure 3


(a) MNIST

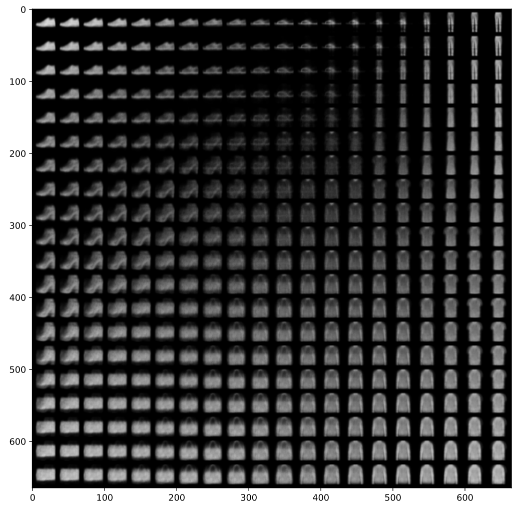

(b) FashionMNIST

FIGURE 3 – Latent spaces obtained for MNIST (a) and FashionMNIST (b) datasets

The interesting aspect of VAEs is to interpolate in the latent space. We picked two random vectors (images) of the database (on figure 4, digits 1 and 7 are taken), projected them in the latent space and choosed a point in between them to observe what vector the decoder would reconstruct from it.

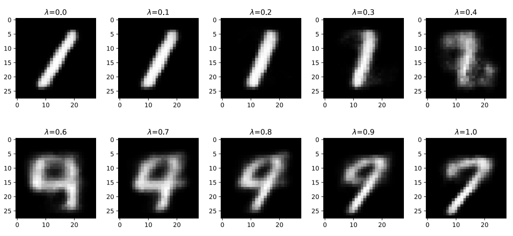
We interpolated in the 2-dimensions latent space by travelling from the latent representation of a first picture to the latent representation of a second picture. Therefore, the coordinates of the interpolated point may be expressed as

$$(x_{interp}, x_{interp})^t = \lambda(x_1, y_1)^t + (1 - \lambda)(x_2, y_2)^t \tag{1}$$

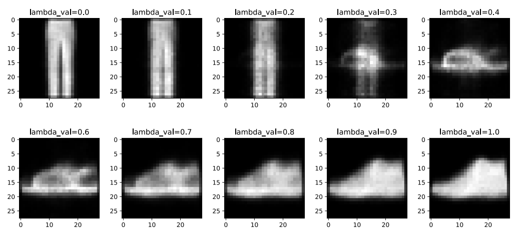
Where $(x_{interp}, x_{interp})^t$ are the coordinates of the interpolated point, and (x_1, y_1)(respectively (x_2, y_2)) the coordinates of the image 1 (respectively 2), and $\lambda \in [0, 1]$ is a scalar factor.

Figure 4 presents pictures generated from interpolated samples for both datasets. Figure 4a shows interpolations from digits "1" to digits "7" : as we can see, the generated picture gradually shifts from one to another. Interestingly, it appears that the digits "9" and "4" lies in-between these two samples in the latent space.

Figure 4b allows to observe one drawback of VAE models : even though these types of models allows one to interpolate between two samples, there is no consistency over the generated data. Contrary to GANs, VAEs exclude the constraint to retrieve the original sample. Therefore, interpolated pictures are likely to be superpositions of the starting and destination picture. This can be seen in figure 4b : the interpolated images are superpositions of trousers and a shoe.



(a) MNIST



(b) FashionMNIST

FIGURE 4 – Images generated from interpolated points in the latent space, both with MNIST (a) and FashionMNIST (b).

VAE also allows us to generate new images by sampling latent vectors and sending them to the decoder.

3 VAE for fixed-lenght samples

3.1 Introduction

The goal of our project is to generate expressive pitch and loudness envelopes. Thus, the first step was to design a network capable of reconstructing fixed length envelopes. The length was arbitrarily set to three seconds sampled at 250Hz (750 samples). As in the previous part, we used variational autoencoder to obtain a latent space based on training data, from which we could sample points fed to the decoder in order to generate expressive envelopes.

The training data was the *solordinario* dataset, which consists of individual notes played by a violonist with varying nuances and strings. The samples were cropped to three seconds. The pitch and loudness envelopes were extracted using the CREPE package [5]. These couples of envelopes are used to train the VAE to encode the input into the latent space. We then sample from the latent space to reconstruct the input. The overall approach is pictured in figure 5.

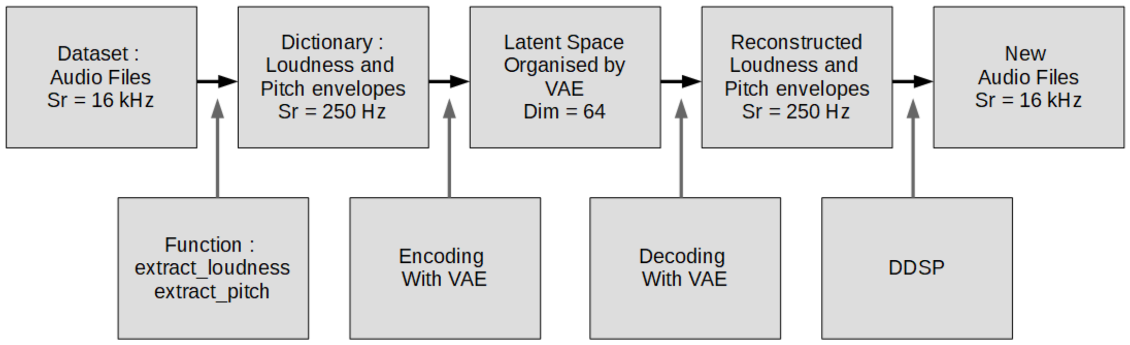


FIGURE 5 – Overview of the project. The base material is a dataset containing single notes played by a solist, sampled at 16kHz. Pitch and loudness are extracted using provided functions, and are used to construct the training dataset, sampled at 250Hz. The VAE is trained to encode and decode these envelopes to and from a 64-dimensionnal latent space. The decoded envelopes are fed to a DDSP in order to produce a sound sample.

3.2 Datasets Preparation

Solordinario dataset is a database of 252 wav recordings of single notes of violin. The idea was to cut the samples in order to obtain uniform three seconds samples. After that, we used the given functions extract_loudness and extract_pitch to obtain a pair of envelopes for each sample. After the outlier samples have been taken out, we obtained a dictionary "data_dict.npy" that we used for the training of the neural network.

To resolve the problem of the small dataset size, the team decided to increase the database with 200 more samples. The violin notes were synthesized using *Ableton live* with the free plugins *Sonatina* and *VSCO2*. We then exported them to .wav format. The samples were then labeled with the corresponding nuances and notes. We obtained an augmented dictionary.

3.3 VAE implementation

3.3.1 Architecture

To simplify the implementation, we first went back to a simple autoencoder, and moved to a VAE when the neural network started to learn correctly. With the MNIST dataset, we used 2-dimensional convolutions as our input and output data were two dimensional (Image data). Here, the input and output data are one dimensional. Indeed, the input is the envelopes of pitch (time, frequency) and loudness (time, amplitude). Thus, we used 1-dimensional convolution layers.

Mixing convolutional and linear layers allowed for an accurate reconstruction (please refer to section 3.4). The encoder is composed of 5 convolutional layers with ReLu activation functions and 2 linear layers with ReLu activation functions, as pictured in figure 6. .

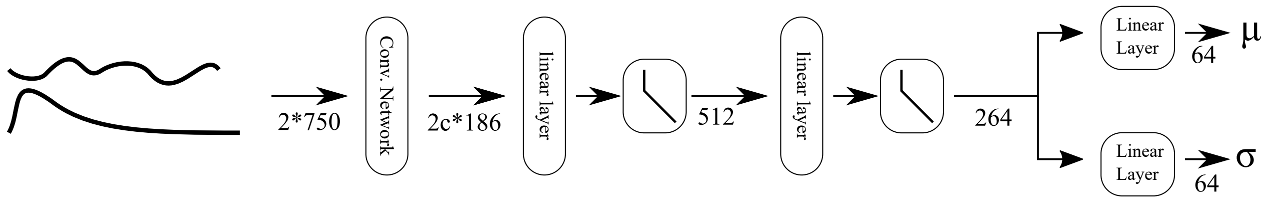


FIGURE 6 – Architecture of the encoder. The network learns to encode a couple of 750 samples long envelopes into a 64-dimensional space. The latent distribution is defined by its mean μ and its standard deviation σ. The *capacity* c was set to 2. The annotations indicate the dimensions at the current stage.

The input couple of envelopes is encoded into a 64-dimensional space. At first the decoder was made of `ConvTranspose` layers, that were then replaced with `conv1D` along with `upsampling` layers in order to avoid overlap artifacts. Indeed, these artifacts are responsible for checkerboard apparitions in the case of images and may affect the reconstruction of envelopes.

Every layer except the last one ends with a Batch normalization, which stabilizes the learning process [1] as detailed in [6]. The architecture is pictured in figure 7 .

1. `https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c`

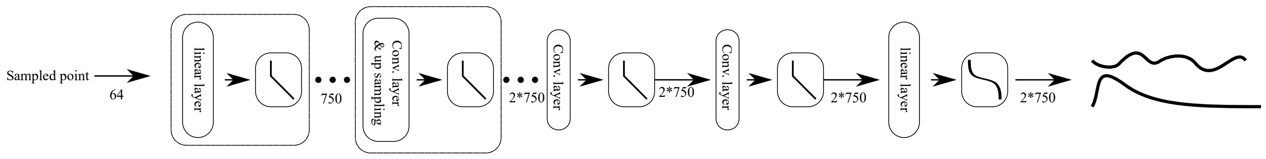


FIGURE 7 – Architecture of the decoder. The ellipsis indicates a cascade of the previous block. We use a cascade of three linear layers with ReLu activation, and a cascade of four convolution layers with up-sampling and ReLu functions to gradually increase the dimension of the sampled point in order to generate the associated envelopes. The annotations indicate the dimensions at the current stage.

3.3.2 Hyperparameters Tuning

After having studied the impact of the hyperparameters on the learning process of the network, and tried many different ones, we eventually found values that gave us the best results for the reconstruction of our samples. The process of tuning the parameters was empiric : we tried different sets of parameters and chose ones by comparing the reconstructions, and the loss graphs. The *capacity* is very close to the model complexity. It reflects how complicated a pattern or relationship a model can express. We set it to 2.

Since the VAE consists in encoding the input data into a lower dimension space, the dimensionality of the latent space is a key parameter. A low value may exclude important information whereas a high dimension may increase the complexity for no tangible results. This parameter was also chosen empirically, and was eventually set to 64.

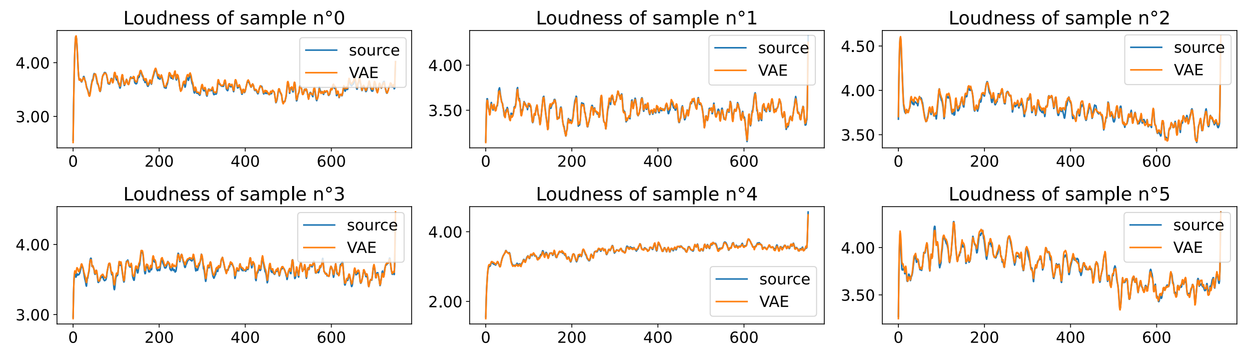
From our testing, a training on 3000 epochs showed good reconstructions.

The batch size is the number of samples processed before the model is updated. We took 25.

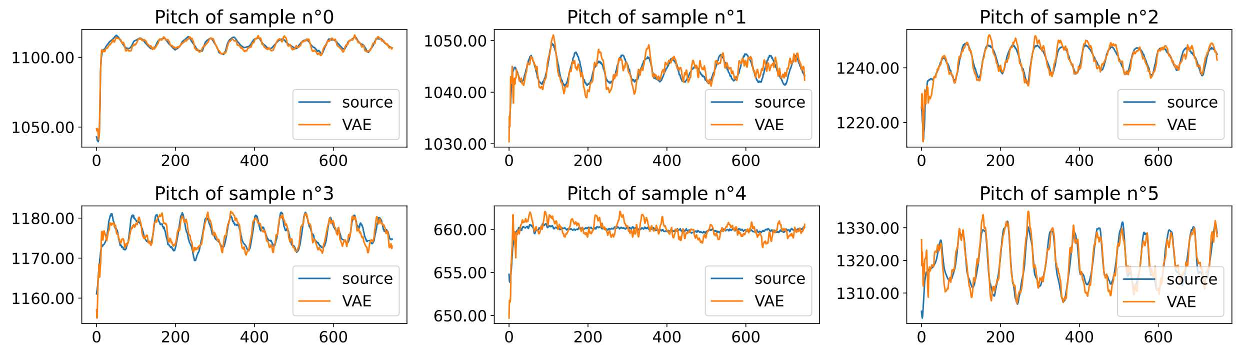
The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated. We set it to 5×10^{-5}.

3.4 Results

Similarly to the MNIST, we ploted the two images of envelopes from the Test Dataset and compared it with the reconstruction obtained with the VAE. These results are presented figure 9. The trained model is able to make a realistic reconstruction of the Loudness and Pitch envelopes.



(a) Loudness



(b) Pitch

FIGURE 8 – Source Envelope image of Loudness (a) and Pitch (b) in blue and its reconstruction in orange

The trained model `vae_model_train` can be loaded in `VAE_Solordinario_ExploreLatentSpace.ipynb`. The pitch is not perfectly reconstructed, but we can still observe the light modulations. Furthermore, we observe from the loudness envelopes that the model may have overfitted the dataset.

4 Latent space visualization and generation

4.1 Generation

As we saw with MNIST dataset, the VAE organizes the latent space and allows the generation of new pitch and loudness envelopes [7]. Similarly to the images of digits and clothes, we randomly picked two pairs of images (pitch-loudness) in the latent space and generated brand new pairs from them. After that, we used the pre-trained DDSP model to generate the samples from the pairs of envelopes and listen to them. [2].

2. `https://jeremybboy.github.io/DDSP/`

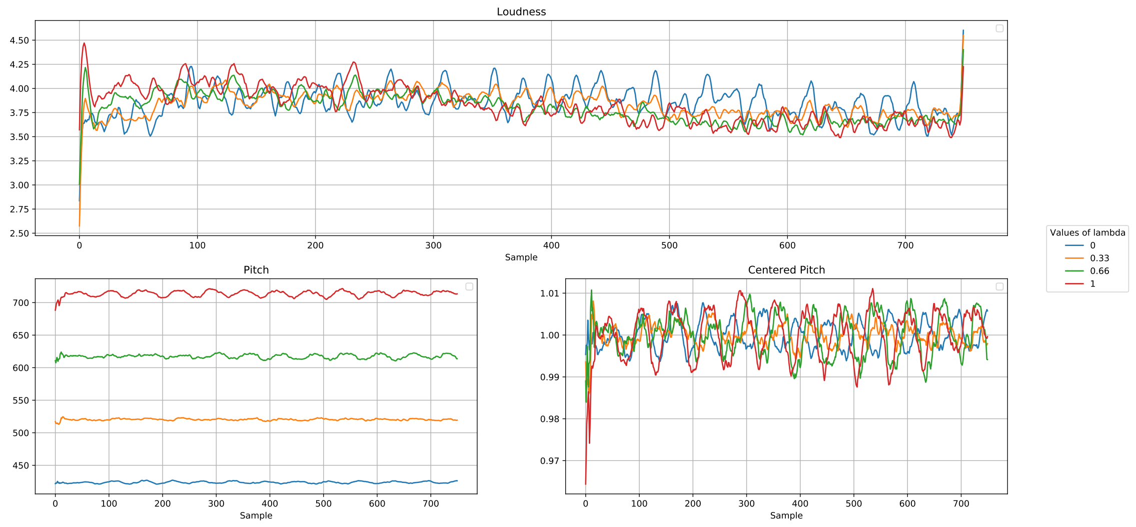


FIGURE 9 – Pitch and loudness envelopes computed with the decoder after interpolation in the latent space. The parameter λ is the one used in equation (1)

4.2 Visualization

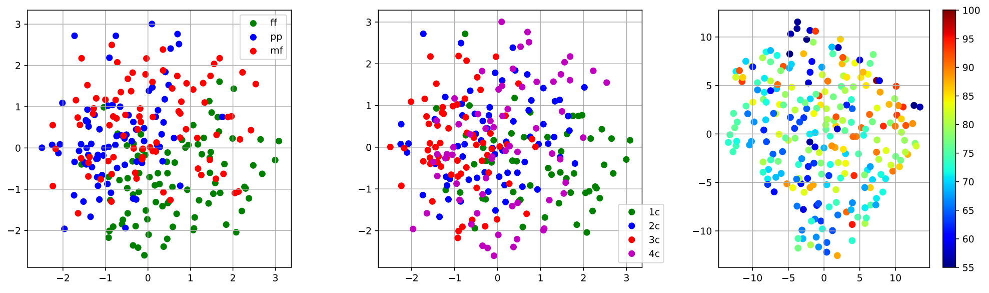
Our VAE latent dimension was 64, so we implemented a Principal Component Analysis (PCA) to reduce the dimensions and visualize the latent space in a two-dimensional graph.

To begin with, we had a look on the dynamics (loudness) latent space organization. The dataset was composed of pianissimo, fortissimo, and mezzo-forte labeled samples. The process to compute those data into the latent space was simple. We took these three different labeled samples and encoded them with our `vae_trained_model`. Samples were then organized in the 64 dimensional latent space, and the PCA reorganized them into a two-dimensional graph that can be observed and interpreted.

It seems that the bottom right is dedicated to organize *forte* samples. This last observation then opens the prospect of setting up a synthesizer based on this model, on which the played notes would come from selected points in a latent space.

Then, we performed the same process for the MIDI pitch and for the string played. If the latent space of the string is relatively well clustered into four groups, the MIDI pitch 2D latent space is harder to interpret.

We can make the hypothesis that the reduction of the dimensionality has impacted the clusterization of the samples.



(a) Classification by loudness (forte, mezzo forte and piano)

(b) Classification by string played

(c) Classification by MIDI pitch

FIGURE 10 – Observation and classification in the latent space, reduced from 64 to 2 dimensions using Principal component analysis or t-distributed Stochastic Neighbor Embedding

Conclusion

The team obtained a VAE capable of reconstructing interesting envelopes of pitch and loudness. This trained model can also be used to generate brand new violin wav samples from the latent space, which can be listened to on the github.io page. [3]

Another interesting approach would have been to train a VAE with a conditional decoder. By doing so, we might be able to control the pitch and loudness of the generated sound. Indeed, In the latent space, we can impose the value of the fundamental frequency and velocity to generate specific samples. We could make a concatenation of the different generated pitches and loudness es, send it to the DDSP, and listen to a "melody".

In audio application, it is important to consider the aspect of diet programs embedded in hardwares such as a synthesizers. Indeed, it would be natural to one day imagine a trained model inside an electronic hardware that would offer the artist the possibility to create new samples. However, deep models like VAEs are usually too heavy for such applications. Thus, the embedded processor may need more computing power to allow real time control. The lottery hypothesis [8] suggests that those deep models are overparameterized, and a sub-network exists that could give similar results.

3. `https://jeremybboy.github.io/DDSP/`

Références

[1] Diemo Schwarz. Concatenative sound synthesis : The early years. *Journal of New Music Research*, 35(1) :3–22, March 2006.

[2] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP : Differentiable Digital Signal Processing. *arXiv :2001.04643 [cs, eess, stat]*, January 2020. arXiv : 2001.04643.

[3] Rodrigo Castellon, Chris Donahue, and Percy Liang. Towards realistic MIDI instrument synthesizers. page 8.

[4] Nicolas Jonason, Bob L T Sturm, and Carl Thome. The control-synthesis approach for making expressive and controllable neural music synthesizers. page 9, 2020.

[5] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. CREPE : A Convolutional Representation for Pitch Estimation. *arXiv :1802.06182 [cs, eess, stat]*, February 2018. arXiv : 1802.06182.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*, 2015.

[7] Antoine Caillon, Adrien Bitton, Brice Gatinet, and Philippe Esling. Timbre latent space : exploration and creative aspects. *arXiv preprint arXiv :2008.01370*, 2020.

[8] Philippe Esling, Ninon Devis, Adrien Bitton, Antoine Caillon, Constance Douwes, et al. Diet deep generative audio models with structured lottery. *arXiv preprint arXiv :2007.16170*, 2020.