Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Latent representations for traditional music analysis and generation

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Marco Amerotti

Correlatore:
Dott. Luca Casini

Sessione I
Anno Accademico 2021/2022

"μίμησιν μὲν γὰρ δὴ καὶ ἀπεικασίαν
τὰ παρὰ πάντων ἡμῶν ῥηθέντα χρεών
που γενέσθαι"
— Plato, *Critias 107b*

Tutto ciò che ognuno di noi potrà
mai dire sarà in un qualche modo
imitazione e rappresentazione.

All that is said by any of us can only
be imitation and representation.

Contents

List of Figures

List of Tables

Sommario

In questo lavoro analizziamo l'uso di modelli generativi basati su reti neurali artificiali applicati alla musica tradizionale irlandese.
Come il linguaggio naturale, il linguaggio musicale segue regole e pattern precisi: si potrebbe, ad esempio, pensare di creare un modello iterativo basato su tali regole; la difficoltà sta nel formalizzarle per concetti di alto livello, come "una frase di senso compiuto" oppure "una buona melodia". Avendo a disposizione un ampio dataset di melodie folk irlandesi, abbiamo deciso di utilizzare il *machine learning*, e in particolare il *deep learning*, che è lo stato dell'arte in ambito generativo per il linguaggio naturale e la musica.
In particolare, siamo interessati a un modello che possa sia generare, sia analizzare materiale musicale: abbiamo dunque deciso di costruire i nostri modelli a partire dai *Variational Autoencoders* [9], modelli generativi *deep* che apprendono una rappresentazione interna dei dati sui quali sono allenati, rappresentazione che noi intendiamo sfruttare per l'analisi di materiale musicale.
Apportando alcune modifiche a un modello esistente per il linguaggio naturale [1], abbiamo allenato due famiglie di modelli:

- folktune-VAE: un modello allenato sul dataset originale di melodie irlandesi;
- folkbar-VAE: un modello allenato su un dataset di singole battute appartenenti a melodie del dataset originale, generato da quest'ultimo.

Per ognuno di questi, abbiamo considerato 5 variazioni di dimensioni crescenti. Abbiamo quindi utilizzato il primo tipo per generare nuove melodie e analizzare la rappresentazione interna delle melodie del dataset; il secondo per analizzare la struttura di una singola melodia come successione delle sue battute, anche qui sfruttando la rappresentazione interna di queste. In particolare, abbiamo condotto i seguenti esperimenti:

- abbiamo analizzato la rappresentazione interna di melodie e battute in termini di tonalità e metro, considerando, nel caso di singole battute, una stima della loro funzione armonica;
- abbiamo analizzato differenze e somiglianze tra la rappresentazione di melodie del

nostro dataset e melodie in uno stile specifico ("*reels*") provenienti da un altro dataset;

- abbiamo calcolato la distanza tra le rappresentazioni di singole battute di una melodia, cercando di ottenere informazioni sulla somiglianza di tali battute e sulla struttura generale del passo scelto.

Per concludere, presentiamo alcuni esempi generativi di melodie complete.
Nonostante i risultati generativi si siano rivelati interessanti e musicalmente coerenti, non siamo riusciti a trovare collegamenti significativi tra la rappresentazione interna imparata dai nostri modelli e concetti di teoria musicale.
In generale, abbiamo trovato correlazioni molto deboli tra melodie in una data tonalità o in un dato metro, che si sono rivelate ancora meno forti analizzando singole battute. Abbiamo potuto identificare una somiglianza tra le rappresentazioni dei *reels*, ma non abbiamo riscontrato significativa separazione da altri tipi di melodie. L'analisi delle rappresentazioni di melodie come sequenza di battute ha mostrato che le rappresentazioni imparate sono troppo generiche per poter evidenziare concetti propriamente musicali nei dati analizzati.
Inoltre, abbiamo riscontrato un peggioramento della qualità di queste rappresentazioni all'aumentare delle dimensioni dei modelli allenati.
Possiamo concludere che la rappresentazione interna analizzata non è interessante da un punto di vista musicale e non può quindi essere analizzata con le finalità che ci eravamo preposti, nonostante ci siano buoni risultati da un punto di vista generativo. In particolare, i modelli che presentano una migliore rappresentazione interna forniscono risultati generativi peggiori.
In definitiva, crediamo che l'attenzione a l'uno o l'altro aspetto sia inerentemente dipendente dagli scopi e dalle intenzioni dell'utente finale di questi sistemi.

Abstract

We create and study a generative model for Irish traditional music based on Variational Autoencoders and analyze the learned latent space trying to find musically significant correlations in the latent codes' distributions in order to perform musical analysis on data. We train two kinds of models: one trained on a dataset of Irish folk melodies, one trained on bars extrapolated from the melodies dataset, each one in five variations of increasing size. We conduct the following experiments: we inspect the latent space of tunes and bars in relation to key, time signature, and estimated harmonic function of bars; we search for links between tunes in a particular style (*i.e.* "reels") and their positioning in latent space relative to other tunes; we compute distances between embedded bars in a tune to gain insight into the model's understanding of the similarity between bars. Finally, we show and evaluate generative examples. We find that the learned latent space does not explicitly encode musical information and is thus unusable for musical analysis of data, while generative results are generally good and not strictly dependent on the musical coherence of the model's internal representation.

Introduction

In the following work, we study the use of generative models based on artificial neural networks applied to Irish folk music.
Similar to natural language, music follows specific rules and patterns: for example, one could think to create an iterative rule-based system, but such an approach quickly becomes impractical because of the difficulty of specifying such rules for high-level concepts (*e.g.* "a meaningful sentence", "a good melody" etc). Having a large dataset of Irish folk music at our disposal, we chose to use machine learning, which does not require an explicit formulation of rules but rather infers them from data. In particular, we use deep learning, which is the state of the art for both natural language and musical generation.
Inspired by "The Ai Music Generation Challenge 2022"[1], we were interested in creating a model capable of both generating and analyzing musical material: as such, we chose Variational Autoencoders as a starting point, because they are generative models and they learn an internal representation of the training data that we aim to use for musical analysis.
We expect to find meaningful links between the model's internal representation and the musical properties of the data while being able to generate plausible and interesting Irish folk music.
Music has often entered the realm of computer science [4]: from Hiller & Isaacson's Illiac Suite in 1958 [6] to the most recent deep models, algorithmic and AI-driven musical composition already has a long history. A variety of approaches have been proposed, from expert systems to Markov Chains; two of the most effective results come from Google's Magenta team, which created MusicVAE [11] and Music Transformer [8]. These models are really good at representation and generation of musical excerpts and have been a source of inspiration for numerous works, including the following. The code, data and checkpoints we used are publicily publicly available on GitHub[2].

[1] www.github.com/boblsturm/aimusicgenerationchallenge2022

[2] www.github.com/amerotz/latent-representations-for-traditional-music-analysis-and-generation

Irish traditional music

By Irish traditional music, we mean a collection of tunes and melodies, mostly monophonic, with no specific harmonic accompaniment, native to Ireland and built up in centuries of continuous playing and composition. It is usually performed in "sessions", where several musicians play the same melody at once, at times backed by a rhythm section (for example a pianist or a guitarist, or an accordion).
This paper focuses on this particular genre of music for a few reasons:

- the presence of an extensive, homogeneous, and cleaned dataset of Irish traditional music in ABC notation;
- a previous history of being used in AI composition and analysis;
- its monophonic structure, allowing us to consider only one voice;
- the regularity of tune structures (mostly AB or ABC) and the repeating elements and *clichés* among melodies.

All of this allows us to easily build and train models and simplifies the process of analyzing generated material.
Moreover, each melody in the dataset is stored in ABC Notation Format, which is a textual, human-readable encoding for musical notation; this allows the repurposing of existing text-based models with few modifications.
As stated before, other works have targeted the problem of Irish folk music generation.

folk-rnn

Folk-rnn [12] [5] is a recurrent neural network that generates Irish traditional music and is freely available to the public at `www.folkrnn.org`. It has seen various iterations and it is the model the dataset we used was created for.

Tradformer

Tradformer [2] is another model for musical generation based on transformers. It has been trained on the same dataset and produced very convincing and musical results. It has also been employed in the task of generating Swedish-style folk tunes, yielding interesting results.

Outline

This work is structured as follows:

- Chapter 1: we introduce the fundamental concepts and techniques involved in our analysis;
- Chapter 2: we show the dataset we used and describe the models we trained, along with implementation considerations and training plots;
- Chapter 3: we show experimental results concerning our models' internal representation and generative examples;
- Chapter 4: we draw conclusions on the total results of our experiments.

Chapter 1

Background

We briefly describe the techniques and deep architectures we used.

1.1 RNNs and GRUs

Recurrent Neural Networks (RNNs) are a family of neural networks for processing sequential data. In an RNN, the state at each time step is dependent on the state at the previous one, and weights are shared across time steps. Thus, given a sequence, the output at the final time step is dependent on the whole input sequence. The ability to remember previously seen tokens and subsequences makes RNNs extremely useful in music-related tasks, where being able to recognize and produce both local and global repeated patterns is a crucial and desired skill.
A problem to which standard RNNs are vulnerable is gradient vanishing or explosion during backpropagation: that is, the convergence to zero or total divergence of the long-term gradients, due to the finite precision of the numbers involved. To circumvent this issue, Long Short-Term Memory networks (LSTMs) [7] have been proposed and have proved to be effective in a variety of applications.
In our implementation, we make use of a particular type of RNN called Gated Recurring Units (GRUs) [3], which is a variation of the LSTM pattern. GRUs are LSTMs capable of "forgetting" and resetting at appropriate times, for example in continuous input streams without explicitly marked sequence ends.
The recurring units are required in order to compress the input training sequence into a single vector for the next layers.
For each element in the input sequence, each GRU layer computes the following function:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr})$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz})$$

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn}))$$

$$h_t = (1 - z_t) * n_t + z_t * h_{(t-1)}$$

where

- h_t is the hidden state at time t;
- x_t is the input at time t;
- $h_{(t-1)}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0;
- r_t, z_t, and n_t are the reset, update, and new gates, respectively;
- σ is the sigmoid function;
- $*$ is the Hadamard product.

1.2 Autoencoders

Autoencoders are deep neural networks that operate on this principle: they try to copy their input to their output by passing it through a bottleneck layer, smaller than the input and output dimensions. While this may seem of doubtful usefulness, the bottleneck layer is really what we are interested in: being forced to reconstruct a target from a reduced version of it, the network often learns useful properties of the data.
Moreover, if we consider the output of the bottleneck layer as a point in some n-dimensional vector space, we find out that similar points tend to be mapped closed together. This vector space is often called "latent space".
The network can then be conceived as made of two parts:

1. an encoder, which processes an input sequence into a point in the learned latent space;
2. a decoder, which tries to map a latent point to its original data point.

This allows interesting operations on data, such as interpolation or other arithmetical operations between latent points, giving us new points which can be decoded back to what the network thinks their precursor should be. Thus we can generate new data or "steer" existing data towards particular regions of the latent space that encode particular features, with a technique known as "attribute vector arithmetic" [11].

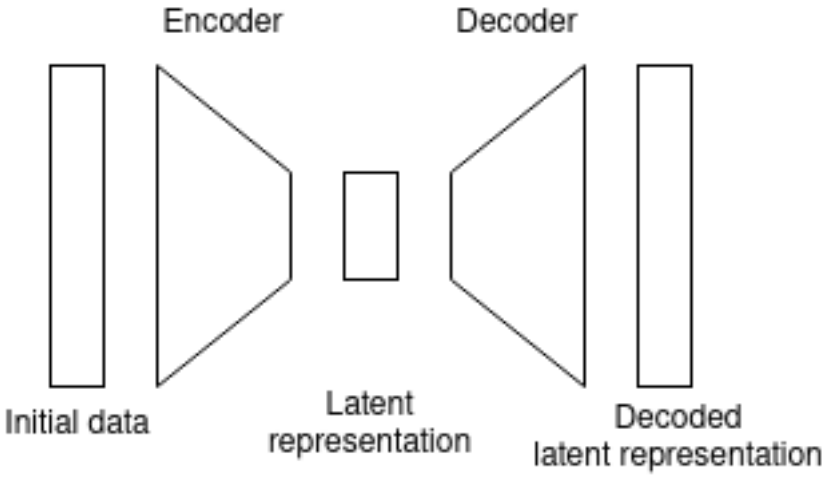


Figure 1.1: The autoencoder architecture.

All of the above assumes the existence of a learned latent space of an appropriate dimension for the complexity of the data, in which latent points are fairly well distributed and there are no large "empty" gaps. While we can always enlarge or reduce the bottleneck layer, as long as we keep it smaller than the original dimension, the irregular point distribution is harder to address and very common in practice. This may result in sampling latent points which are too far from true data points to be decoded meaningfully.

1.3 Variational Autoencoders (VAEs)

To solve the issue mentioned above, Variational Autoencoders (VAEs) [9] have been developed. VAEs share the same architectural features as standard Autoencoders, but encode data points into a distribution of points in latent space, rather than a single point.
The encoder learns two vectors that represent the mean and variance of a normal distribution in latent space.
The decoder samples a random point from such distribution that will then be used to reconstruct the original input. Since the sampling operation is not differentiable, we used the so-called "reparametrization trick": a sample is drawn from a standard Gaussian distribution and is then combined with the aforementioned mean and variance. During backpropagation, mean and variance vectors are adjusted accordingly, ignoring the stochastic operation.
This has the effect of generating a space of distributions over data points that can overlap and create a more significant latent space. Points that would have been isolated with a vanilla Autoencoder approach may now appear between more distributions and would

be decoded in a more meaningful way.
A special kind of loss function is used, by the name of Evidence Lower Bound (ELBO) loss.

1.3.1 ELBO loss function

The ELBO loss function allows us to backpropagate considering both the reconstruction error between input and output and the distance between the model's distribution of data and the data's distribution.
In our approach, we employed the following ELBO loss function:

$$\mathcal{L}(\theta; x) = -\mathrm{KL}(q_\theta(\overline{z}|x)||p(\overline{z})) + \mathbb{E}_{q_\theta(\overline{z}|x)}[\log p_\theta(x|\overline{z})]$$

where

- x is an input;
- $\overline{z}$ is the model's learned code for x;
- KL is the Kullback-Leibler divergence, which is a way of representing the difference between two distributions [9];
- $q_\theta(\overline{z}|x)$ is the model's approximate posterior distribution over $\overline{z}$;
- $p(\overline{z})$ is the model's prior distribution over $\overline{z}$; in this case, a standard Gaussian;
- $p_\theta(x|\overline{z})$ is the likelihood of x conditioned on $\overline{z}$;
- $\mathbb{E}_{q_\theta(\overline{z}|x)}[\log p_\theta(x|\overline{z})]$ is the expected value of $p_\theta(x|\overline{z})$ with respect to $q_\theta(\overline{z}|x)$.

1.3.2 β-VAEs

The two-term structure of the ELBO loss allows for tweaks in order to prioritize the reconstruction or distribution error. This is achieved through a weighted Kullback-Leibler Divergence with a parameter β, whence the name. The new loss function follows this equation:

$$\mathcal{L}(\theta; x) = \beta \cdot -\mathrm{KL}(q_\theta(\overline{z}|x)||p(\overline{z}) + \mathbb{E}_{q_\theta(\overline{z}|x)}[\log p_\theta(x|\overline{z})]$$

where β is the weight we defined.

1.3.3 Annealing

The weight β mentioned above can be fixed during the whole training process, or gradually increased at each step, with the idea of emphasizing reconstruction accuracy in the first epochs and focusing later on the minimization of the Kullback-Leibler Divergence. Such a process is known as "annealing" and has been used in training our models.
In particular, we initialize the weight of the KL-divergence at 0 and bring it to 1 following a sigmoid contour during the first epochs of training. This contour is a function of the current time step, defined as follows:

$$\frac{1}{1 + e^{-k \cdot (t - x0)}}$$

where

- t is the current time step;
- k is a parameter that encodes the "steepness" of the contour;
- x_0 is the inflection point (*i.e.* time step) of the sigmoid.

During training, we used $k = 0.0025$ and $x_0 = 2500$.

1.4 Uniform Manifold Approximation and Projection for Dimension Reduction

Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [10] is a technique for projecting high-dimensional spaces into lower-dimensional ones. The goal is to obtain a projection of the original space which is its closest lower-dimensional topological equivalent. We employ this technique to visualize the high-dimensional learned latent space in two dimensions.

Chapter 2

Models and data

We show the dataset we used and how we processed it, followed by the types of models we have trained and their training plots.

2.1 Main dataset

As mentioned before, the training data we used is a collection of Irish folk tunes in ABC notation format, available with folk-rnn's release on GitHub[1]. These are already tokenized and arranged in a text file. There are multiple versions of this dataset, which include tune titles, omit repetitions, transpose to other keys, etc. We opted for the second iteration of this dataset, which contains more than 20,000 tunes transposed to C without titles, with meter and C mode indication.
Transposing tunes to C incentivizes the model to learn structures and patterns tied to scale degrees more than to individual notes.
This is what a standard tune looks like in the source dataset:

```
M:6/8
K:Cmaj
|: E 2 G G E G | A c A G 2 F | E 2 D D C D | E 2 C C2 D |
E 2 G G E G | A /2 B /2 c A G 2 G | A B c d c B |1 c 2 G A 2 G :|
|2 c 3 c 2 d |: e 3 e d c | f 2 d d c d | e g e e d c |
g ^f g a g =f | e 3 e d c | f 2 d d B G |1 A A /2 B /2 c d B G |
c 3 c 2 d :| |2 A B c d c B | c 2 G A 2 G |
```

and this is the musical notation associated with it.

[1] www.github.com/IraKorshunova/folk-rnn

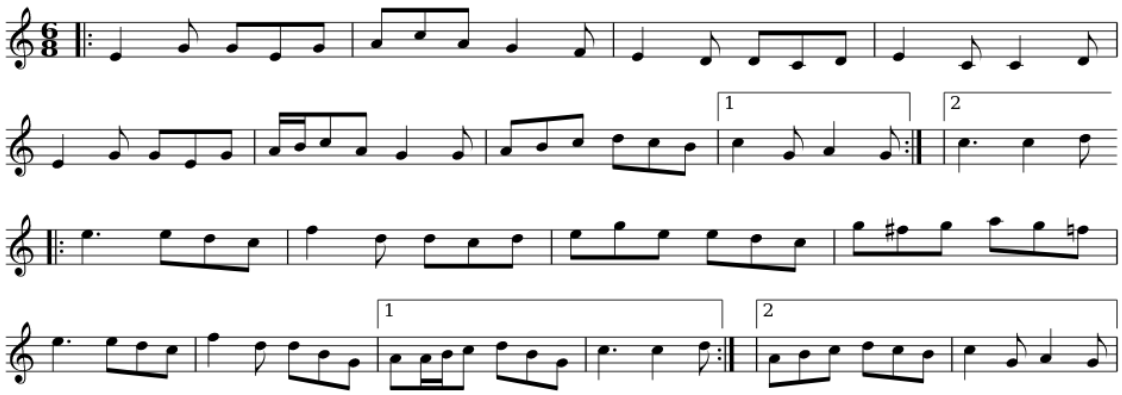


Figure 2.1: The musical notation associated with the ABC notation above.

We performed a minimal amount of cleaning removing tunes that exceeded a length of 256 tokens and that presented rare tokens (less than 100 total occurrences); we discarded less than 10% of the total tunes.
The final dataset is structured as follows:

Key signature	Cmaj	Cmix	Cmin	Cdor	Total
Number of tunes	14,260	1,432	2,876	2,702	21,270

Table 2.1: Number of tunes in the dataset by key signature.

Time signature	2/4	3/4	4/4	6/8	9/8	12/8	3/2	Total
Number of tunes	1,637	1,576	11,044	5,562	801	475	175	21,270

Table 2.2: Number of tunes in the dataset by time signature.

2.2 Models

We have trained a variety of models of multiple sizes and experimented with other configuration parameters, such as KL-Divergence weight and batch size.
We have trained two kinds of models: one on full tunes, as in the source dataset, and one on individual bars, generated from the same data by splitting each tune into a collection of bars. We trained this last model in order to analyze the paths created by sequences of points in latent space corresponding to bars belonging to a tune.

All models stem from a PyTorch implementation of a natural language processing model called Sentence-VAE [1], which we have tweaked to meet our goals.
SentenceVAE is an RNN-based variational autoencoder generative model that encodes latent representations of entire sentences. This allows encoding properties of the entirety of the sentence, such as style, topic, and other high-level syntactic features. It consists of a two-layer encoder RNN and a three-layer decoder RNN which acts on natural language sentences. Each word is represented using a learned dictionary of embedded vectors. It is able to encode sentences and interpolate between latent points, decoding those back to new sentences with a certain degree of meaningfulness.
The textual format of our dataset and the elegance and simplicity of SentenceVAE allowed us to apply it to Irish folk tunes out of the box, with interesting and promising preliminary results.
However, being interested in musical analysis, we deemed necessary to adapt part of its architecture to treat musical data.

2.2.1 folktune-VAE

We first turned our attention to training our first model on the entirety of our tune dataset, using 80/10/10 train/validation/test splits. We will refer to this kind of model as "folktune-VAE" from now on, attaching "latent space size / hidden size" when needed.
Since we know *a priori* the contents of our dataset, and thus the tokens that are present, we substituted the learned word embedding with one-hot encoding. Our dictionary counts 119 tokens, including the model's auxiliary tokens (such as padding, start/end of stream, etc).
Having limited time and computing power at our disposal, we settled on a two-layer encoder GRU and a two-layer decoder GRU, while trying different hidden and latent sizes.

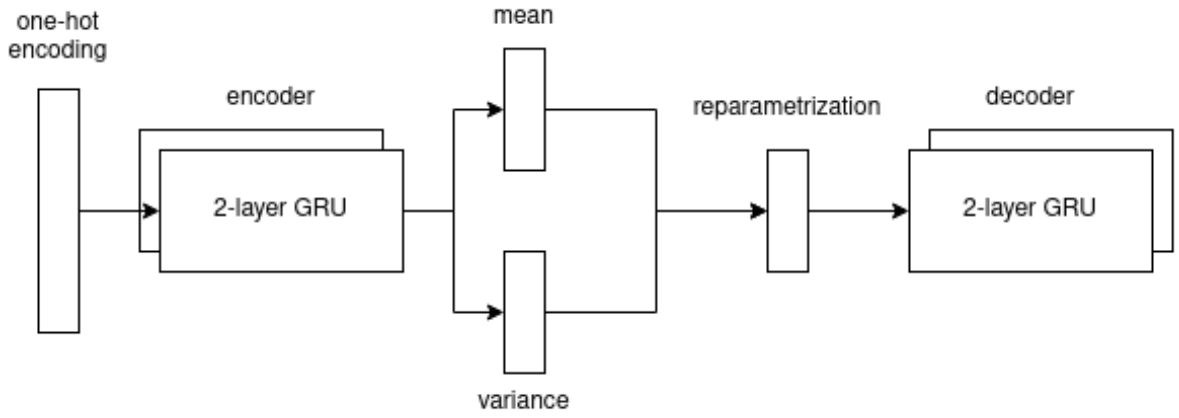


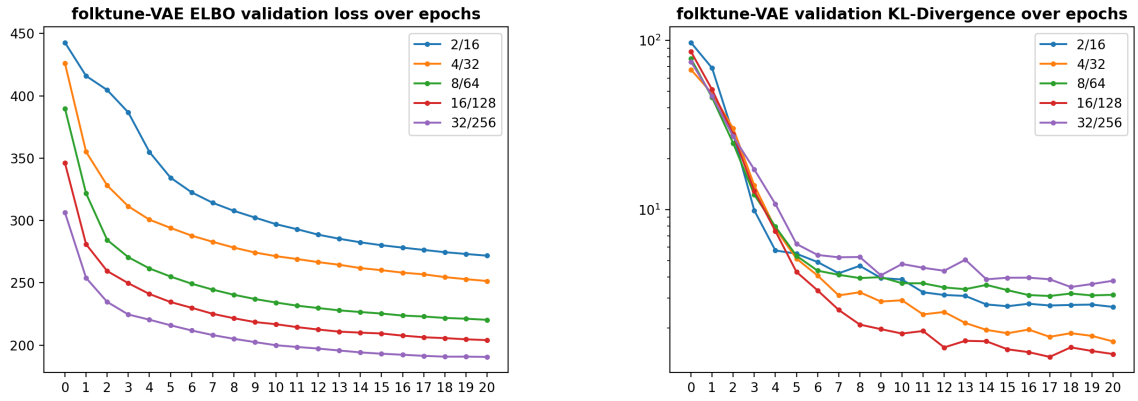
Figure 2.2: The general architecture of folktune-VAE.

The general architecture of folktune-VAE.
To analyze how the model's size affected the latent space and the reconstruction, we trained the following models:

- 2-dimensional latent space, 16-dimensional hidden size;
- 4-dimensional latent space, 32-dimensional hidden size;
- 8-dimensional latent space, 64-dimensional hidden size;
- 16-dimensional latent space, 128-dimensional hidden size;
- 32-dimensional latent space, 256-dimensional hidden size.

This particular choice of values seemed appropriate because of the aforementioned trade-off between computing resources and time. Moreover, it is sensible to start with the smallest model possible and gradually increase its size, recording the changes in performance that occur and how large those are.
We trained each model for 20 epochs, which preliminary experiments showed to be enough for the model to settle, without noteworthy improvements in terms of loss beyond this point.
We also conducted preliminary verifications on what the most useful batch size would be and obtained the least loss models with a value of 32, which we choose to keep for all the models above. We used a learning rate of 0.001 and the Adam optimizer.
We trained each model with annealing of the KL-Divergence.

Training plots



(a) (b)

Figure 2.3: folktune-VAE validation ELBO loss (a) and KL-Divergence (b) for each trained model.

As the figures show, increasing the model's size causes a lower ELBO and reconstruction loss in general.
Analyzing the KL-divergence trend, we see that smaller models, namely folktune-VAE 16/128 and 4/32, seem to outperform larger ones. We will successively inspect if this inferior loss value corresponds to an actual better understanding of the data's distribution by such models. Since the KL-divergence is responsible for the structure of the latent space, we speculate that the aforementioned models will show a more coherent latent space.
We used the last checkpoint for each model since it has the lowest loss value in all cases.

2.2.2 folkbar-VAE

Using the same structure of folktune-VAE, we trained a series of models on individual bars, with the intent of analyzing relationships between bars in a given tune. We obtained our bars dataset by splitting the original tunes into bars and removing key and time signature indications. We processed each original split individually so that there is a correspondence between train, validation, and test data between this model and folktune-VAE. We will use the name "folkbar-VAE" with the usual latent space size and hidden size indications.
Each model was trained for 20 epochs and with the same parameters as before. As we did with complete tunes, we considered only bars not exceeding 32 total tokens, in practice

excluding only a few examples from our generated bars' dataset.
We stored key and time signatures separately with each bar to allow further analysis.
Using measures instead of complete tunes as our training data, we find it sensible to analyze what the harmonic context of a single bar is: in order to do so, we store an estimate of the bar's harmonic function, computed with the following procedure.

Harmonic function estimate of individual bars

By the harmonic function of a bar, we mean the "role" that is played by that bar in the piece based on the implied harmony of its notes.
We can rely on Irish folk music being strongly tonal and thus employing basic and traditional harmonic rules, which in essence are tied to the concept of Tonic, Plagal, and Dominant harmonic functions. Without delving too deep into music theory, we can say that:

- the Tonic function is the harmonic function of "home" chords, that is, the root chord of the key and its minor relative, and does not suggest any movement towards other chords;
- the Dominant function, on the contrary, is the function of the most unstable chords in the key, namely the chords on the fifth and seventh degrees of the scale, which provide a strong sensation of movement towards the Tonic;
- the Plagal (or Subdominant) function is the function of chords that suggest a movement toward both the Tonic and the Dominant chords, that is the second and the fourth degrees of the scale.

The third degree of the scale usually has a more ambiguous role, which is highly dependent on context; as such, it is often employed as a passing chord between "stronger" chords.
Having made said simplifications, we assign to each bar one of the aforementioned functions with a score-based system:

- we initially assign a score of 0 for each harmonic function;
- we iterate through the pitches in the bar and associate them with the chords in which they appear;
- we add one point to each function that appears at least once in the chords' harmonic functions.

We then simply choose the harmonic function with the highest total score. For each pitch, the number of points per function is defined by the following tables:

Chord	Function
I	Tonic
II	Plagal
III	Tonic
IV	Plagal
V	Dominant
VI	Tonic
VII	Dominant

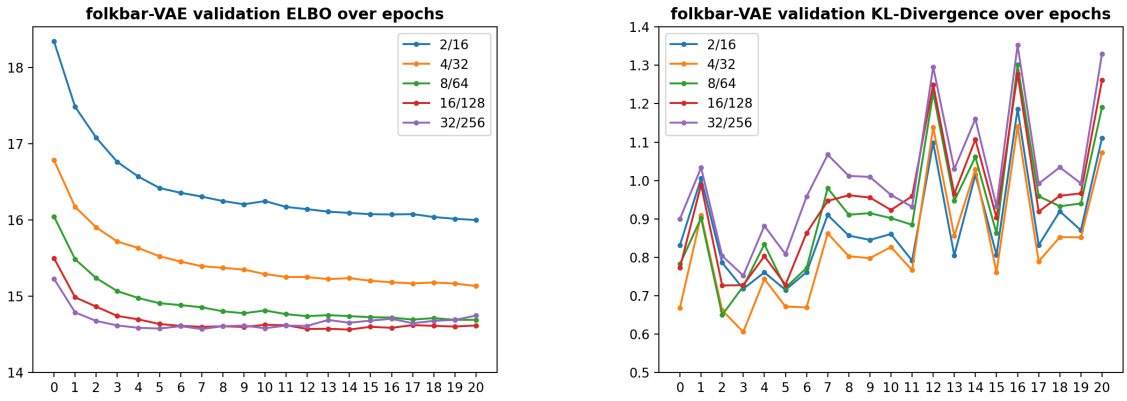
Table 2.3: Harmonic function of chords built on scale degrees.

Pitch	Tonic	Plagal	Dominant	Chords
C	1	1	0	I, IV, VI
D	0	1	1	II, V, VII
E	1	0	0	I, III, VI
F	0	1	1	II, IV, VII
G	1	0	1	I, III, V
A	1	1	0	II, IV, VI
B	1	0	1	III, V, VII

Table 2.4: Score system used to estimate a bar's harmonic function.

We arbitrarily decided to consider the chord on the III degree as a Tonic chord, which appeared to be a more appropriate general function than Dominant or Plagal in this kind of music.

Training plots



(a) (b)

Figure 2.4: folkbar-VAE validation ELBO loss (a) and KL-Divergence (b) for each trained model.

According to the validation ELBO loss, bigger models should perform better and should considerably outperform the smaller ones. We can see that the loss starts to rise towards the end, signaling that less training would have been enough to reach the lowest loss checkpoint. This did not happen when working with complete tunes, which is expected, considering the inevitable inferior complexity of single measures with respect to a complete musical piece. Moreover, our bars dataset counts a lot more single training examples than the tunes dataset, given that usually, each piece contains around 16 bars. The KL-Divergence follows the same trend. In order to meaningfully evaluate the bars' latent space, we chose the best checkpoint for each model.

Chapter 3

Results

In the following section, we analyze a variety of experimental results concerning the structure of folktune-VAE and folkbar-VAE's learned latent space. We conducted the experiments below:

- an analysis of folktune-VAE's latent space by projecting individual tunes;
- an analysis of how tunes belonging to a specific genre are projected relative to other tunes in folktune-VAE's latent space;
- an analysis of folkbar-VAE's latent space by projecting individual bars;
- an analysis of distances between latent bars belonging to a given tune in folkbar-VAE's latent space.

All of the above aim to understand whether folktune-VAE and folkbar-VAE's internal representations are musically meaningful and could thus be used for the purpose of musical analysis.

3.1 Tune projection

We start our analysis by taking a general look at the latent space of tunes and trying to understand its characteristics.
We projected tunes by feeding them into the model, which gives us a latent point for each tune. We then run the UMAP algorithm for dimensionality reduction, obtaining the two-dimensional projection of such points.
The plotted tunes are a subset of the dataset chosen with a stratified selection of 200 tunes for each key and time signature. When there were less than 200 tunes for a certain value, we considered the minimum amount of tunes for each category to be represented equally.

In the exceptional case of folktune-VAE 2/16, which already maps tunes into two-dimensional latent points, we provide the original latent space mapping instead of its UMAP projection, being dimensionality reduction the only reason behind the use of this algorithm.

3.1.1 folktune-VAE 2/16

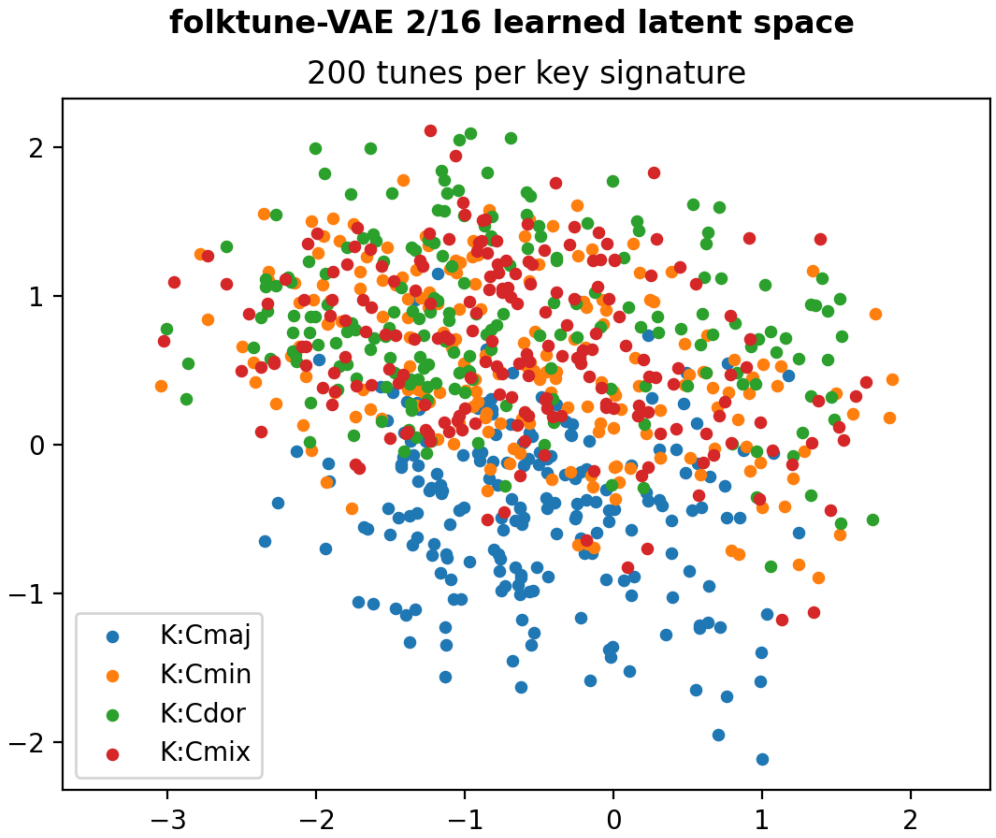


Figure 3.1: folktune-VAE 2/16 learned latent space by key signature.

The model seems to have learned some kind of distinction between different C modes, although not very clear. Given that ABC notation does not explicitly mark accidentals when already implied by the key field, this suggests that there are patterns or figures more common within tunes of a given mode.
C Major seems to be best separated from the other keys: one possible interpretation may be the presence of the B natural as the leading tone, which allows specific melodic

and harmonic patterns that are weaker in the other modes, where we instead have a B flat.
There is no clear distinction between C Dorian, C Minor, and C Myxolidian, and no particular similarity between the first two, which both employ an E flat instead of an E natural. This may be explained by the quality of the median being less influential in melodic patterns: its relationship with the tonic remains consonant and it does not appear in the subdominant and dominant triads, which have a prominent role in this kind of music.

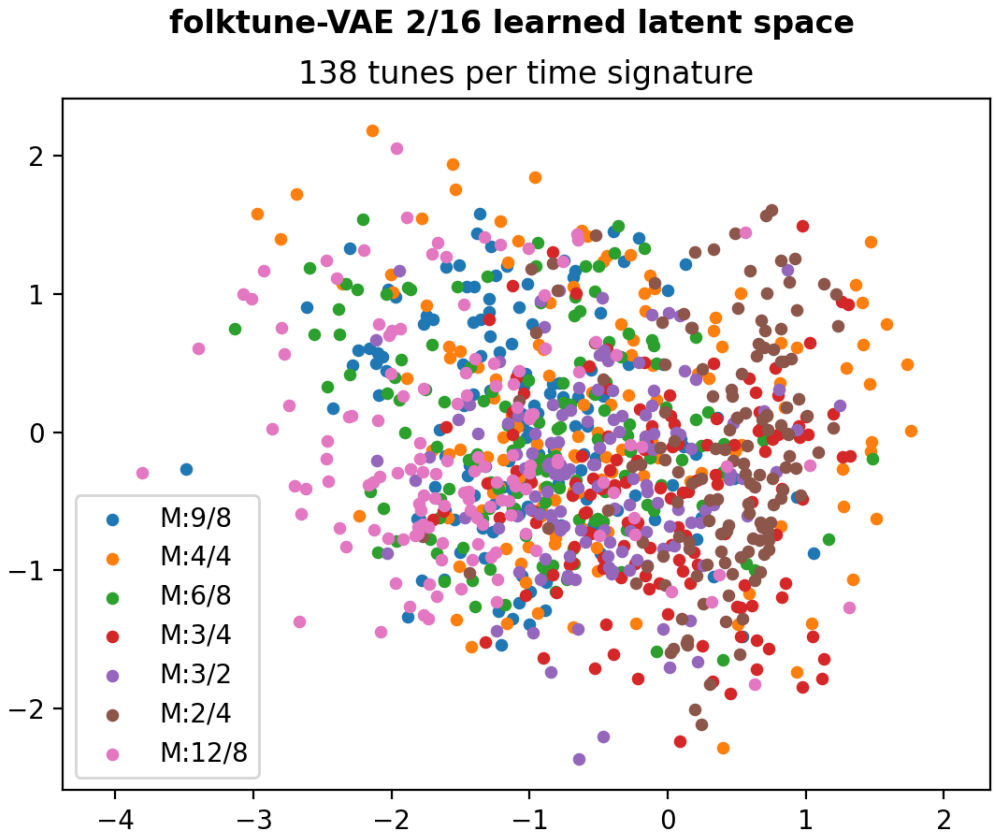
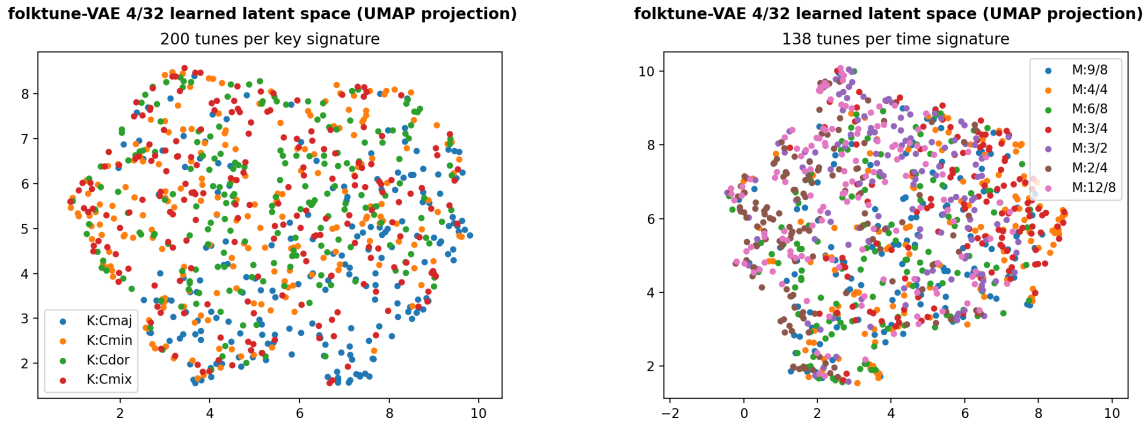


Figure 3.2: folktune-VAE 2/16 learned latent space by time signature.

No particular distinction seems to arise when considering time signatures. Certain areas are more populated by tunes in specific time signatures than others, but there is a considerable amount of overlap.
The only exception is the cluster of tunes in $\frac{2}{4}$: we speculate that this is due to the general inferior tokenwise length of these tunes, which accommodate only two beats per bar.

3.1.2 folktune-VAE 4/32

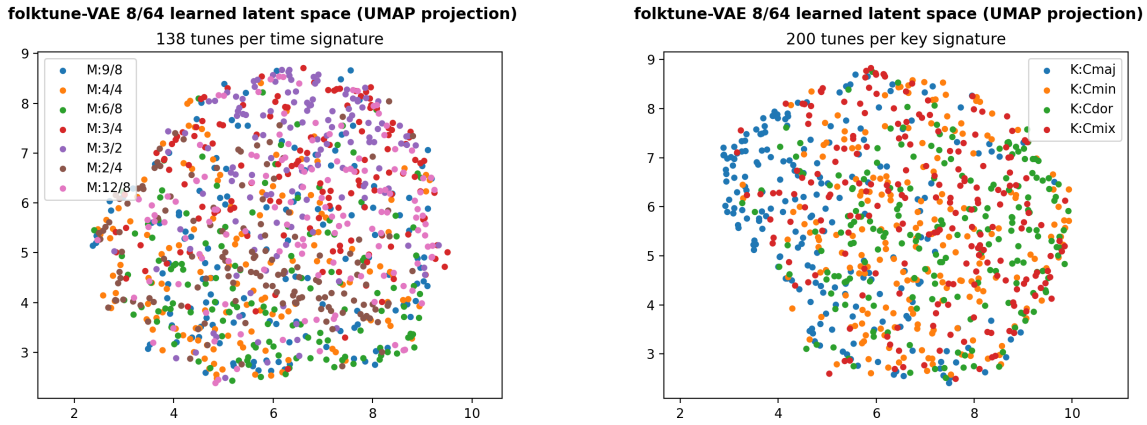


(a) (b)

Figure 3.3: folktune-VAE 4/32 learned latent space by key (a) and time (b) signature.

There is a lot of overlap between point sets, which leads to no interesting interpretation of the latent space.
There seems to be a relative abundance of $\frac{2}{4}$ tunes on the left side and of $\frac{3}{4}$ tunes on the right side, while other time signatures are equally spread throughout.
We exclude tune length and binary/ternary divisions being a significant factor, given the equal distribution of other time signatures in the plot, but we cannot trace this behavior to any significant musical feature.

3.1.3 folktune-VAE 8/64

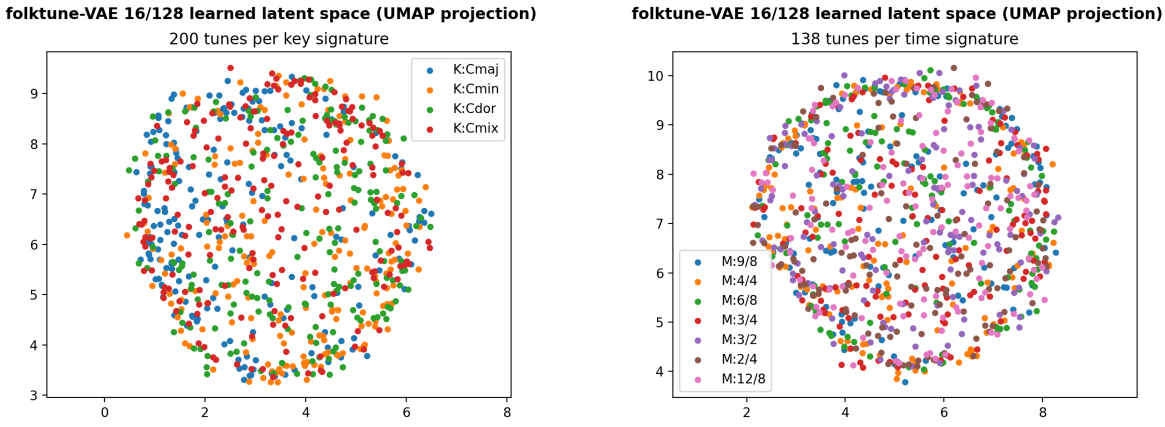


(a) (b)

Figure 3.4: folktune-VAE 8/64 learned latent space by key (a) and time (b) signature.

No new interesting musical considerations arise from this model's results.

3.1.4 folktune-VAE 16/128

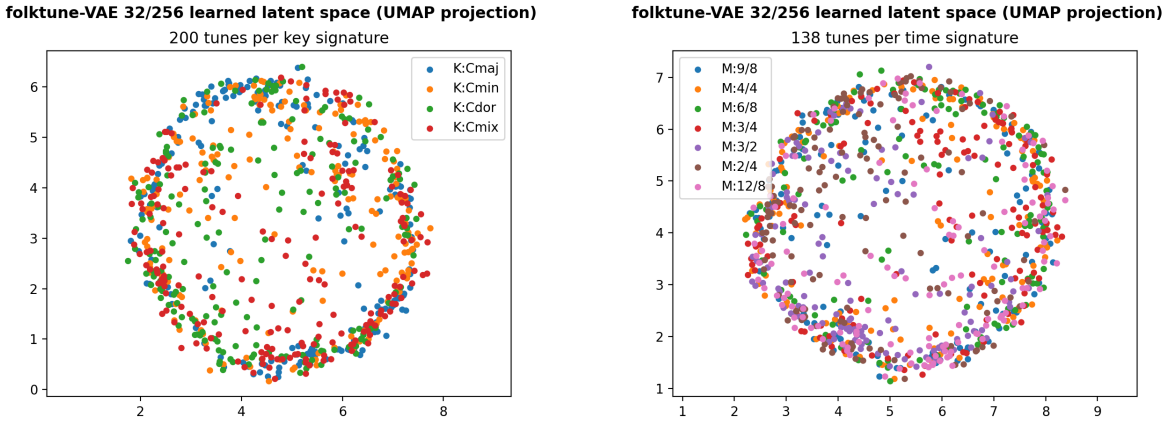


(a) (b)

Figure 3.5: folktune-VAE 16/128 learned latent space by key (a) and time (b) signature.

There seems to be a tendency for tunes to accumulate towards the outer bounds of the plotted region, regardless of time or key signature. We speculate it could be an artifact caused by the UMAP processing step.

3.1.5 folktune-VAE 32/256



(a) (b)

Figure 3.6: folktune-VAE 32/256 learned latent space by key (a) and time (b) signature.

The previously noted tendency seems to be accentuated by this model's projection. We can see groups of tunes in $\frac{12}{8}$ and $\frac{3}{2}$ clustered together, although not clearly separated.

3.1.6 Conclusions

These results are more or less in line with what we predicted when examining the KL-Divergence of the trained models: in fact, bigger models do not seem able to show us any interesting musical properties of the data, while we could distinguish different groups of similar points in folktune-VAE 2/16 and 4/32.
The model that performs best should be 8/64, which could not give us any useful musical information instead. It is however evident that its points are fairly well distributed and that there are no particular clusters or gaps, which could account for the low KL-divergence value, even though this does not seem to reveal anything in terms of key or time signature distribution. It is possible that a per-tune analysis could give us more insight into this particular latent space, but such an analysis is beyond the scope of this work.

3.2 Distance-based tune type recognition

With the goal of recognizing different tune types, we speculate on the idea of using the distance between latent encoded tunes to highlight belonging to a specific musical genre. We turned our attention to a particular kind of Irish tune called "reel". A reel is a tune composed in $\frac{4}{4}$, usually of binary structure, played with even beats.
Relying on this metrical feature as a possible discriminator and on the presence of an Irish reels dataset in ABC format (not included in our training data), we investigate whether the learned space of our models actually encodes high-level musical features, such as tune type.
We cleaned and formatted the reels dataset in order for it to be compatible with our already trained models. The final reel collection counts 348 tunes. By definition, each reel is in $\frac{4}{4}$ time signature; key signatures are distributed as follows:

	Cmaj	Cmix	Cmin	Cdor	Total
Number of reels	256	15	58	18	348

Table 3.1: Number of reels by key signature.

We then projected each reel into latent space and calculated the centroid of this collection of points as follows:

$$C = \frac{x_0 + ... + x_{n-1}}{n}$$

where x_i is a latent point corresponding to a reel.
We then calculated the Euclidean distance between each reel and the centroid C.
Similarly, we repeated the process with a stratified selection of pieces, with a number of tunes equal to the number of reels for each category. When a certain category counted fewer tunes than there are reels, we considered the maximum number of tunes that could represent each class equally. We then encoded them as latent points and computed their Euclidean distances from the centroid.
We plotted both the latent space projection of all encoded points and the distribution of euclidean distance values for each category compared with the reels' distribution.
We provide plots for each trained folktune-VAE model. As before, points encoded by folktune-VAE 2/16 were not subject to UMAP dimensionality reduction.

3.2.1 folktune-VAE 2/16

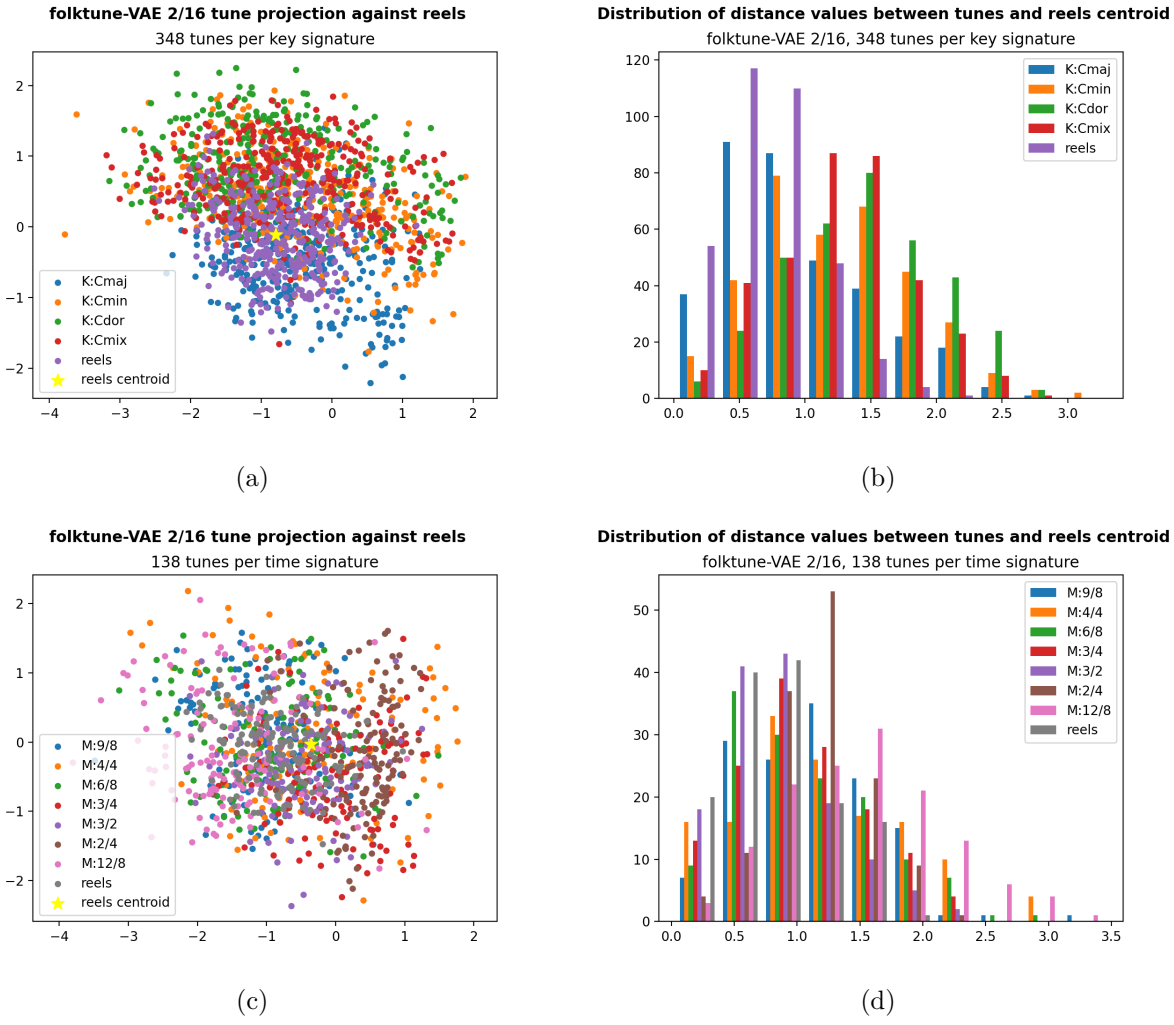


Figure 3.7: folktune-VAE 2/16 learned latent space with reels and distance values from the centroid by key (a & b) and time (c & d) signature.

Tunes in C Major seem to be linked to reels more than tunes in other C modes, which is traceable to C Major being the most common key signature among reels. Reels appear to be fairly evenly distributed between time signatures: we expected the $\frac{4}{4}$ distance distribution to follow the reels more distinctively than others instead. The large spike corresponding to tunes in $\frac{2}{4}$ highlights the cluster on the right.
The centroid lies almost exactly in the center of the whole projection.

3.2.2 folktune-VAE 4/32

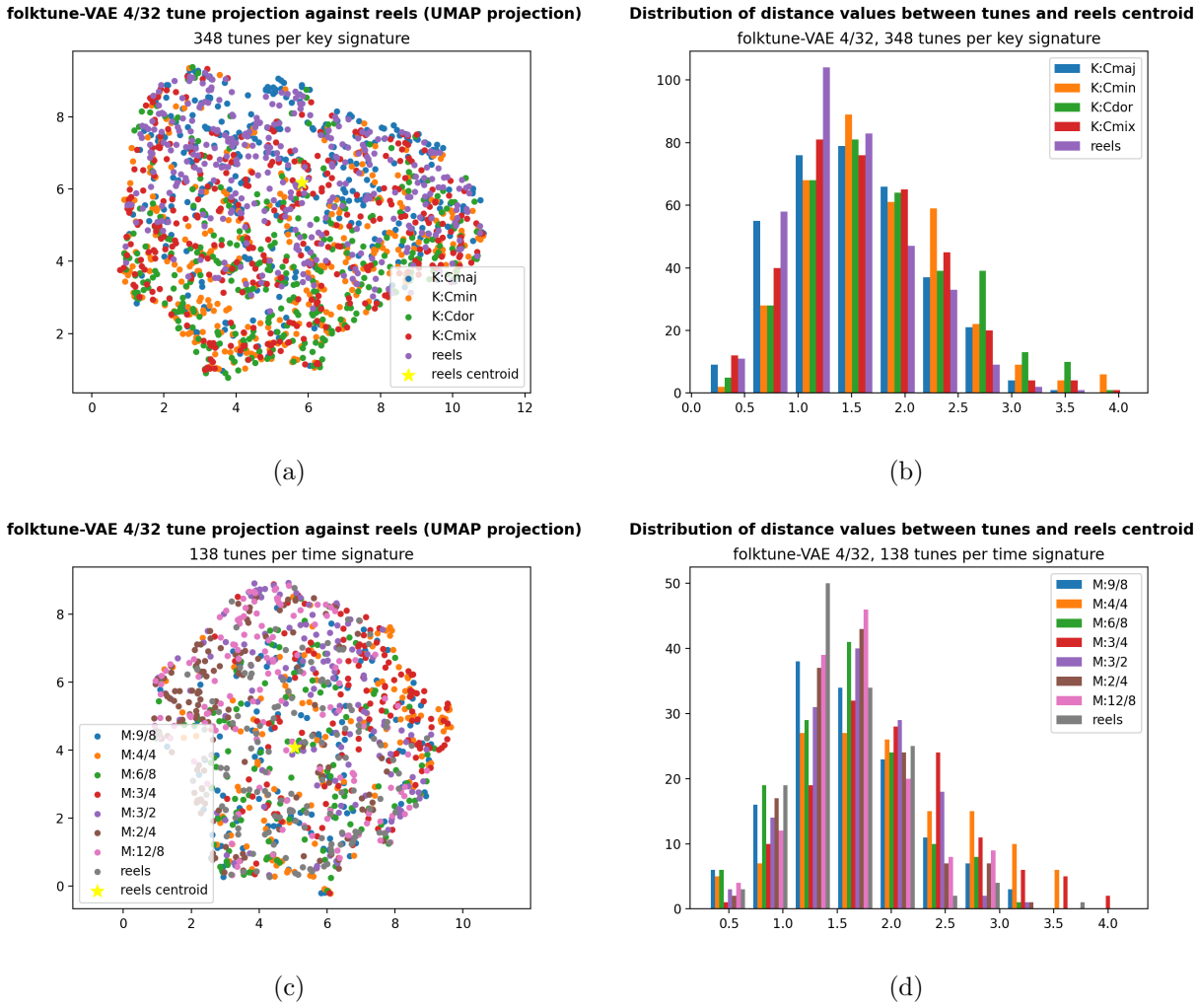


Figure 3.8: folktune-VAE 4/32 learned latent space with reels and distance values from the centroid by key (a & b) and time (c & d) signature.

There does not seem to be any correlation between reels and key signature, although reels tend to be encoded in the upper half of the point set, together with C Major tunes. No significant time signature correlation is noted.

3.2.3 folktune-VAE 8/64

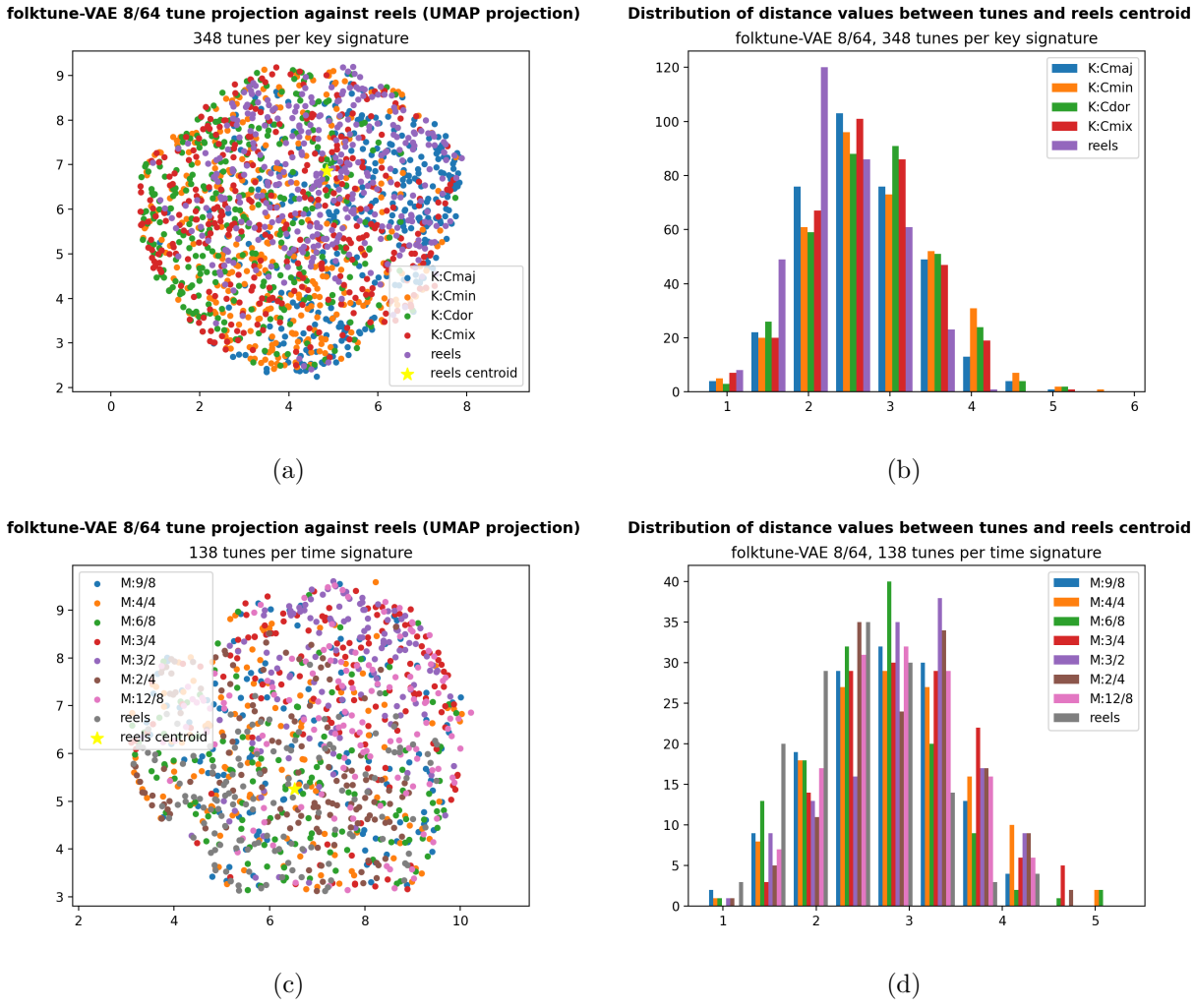


Figure 3.9: folktune-VAE 8/64 learned latent space with reels and distance values from the centroid by key (a & b) and time (c & d) signature.

The considerable amount of overlap does not highlight any musical feature. Reels tend to occupy a somewhat limited region of the plotted space, which is reflected by the centroid being slightly offset from the center.

3.2.4 folktune-VAE 16/128

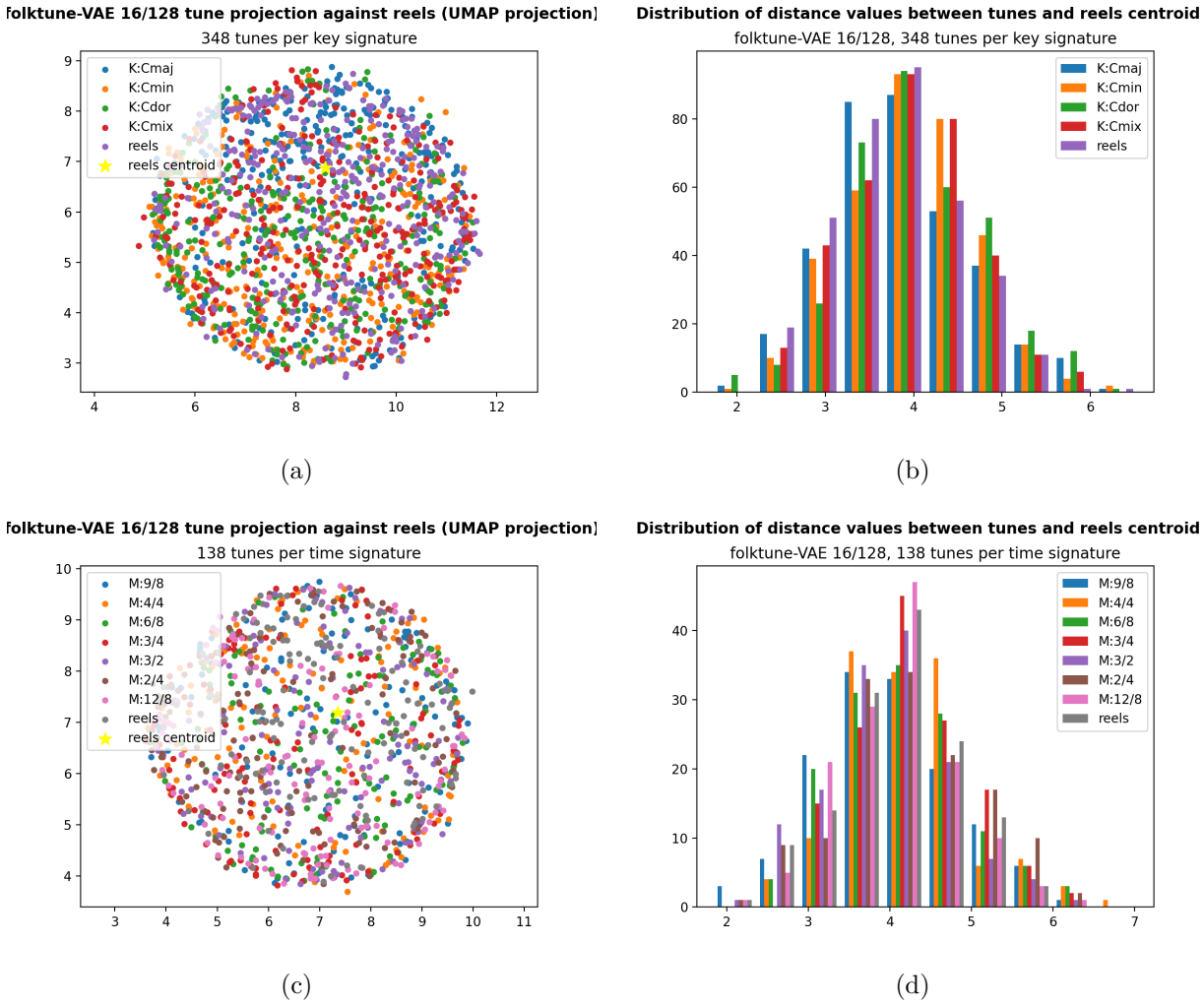


Figure 3.10: folktune-VAE 16/128 learned latent space with reels and distance values from the centroid by key (a & b) and time (c & d) signature.

Again, the considerable amount of overlap does not highlight any musical feature.

3.2.5 folktune-VAE 32/256

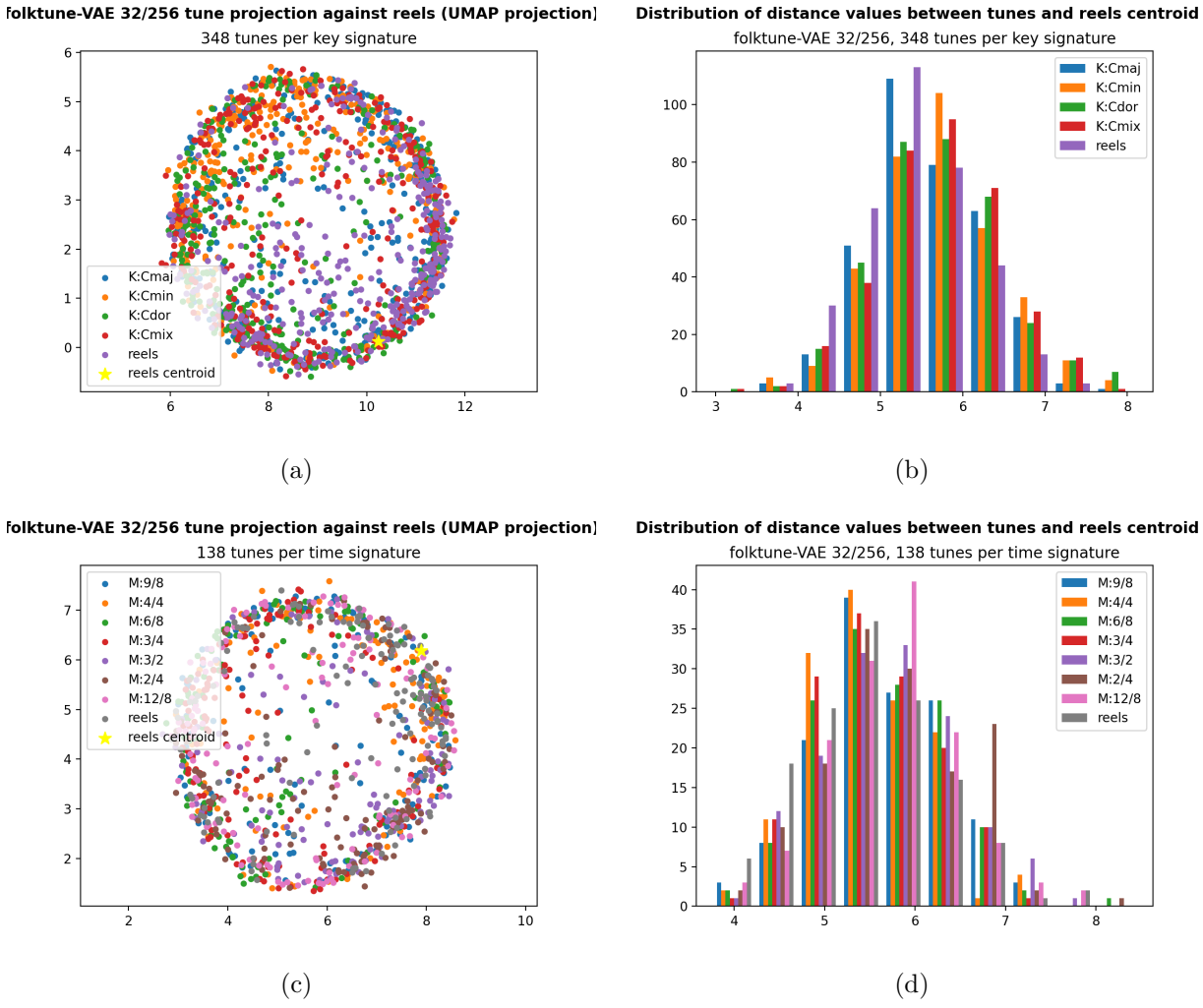


Figure 3.11: folktune-VAE 32/256 learned latent space with reels and distance values from the centroid by key (a & b) and time (c & d) signature.

We encounter the same alleged UMAP artifacts as before. Reels are grouped together in both projections.
In terms of key signature, there is a weak correlation between C Major tunes and reels. The time signature distance histogram does not show interesting tendencies apart from the $\frac{2}{4}$ and $\frac{12}{8}$ spike. An inspection of the latent projection shows that reels and tunes in $\frac{4}{4}$ loosely accumulate in the same region.

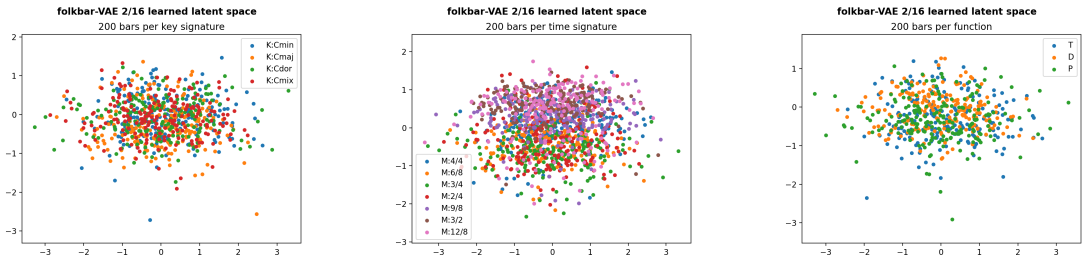
3.2.6 Conclusions

We see that in almost all projections, reels have been mapped to points close to each other. This is an interesting result, as it suggests that there are some global properties of reels as a musical form that our model was able to recognize, although not very clearly. The occurrence of other tunes in $\frac{4}{4}$ as fairly distant points from reels allows us to say that our model is looking at something more than simply meter; at the same time, this tells us that there is no strong association between $\frac{4}{4}$ meter and reels when there should be a major and obvious link, given that $\frac{4}{4}$ time is a distinctive feature of such tune type. Almost all models seem to have found some kind of relationship between $\frac{2}{4}$ tunes, which are often grouped together.

3.3 Bar projection

Using the same approach as before, we now inspect the latent space generated by the folkbar-VAE models, which encode individual bars as latent points. Along with the usual key and time signature projections, we provide a novel stratified selection of bars based on their estimated harmonic function. We used the best checkpoint for each model.

3.3.1 folkbar-VAE 2/16

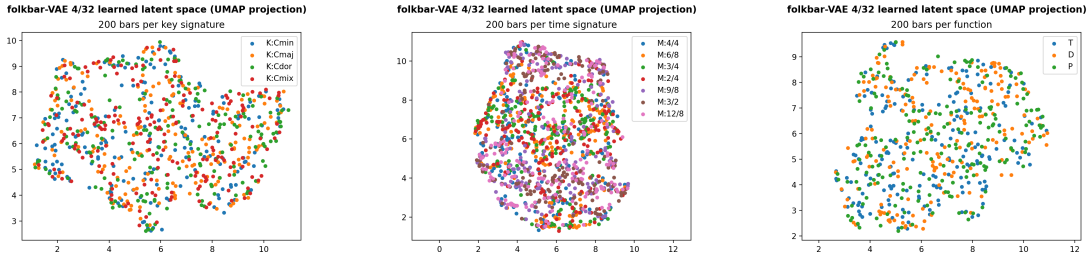


(a) (b) (c)

Figure 3.12: folkbar-VAE 2/16 learned latent space by key (a) and time (b) signature and harmonic function (c).

The time signature projection only shows some signs of separation between bars of different meters, specifically grouping together bars in "long" meters, such as $\frac{12}{8}$, $\frac{9}{8}$ and $\frac{3}{2}$. We speculate that this is due to the greater length of those bars with respect to the other time signatures.

3.3.2 folkbar-VAE 4/32

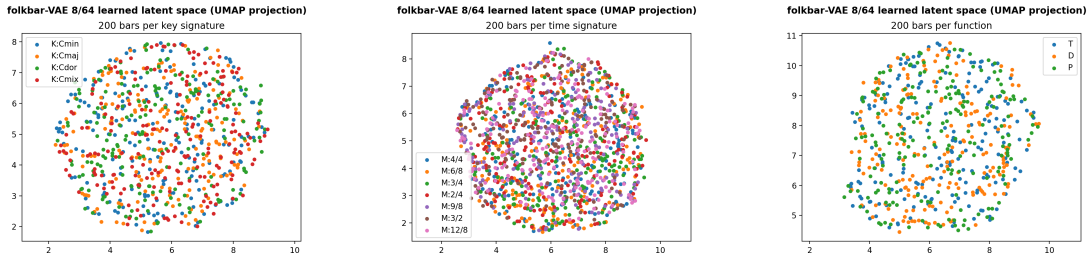


(a) (b) (c)

Figure 3.13: folkbar-VAE 4/32 learned latent space by key (a) and time (b) signature and harmonic function (c).

We continue to see the time signature distinction made above. No other notable changes are present. This trend remains unchanged in the following plots too.

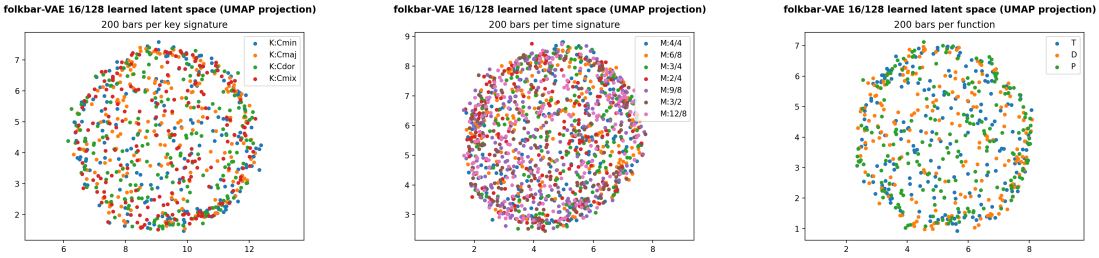
3.3.3 folkbar-VAE 8/64



(a) (b) (c)

Figure 3.14: folkbar-VAE 8/64 learned latent space by key (a) and time (b) signature and harmonic function (c).

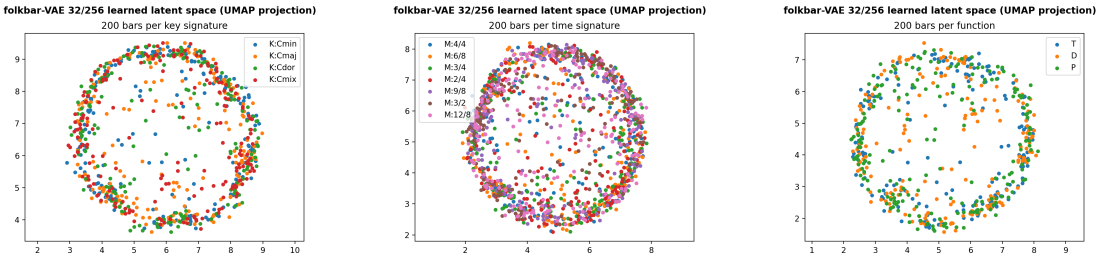
3.3.4 folkbar-VAE 16/128



(a) (b) (c)

Figure 3.15: folkbar-VAE 16/128 learned latent space by key (a) and time (b) signature and harmonic function (c).

3.3.5 folkbar-VAE 32/256



(a) (b) (c)

Figure 3.16: folkbar-VAE 32/256 learned latent space by key (a) and time (b) signature and harmonic function (c).

As in folktune-VAE 32/256, this model's UMAP projection suffers from the already observed artifact of points being pushed outwards, away from the center.

3.3.6 Conclusions

Analysis of the learned latent space does not provide us with useful insight into its structure. The only consisting distinction throughout models is between $\frac{12}{8}$, $\frac{9}{8}$ and $\frac{3}{2}$ bars and other time signatures. This family of models does not appear to be capable of learning useful associations in terms of key and time signature and estimated harmonic function.

3.4 Similarity of bars within tunes

We originally intended to project a complete tune as a sequence of connected latent points corresponding to its bars and inspect the path that they generate. We divided a tune into bars and projected each one as a latent point. We then plotted and connected them in the same order as the corresponding bars in the source. We hoped to gain an interesting insight into the tune's structure and internal repetitions and similarities; however, this analysis did not prove to be as meaningful as we hoped.
As we can see in the following plot, the path created by the sequence of encoded bars does not provide us with useful information on the tune structure. We marked the points corresponding to the start and end of the tune with a star and an "X" respectively. We provide the full series of plots by key signature, meter and function for each model in the Appendix.

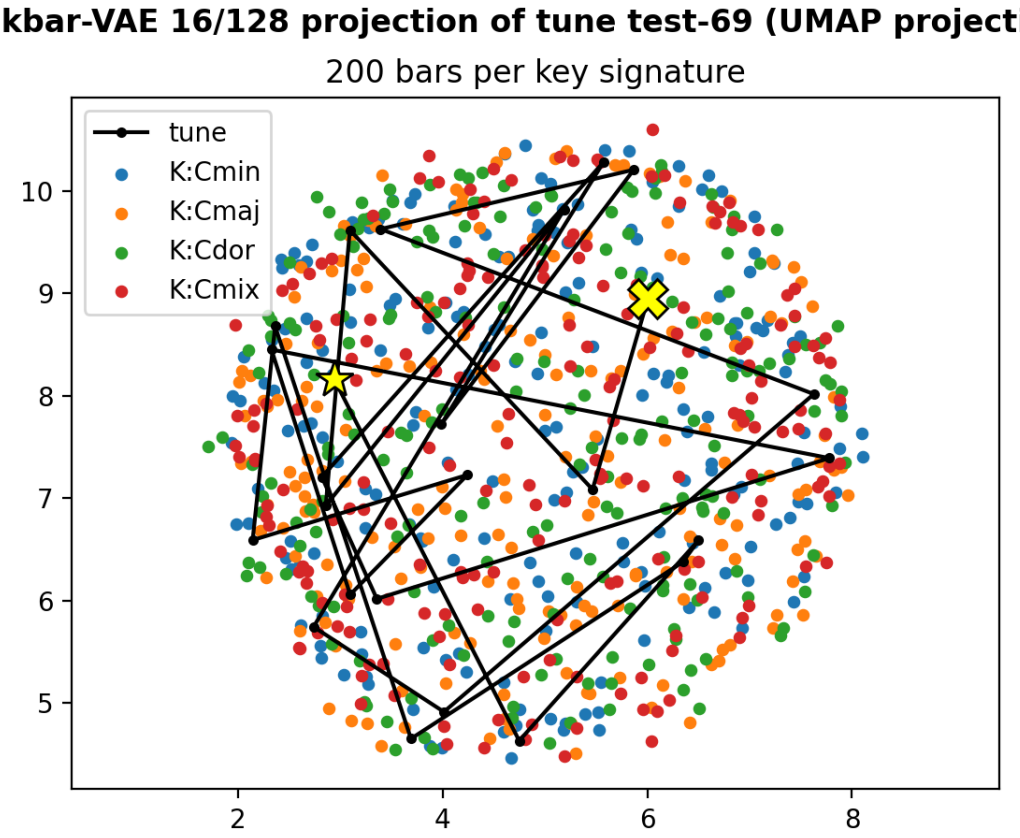


Figure 3.17: Sequence of points corresponding to bars belonging to tune test-69 projected in latent space.

Put aside these uninteresting results, we tried a different approach to highlight a tune's structure and inner similarities. As before, we divided a tune into bars and projected them into latent points with each model. We then computed the Euclidean distance between each possible couple of bars and visualized it as a heatmap.
Irish traditional music makes ample use of repetitions and standard motifs and figures: we expect to see similar bars encoded to points near to each other and, in particular, identical bars should ideally be mapped to the same point and we should see very low distance values when comparing such bars.
We arbitrarily chose tune #69 from the test split as our test tune.
This is the tune's ABC notation:

```
M:6/8
K:Cmaj
|: d | e 2 d c 2 C | E G G A G E | G A c d c A | d e d d e f |
e 2 d c 2 C | E G G A G E | G A /2 B /2 c d c A | c d B c 2 :|
|: d | e d e g 2 e | g a e g e d | c 2 e d c A |1 d e d d B d |
e d e g 2 e | g a e g e d | c 2 e d c A | c d B c 2 :|
|2 d e d d B f | e 2 d c 2 C | E G G A G E | G A c d c A | c d B c 2 |
```

and this is the sheet music it represents.

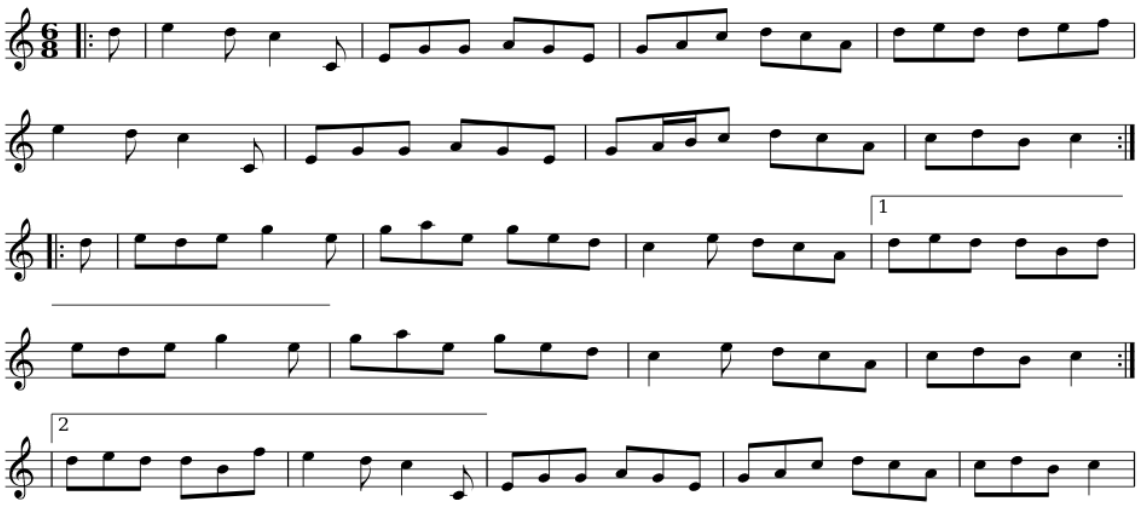
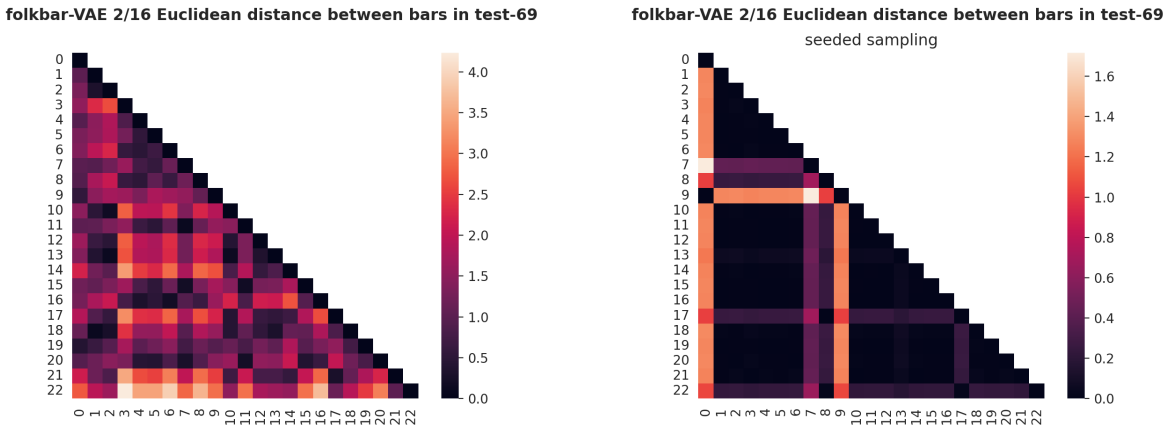


Figure 3.18: Sheet music associated with the ABC notation of tune test-69.

For each folkbar-VAE model we provide two plots: one where latent bar points have been randomly sampled by the model without any seed indication and one where we seeded

the model with `torch.manual_seed(0)` right before forwarding each bar into the model. This last variation essentially removes sampling randomness and ensures that identical bars are mapped to identical points; while this is not generally how one would use such models, it allows us to see correlations between bars at a glance.
We can then compare the plots and see if these correlations are preserved: for each bar, models with a good understanding of the bars' space should learn a distribution "sharp" enough that sampled latent codes inhabit a limited region of the latent space and are projected into points near to each other. On the contrary, "bad" models will learn broad distributions for each bar, and the resulting latent codes will be on average equally distant from each other.

3.4.1 folkbar-VAE 2/16

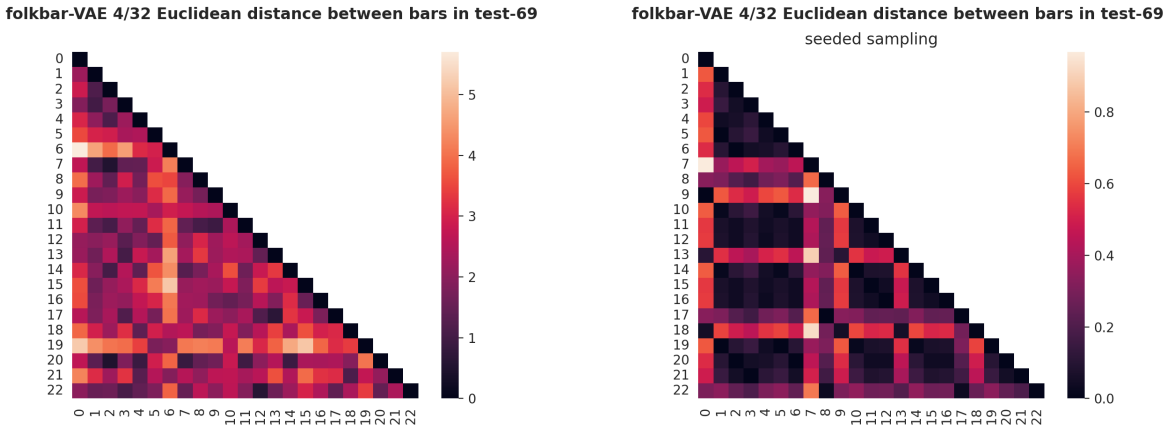


(a) (b)

Figure 3.19: folkbar-VAE 2/16 heatmap of distances between unseeded (a) and seeded (b) pairs of latent projections of bars in tune test-69.

We can immediately see that the seeded projection labels all bars as almost identical to each other. The most notable feature is the bright stripe corresponding to bar 9, which is the short upbeat bar right before the effective tune start. A similar thing happens for bar 7, which is the only bar containing 16th notes, and bar 17, which contains fewer notes than usual, compensated by the upbeat bar at the start when repeating the section. This is not standard musical practice and should be considered incorrect; however, it is common when notating folk tunes to adopt such a notation, that ultimately does not affect the intelligibility of the melody or our models.
We find the relationships above in the first graph too, but in a less evident way due to the randomness introduced by sampling.

3.4.2 folkbar-VAE 4/32

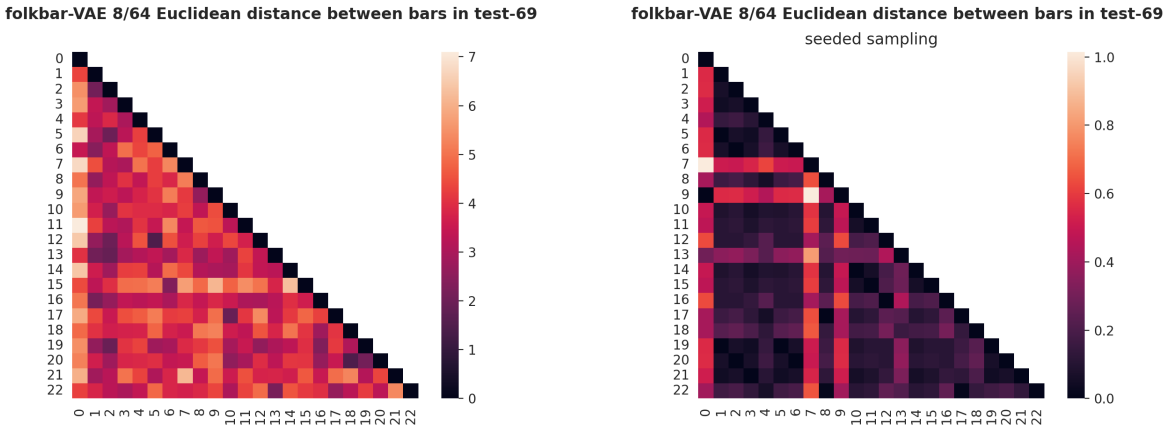


(a) (b)

Figure 3.20: folkbar-VAE 4/32 heatmap of distances between unseeded (a) and seeded (b) pairs of latent projections of bars in tune test-69.

The seeded projection highlights a pattern of rectangles and squares of height 3 repeating after one or more different bars. We can easily trace this pattern to the repeated ascending and descending figures and the reprises of the main theme throughout the tune. In particular, bar 7, containing the first repetition token, is the furthest from the single-note upbeat bars.
Strangely, we do not see these relationships in the first plot: they are shadowed by the higher distance values of other bars, such as bars 6 and 19. Nevertheless, we did not expect these bars to be so far from the others, even from identical bars that repeat in the tune. This signals that the model has learned too broad distributions of bars in order for sampled points to be close to each other.

3.4.3 folkbar-VAE 8/64

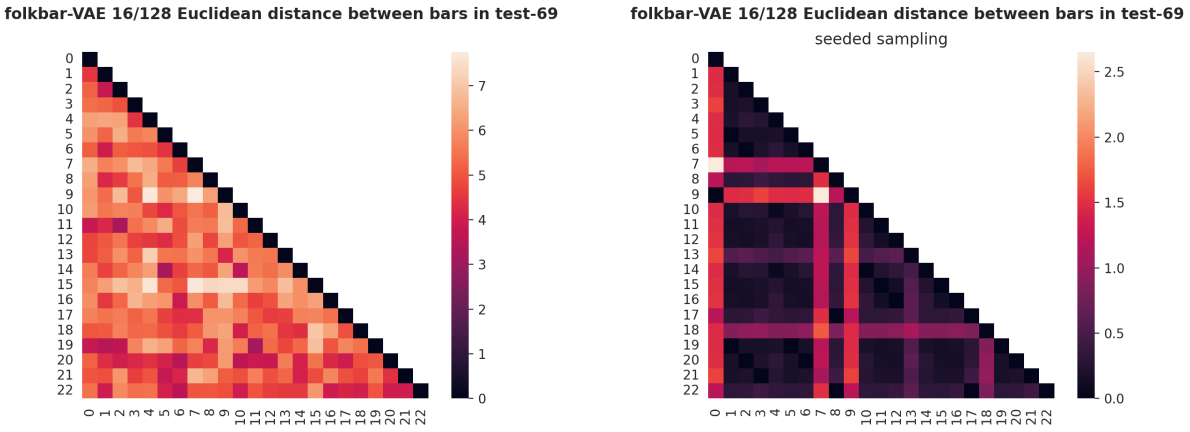


(a) (b)

Figure 3.21: folkbar-VAE 8/64 heatmap of distances between unseeded (a) and seeded (b) pairs of latent projections of bars in tune test-69.

While the seeded plot continues to show the patterns we have already seen, the unseeded one shows average values for each couple. This model appears to almost ignore bars' similarity. This trend continues and is accentuated in the remaining models, which have even less understanding of bars' similarity in terms of Euclidean distance between latent points.

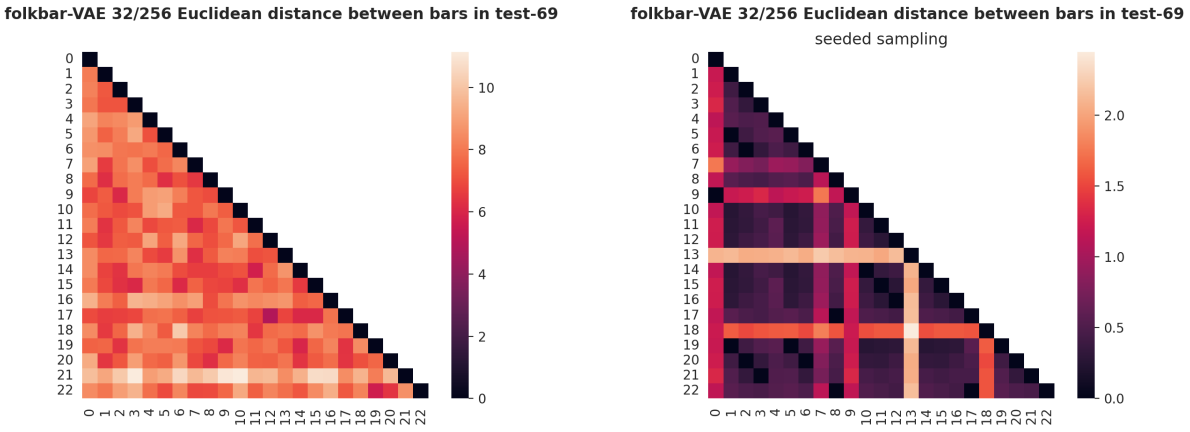
3.4.4 folkbar-VAE 16/128



(a) (b)

Figure 3.22: folkbar-VAE 16/128 heatmap of distances between unseeded (a) and seeded (b) pairs of latent projections of bars in tune test-69.

3.4.5 folkbar-VAE 32/256



(a) (b)

Figure 3.23: folkbar-VAE 32/256 heatmap of distances between unseeded (a) and seeded (b) pairs of latent projections of bars in tune test-69.

3.4.6 Conclusions

Contrary to what one would expect, but in an analogous way to what we have seen with the folktune-VAE models, bigger models do not necessarily bring improvements. In fact, what we have seen shows an opposite trend, with the best results achieved by the lower-dimensional models and far worse results achieved by higher-dimensional ones. We speculate that these models were too big for bars consisting of at most 32 tokens in order to learn useful associations: trivially, the 32-dimensional latent space model folkbar-VAE 32/256 could have learned embeddings in which each component of the 32-dimensional code is mapped to the token occupying that position in the input bar. Such an "alphabetical" encoding obviously does not convey musical meaning, at least in the way we are interested in.

3.5 Generation

In the following section, we show some generative results.
We sampled a random latent code x and asked the decoder to generate a tune with it. We implemented greedy, TOP-k, and TOP-p sampling with temperature.
We used both our worst and best models in terms of ELBO loss, which are folktune-VAE 2/16 and folktune-VAE 32/256 respectively.

3.5.1 folktune-VAE 2/16

This is the worst model in terms of ELBO loss.
The following sampled pieces show musical features and patterns that are common in Irish folk music. However, tunes lack any kind of high-level structure and melodies resemble more a random juxtaposition of melodic fragments rather than a coherent musical piece.
The model has learned to "count" notes in bars correctly, although not reliably. Nevertheless, the rhythmic patterns it generates are unusual and not consistent throughout the piece.
We expected such results because of the reduced size of this model.

```
M:4/4
K:Cdor
|:c>decc2c>d|e>fg2g2g>f|e>fg>fe>dc>B|c>de>fg>fe>d|
c>Bc>de>dc>B|c>Bc>de>fg>f|e>fg>fe>dc>B|c>Bc>de>fg>f|
e>fg>fe>dc>B|c>Bc>de>fg>f|e>fg>fe>dc>B|c>Bc>de>fg>f|
e>fg>fe>dc>B|c>Bc>de>fg>f|e>fg>fe>dc>B|c>Bc>de>fg>f|
e>fg>fe>dc>B|c>Bc>de>fg>f|e>fg>fe>dc>B|c>Bc>de>fg
```

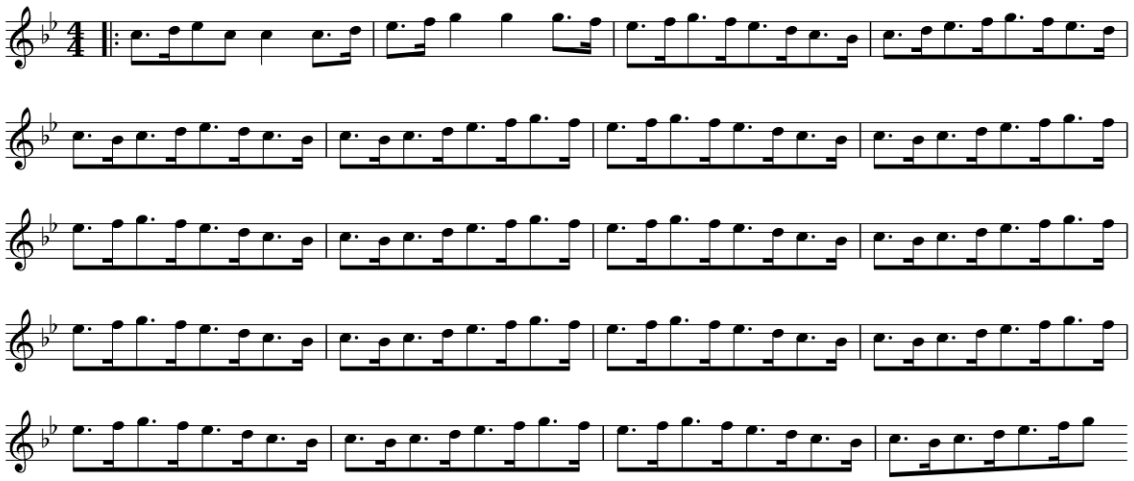


Figure 3.24: Tune generated by folktune-VAE 2/16 with greedy sampling.

```
M:4/4
K:Cmaj
|:c2cAA2cG|(3GAdc2eG2f|e2dd3dB|c2d2BAF2|
F3FD2C2|AGABA2c2|(3GFD ED3D2|CEG2G2F2|
F3CA,2C2|E4D2C2|G2c3B2G|E2D2D2A2|C3E2C2A:|
```

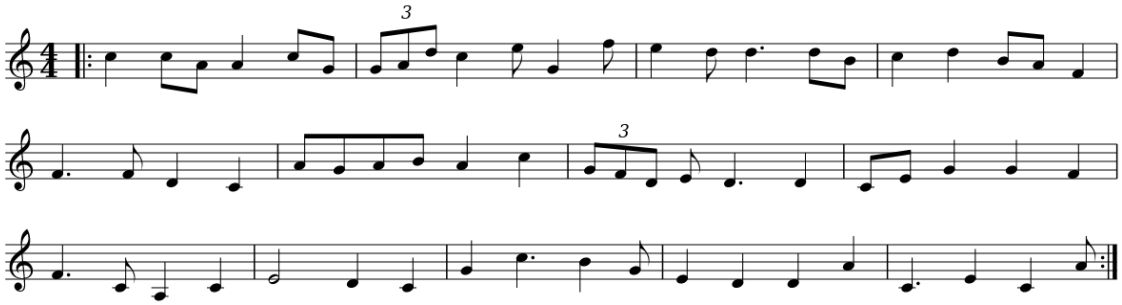


Figure 3.25: Tune generated by folktune-VAE 2/16 with top-k sampling, k=10, temperature 1

```
M:2/4
K:Cmaj
CG CG|A>B c2|Gc cG|FD ED/2D/2|EF G>c|AG A>G|A2 AA|
c>A B/2c/2A|GE Gc| dc BG|AG ED|EG ce|g/2e/2e/2c cG|c3:|
|2(3GFE FG|e/2d/2c de/2c/2|ed ce|cd ec|e/2f/2g/2e/2 ec|c3c|
```

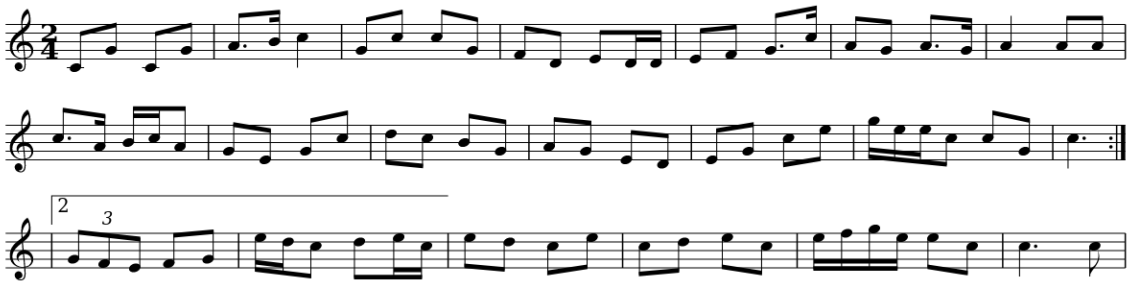


Figure 3.26: Tune generated by folktune-VAE 2/16 with top-p sampling, p=0.9, temperature 1

3.5.2 folktune-VAE 32/256

This is the largest and best model we have trained on complete tunes.
We can immediately see characteristic melodic and rhythmic features used consistently in the scope of a piece, even with reprises and variations. The model reliably counts notes in bars.
Tunes show a high-level musical structure: we see the first bar being recalled every 2 or 4 bars and melodic phrases imply a constant Tonic to Dominant or Tonic to Plagal harmonic movement, which is common practice and the desired feature in this genre of music.
We expected this model to perform best due to its size; however, we are not sure of how large of a contribution the internal representation makes to generation, given that we could not see any interesting results in this model's latent space analysis. We would not be surprised if the very large decoder RNN completely ignores the latent code after generating the first few tokens and carries on with learned probabilities as any other RNN.
Nevertheless, from a purely generative point of view, these results are interesting and musically pleasing.

```
M:4/4
K:Cmaj
|:GF|ECC2EGcG|ECC2DEFD|ECC2EGcG|ECDC DEFD|
ECC2EGcG|ECDC A,CCE|F3G ABcA|GEDEC3:|
|:G|c2cG cGEG|cGEC DCA,G,|C2EG cGEG|cGEC DCA,G,
|C2EG cGEG|cGEC DCA,G,|C2EG cGEG|FDB,DC3:|
```

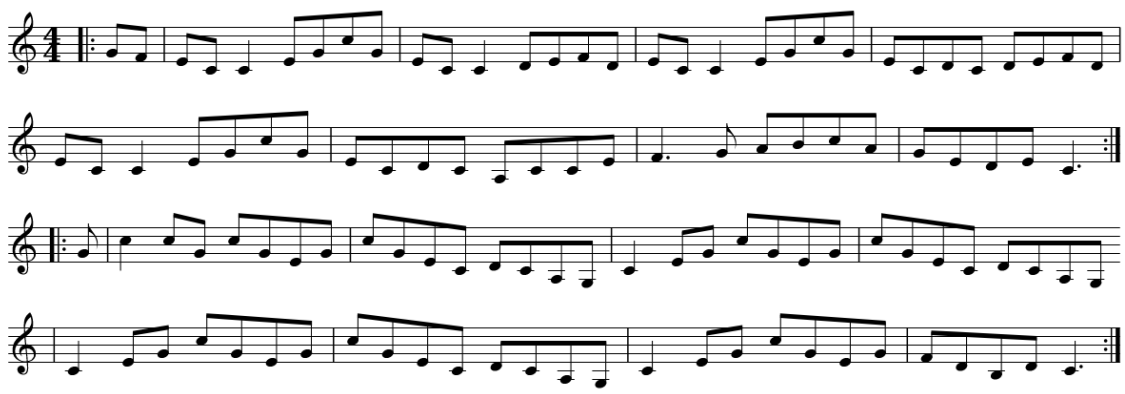


Figure 3.27: Tune generated by folktune-VAE 32/256 with greedy sampling.

```
M:9/8
K:Cmaj
|:CB,C EDC EGc|ECE EDC A,CE|cBc GEC EDC|DED DEC B,3:|
|:CDE GFE FGE|AcA GFD cBA|GEG cAF cAG|EDE CEG A2G:|
```

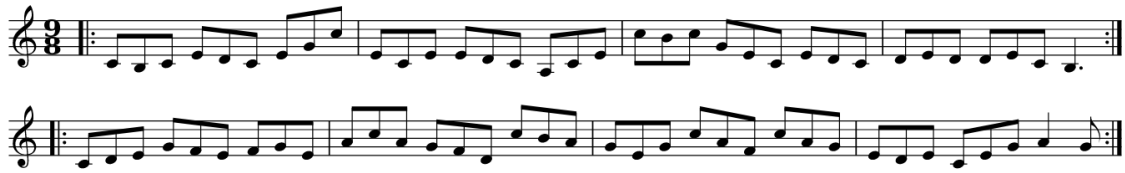


Figure 3.28: Tune generated by folktune-VAE 32/256 with top-k sampling, k=10, temperature 1.

```
M:2/4
K:Cdor
|:Gc cB/2c/2|dc B2|c/2c/2cc2|dc cd|ed cB|G2 c2|dc Bd|c2c2:|
|:gc' bg|fd ec|BB/2c/2 df|ec cB|G2 c>d|eg fd|c>d cB|G2:|
```

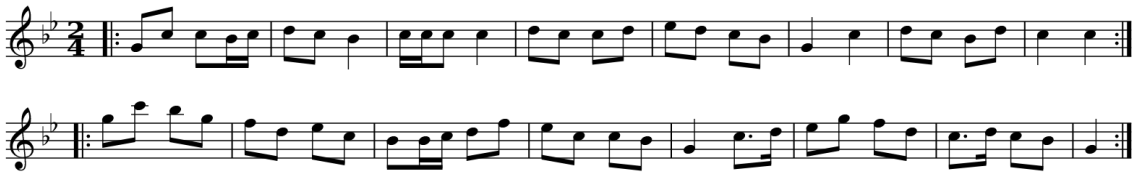


Figure 3.29: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.9, temperature 1.

Chapter 4

Conclusion

The goal of this work was to create a generative model for Irish traditional music which could also provide a meaningful representation of data for musical analysis. As such, we chose to use Variational Autoencoders.
Using a text-based model for natural language as a starting point, we trained two types of models:

- folktune-VAE: trained on a dataset of Irish folk melodies, with an analytic and generative purpose;
- folkbar-VAE: trained on bars extrapolated from the melodies dataset, in order to analyze tunes as a collection of bars.

For each type, we trained five variations of progressively increasing size.
We then tried to explore their internal representation from various musically significant points of view:

- we inspected the latent space of tunes and bars in relation to key, time signature, and estimated harmonic function of bars;
- we searched for links between tunes in a particular style (*i.e.* "reels") and their positioning in latent space relative to other tunes;
- we computed distances between embedded bars in a tune to gain insight into the model's understanding of the similarity between bars.

Finally, we showed and evaluated generative examples.
While we can say that we met our generative goal with the largest folktune-VAE model, we could not find significant links between the models' internal representations and our understanding of music theory.

We found very weak correlations between the latent representations of tunes with the same key or time signature, even weaker when analyzing individual bars. We could see that specific-style tunes had a tendency to group in certain regions of the latent space, but without being clearly separated from other tunes. The analysis of the Euclidean distance between embeddings of bars belonging to a tune showed that our model learned very broad latent distributions that overlap too much in order to highlight the musical properties of the data.
The larger the evaluated model, the less we could find musically significant internal representations, even though generally lower validation loss values for bigger models would suggest the opposite. We found the KL-Divergence to be partly responsible for this behavior.
We can conclude that the internal representation of the analyzed models is not musically significant and cannot be used to perform any kind of musical analysis or genre recognition task, even when the same model yields good generative results. This suggests that learned latent codes are not of vital importance for the decoder, which can minimize the ELBO loss more easily by focusing on the reconstruction term rather than on the KL-divergence; in fact, the models that could give us some kind of musical information on data are also the ones that perform worse in the generation task. Nevertheless, this very issue allows the model to generate coherent and interesting pieces when of a big enough size. The importance of one or the other aspect is ultimately dependent on the user of such models and their goals.

4.1 Future work

There are endless possibilities to try and improve the musical meaningfulness of our models. We cite the ones we considered when realizing this work:

- one could use it as a β-VAE and cap the weight of the KL-divergence to values less than 1 or use cyclical annealing;
- one could train a conditioned VAE by concatenating information to be explicitly encoded in latent space to each latent code;
- one could concatenate the latent code to each hidden layer of the decoder GRU for each time step in order for it to be more influential on the decoded sequence.

Our implementation supports both conditioning and realization as β-VAE. We have briefly experimented with such approaches, but we considered them to be beyond the scope of this work.

4.2 Acknowledgments

We thank the supervisor Maurizio Gabbrielli, who allowed and supported the writing of this thesis; the co-rapporteur Luca Casini, who provided great technical and theoretical aid; Bob L.T. Sturm, the supervisor during the traineeship at the Royal Institute of Technology of Stockholm that inspired this work, together with Nicolas Jonason, Laura Cros Vila, Joris Grouwels and everyone at MUSAiC, who greatly supported and contributed to the core ideas behind this work.

Appendix

Other generative examples

```
M:6/8
K:Cmaj
|:G|c2c cBc|d2d def|edc cGc|c2d e2f|g2g gfe|d2c Bcd|e2c f2d|ecc c2:|
|:d|ecc gcc|agf gec|d2c def|ecc c2g|agf gec|def edc|gec cde|dcB c2:|
|:f|ecc ecc|gcc ecc|gcc ecc|dcd ecc|Gcc ecc|gcc ecc|gcc ecc|dcc c2:|
```

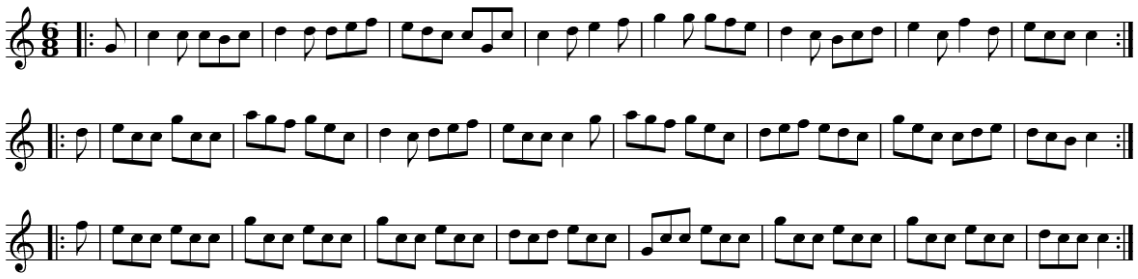


Figure 4.1: Tune generated by folktune-VAE 32/256 with greedy sampling, tempreature 1.

```
M:6/8
K:Cmaj
|:G|c2G E2G|cGE C2G|cGc e2d|cBA G2E|FEF G2F|EGc edc|BGG G2G|AGF E2:|
|:G|cBc G2E|FGA G2E|FEF G2F|EDC D2G|cGc edc|BGG G2d|ecc d2B|c3c2:|
|:f|ecc G2c|ece g2f|ecc dcB|cBc G2F|EFG cGc|ecc G2c|ecc f2d|ecc c2:|
```

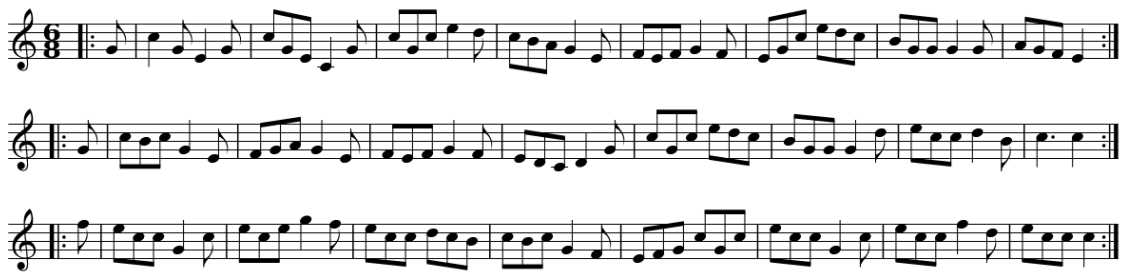


Figure 4.2: Tune generated by folktune-VAE 32/256 with greedy sampling, tempreature 1.

```
M:9/8
K:Cmin
|:E2D ECB, CDE|E2F GEC B,CD|E2C CDE CDE|EDE FDB,C3:|
|:G3 GBd edc|A/2B/2cA BAB def|g2e dcB cBA|G2F DB,C DED:|
|:G2G cGc edc|g2e fed ede|f/2g/2ff edc BAB|GFE BGBc2G:|
```

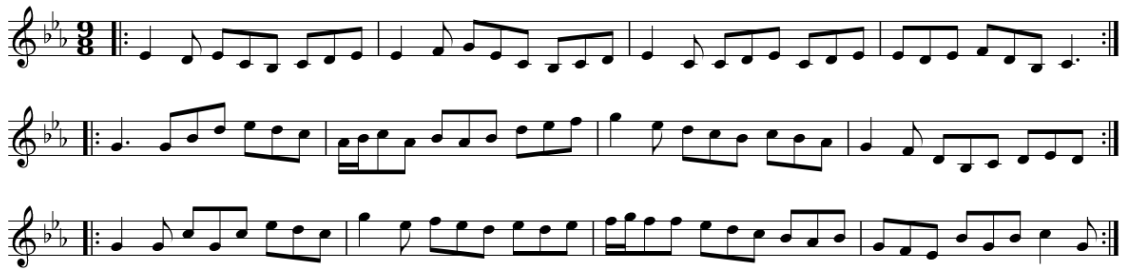


Figure 4.3: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.8, temperature 1.25.

```
M:4/4
K:Cmaj
|:c2ec GFED|C2EC G,CEG|cege cege|defe defd|
cGec GFED|CEGE F2EF|G2Bd cdeg|1fdBG c2eg:||2fdcB c2cd
|:ecc2 ecgc|ecc2 acge|f2fg fdde|1fagf eccA|efec f2gf:||2gfed eccB|
```

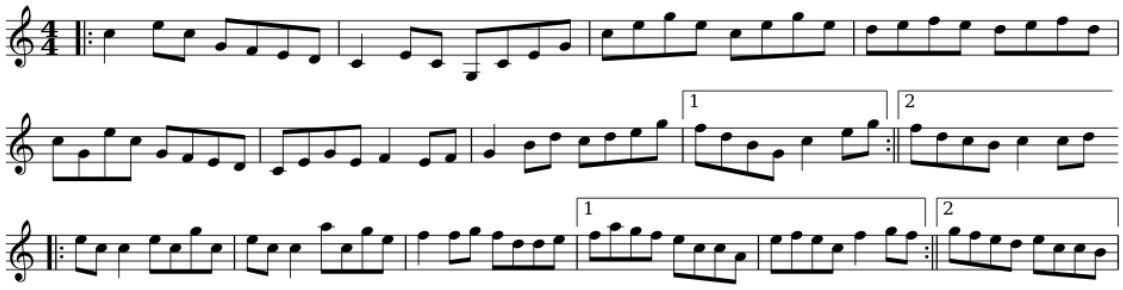


Figure 4.4: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.8, temperature 1.25.

```
M:4/4
K:Cmaj
|E3F GAGF|ECDE FGAB|c2Bc AFDE|FGAF EDD2|
CDEF GcBG|FDBD c2Bc|AFDF EFGE|FDEC D2CD:|
|:cBcd edcB|AcGE D3c|BdcB AGFA|GEFD ECCE|
GccB c3d|edce dBAB|cBcG AGFD|1DCB,D C2GF:||2DCDB,C4|
```

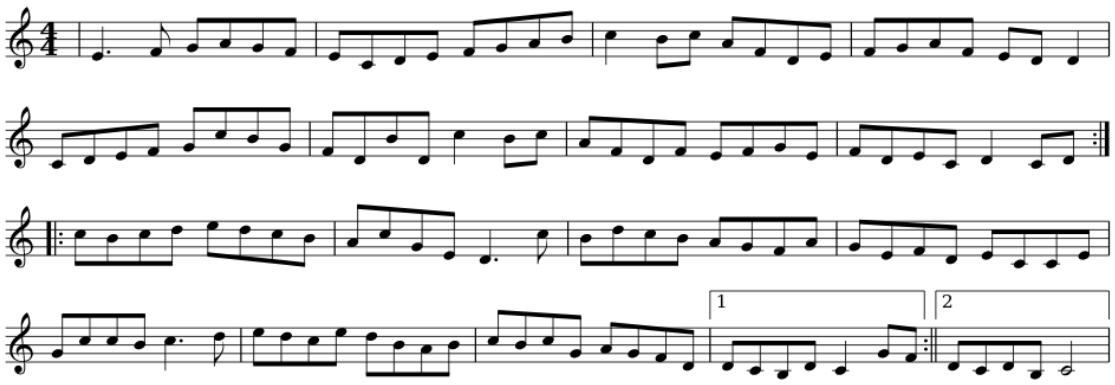


Figure 4.5: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.8, temperature 1.25.

```
M:12/8
K:Cmin
ef|:e2c d3 d3ed|c3d2e dc2cBG|1edec2Gc3e2g:||2e2d cBG Bc2d2ed2e:|
|:f2de2fg3gab|c'bag3a2f agf|d2c deg fedc3:|
```

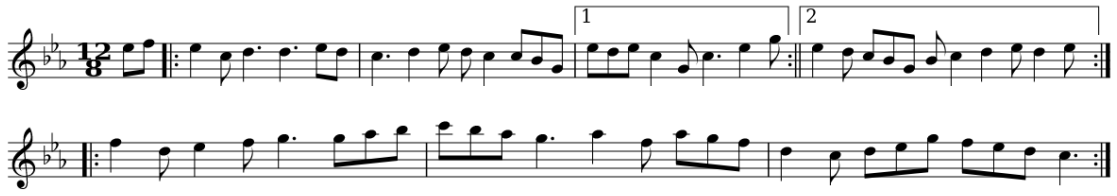


Figure 4.6: Tune generated by folktune-VAE 32/256 with top-k sampling, k=5, temperature 2.

```
M:6/8
K:Cmaj
|:G|ECC GCC|AGE CDE|GAc GEC|DED D2G|
ECC GCC|ECC G2c|AGE CDE|GAG G2:|
|:E/2G/2|AGE cGE|GAG GED|EGG GEC|DDD DE/2F/2G|
AGE GEC|EGc c2A|GEG AGE|GAB c2:|
```

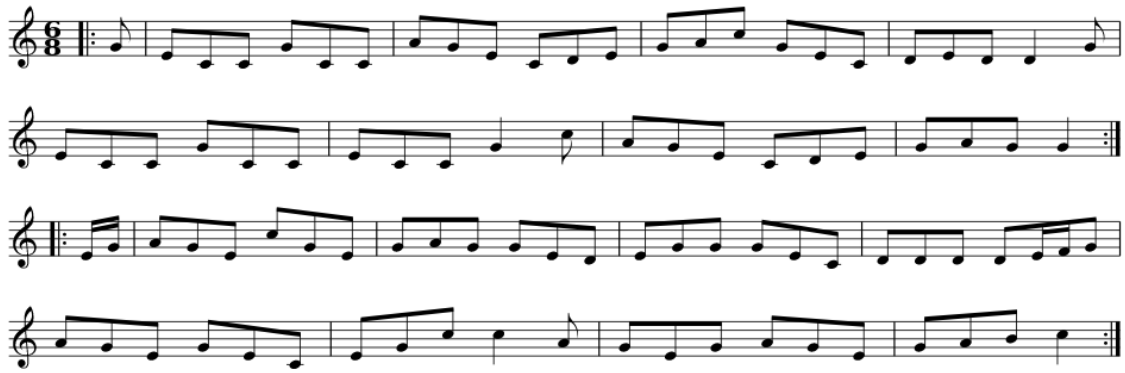


Figure 4.7: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.5, temperature 1.5.

```
M:6/8
K:Cmaj
|:G,|C>DC E>FG|A>GA G2C|D>ED DE/2F/2G|A>GAB>AG|
E>FG C>DC|D>ED D2E|F>EF G>AG|F>ED C2:|
|:F|E>FG c2G|A>Gc c2G|A>GE C2C|D>EF G2F|
E>FG A>GE|FDB, C2G,|C>DE E>FG|c>GA GEC|F>GA G2E|C>DEC2:|
```

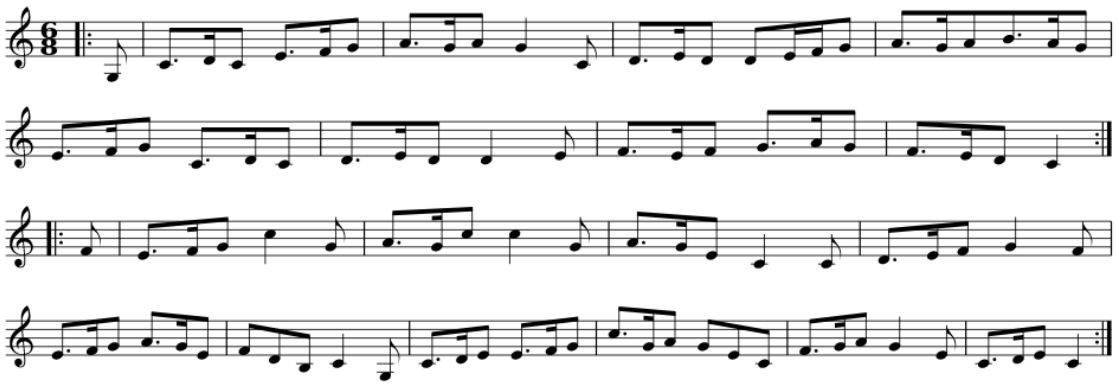
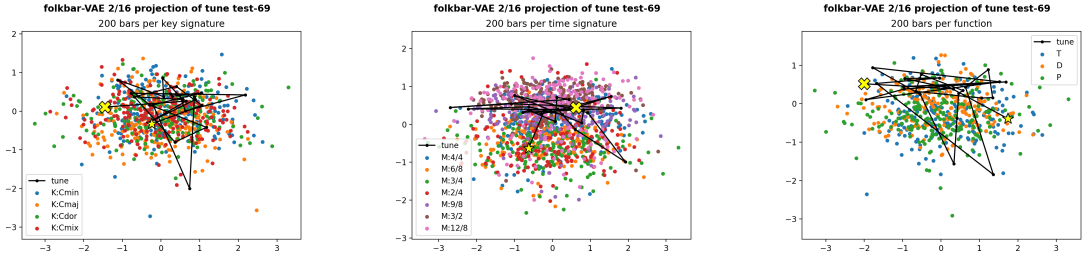


Figure 4.8: Tune generated by folktune-VAE 32/256 with top-p sampling, p=0.5, temperature 1.5.

Tune projection as latent paths

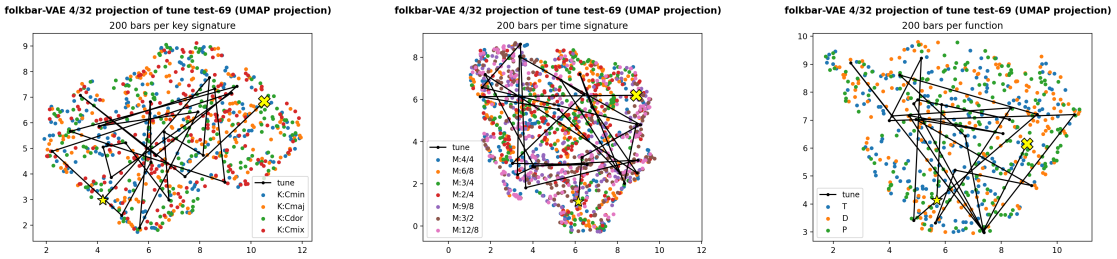
folkbar-VAE 2/16



(a) (b) (c)

Figure 4.9: Latent path of bars belonging to tune test-69 projected in latent space by key (a) and time (b) signature and harmonic function (c), folkbar-VAE 2/16.

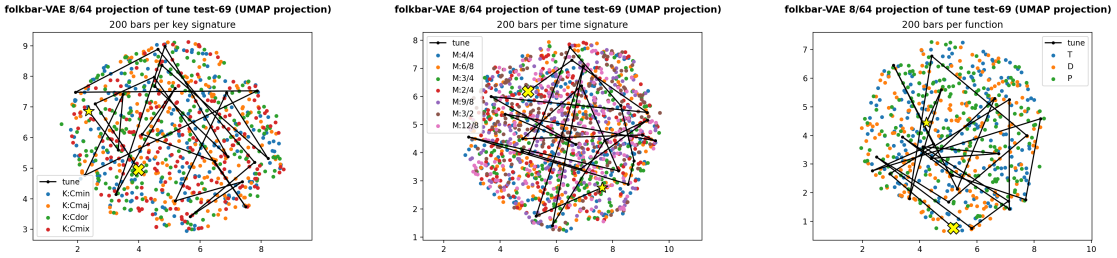
folkbar-VAE 4/32



(a) (b) (c)

Figure 4.10: Latent path of bars belonging to tune test-69 projected in latent space by key (a) and time (b) signature and harmonic function (c), folkbar-VAE 4/32.

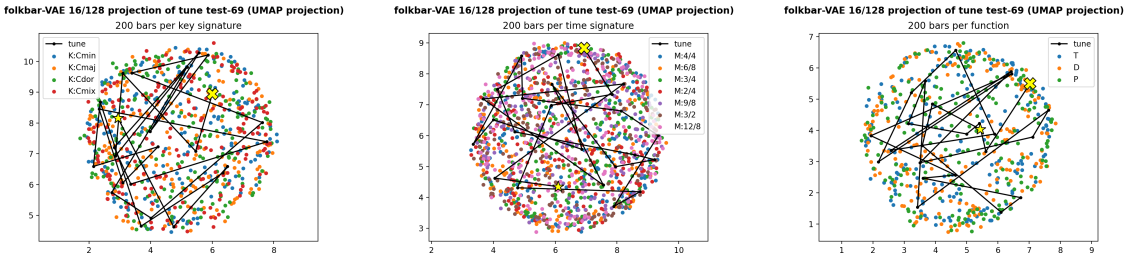
folkbar-VAE 8/64



(a) (b) (c)

Figure 4.11: Latent path of bars belonging to tune test-69 projected in latent space by key (a) and time (b) signature and harmonic function (c), folkbar-VAE 8/64.

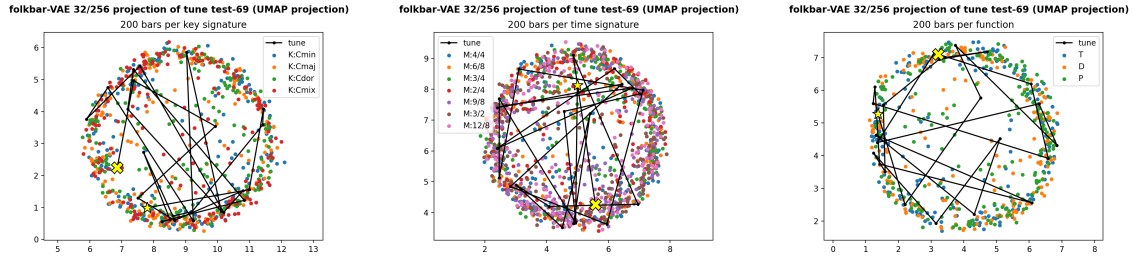
folkbar-VAE 16/128



(a) (b) (c)

Figure 4.12: Latent path of bars belonging to tune test-69 projected in latent space by key (a) and time (b) signature and harmonic function (c), folkbar-VAE 16/128.

folkbar-VAE 32/256



(a) (b) (c)

Figure 4.13: Latent path of bars belonging to tune test-69 projected in latent space by key (a) and time (b) signature and harmonic function (c), folkbar-VAE 32/256.

Bibliography

[1] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *CoRR*, abs/1511.06349, 2015. arXiv: 1511.06349.

[2] L. Casini and B. L. T. Sturm. Tradformer: A Transformer Model of Traditional Music. In *Proc. Int. Joint Conf. Artificial Intell.*, 2022.

[3] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259, 2014. arXiv: 1409.1259.

[4] J. D. Fernandez and F. Vico. AI Methods in Algorithmic Composition: A Comprehensive Survey. *J. Artificial Intell. Res.*, 48(1):513–582, October 2013.

[5] E. Hallstrom, S. Mossmyr, B. L. Sturm, V. H. Vegeborn, and J. Wedin. From Jigs and Reels to Schottisar och Polskor: Generating Scandinavian-like Folk Music with Deep Recurrent Networks. In *Proc. Sound and Music Computing Conf.*, 2019.

[6] Lejaren Hiller, Leonard M. Isaacson, and Lejaren Hiller. *Experimental music: composition with an electronic computer*. Greenwood Press, Westport, Conn, 1979.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997.

[8] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. Music Transformer: Generating Music with Long-Term Structure. *arXiv preprint arXiv:1809.04281*, 2018.

[9] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2013.

[10] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[11] Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. *CoRR*, abs/1803.05428, 2018. arXiv: 1803.05428.

[12] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova. Music Transcription Modelling and Composition Using Deep Learning. In *Proc. Conf. Computer Simulation of Musical Creativity*, Huddersfield, UK, 2016.