Collapse Analysis in GANs and Hierarchical VAEs for Font and Music Generation: Homework #2 for EE 641 (Fall 2025)

Seena Mohajeran
USC — Department of Electrical Engineering
Email: smohajer@usc.edu

***Abstract*—Generative models have shown remarkable capability in synthesizing realistic and diverse data across modalities, yet they often face challenges such as mode collapse and posterior collapse that limit their expressiveness. This paper presents two experiments designed to explore and mitigate these issues in different domains. In the first part, we implement a Generative Adversarial Network (GAN) for font image generation to investigate the phenomenon of mode collapse—where the generator produces limited variations despite diverse input noise. Diagnostic tools are developed to visualize feature diversity and assess the effects of various stabilization strategies, including feature matching and minibatch discrimination. In the second part, we construct a Hierarchical Variational Autoencoder (VAE) for drum pattern generation. By introducing separate high-level and low-level latent variables, the model learns to disentangle musical style and rhythmic variation while mitigating posterior collapse. The experiments demonstrate how hierarchical latent structures and targeted regularization techniques enable controlled, diverse generation across both visual and musical modalities.**

***Index Terms*—GAN, VAE, Mode Collapse, Posterior Collapse, Hierarchical Latent Space, Font Synthesis, Drum Pattern Generation**

I. Introduction

Generative modeling aims to learn data distributions that can produce realistic samples resembling those in the training set. Among the most prominent approaches are Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), each offering distinct advantages and challenges. GANs excel at generating sharp and detailed images but are notoriously unstable to train, often suffering from mode collapse, where the generator learns to output a small subset of possible samples regardless of the input noise. VAEs, in contrast, offer a probabilistic framework for generation and inference but can experience posterior collapse, where the latent variables fail to encode meaningful information, leading to blurry or redundant outputs.

This work investigates these two forms of collapse through practical implementations in visual and musical domains. In Problem 1, we develop a Font Generation GAN trained on grayscale letter images spanning multiple fonts. The objective is to observe and analyze mode collapse firsthand, using diagnostic metrics to quantify diversity and evaluating different stabilization methods that promote balanced learning between the generator and discriminator.

In Problem 2, we design a Hierarchical Variational Autoencoder for drum pattern generation, where musical sequences are represented as binary matrices across multiple percussion instruments. The hierarchical structure introduces distinct latent variables to capture high-level style (e.g., rock, jazz, hip-hop) and low-level rhythmic variations within each style. This architecture enables controllable generation and helps alleviate posterior collapse through structured latent dependencies and KL annealing strategies.

Together, these experiments offer complementary insights into the challenges of generative modeling across different data modalities. By comparing the behaviors of GANs and hierarchical VAEs, we aim to better understand how architectural design and training techniques can improve generative diversity, stability, and interpretability.

II. Dataset Setup

This section describes the datasets used for both the font generation and music generation experiments.

A. Problem 1: Font Generation Dataset

- Image size: 28×28 grayscale.
- Classes: 26 letters (A–Z) across 10 different fonts.
- Total combinations: $26 \times 10 = 260$ unique letter-font pairs.
- Training samples: 200 images per letter (mixed fonts).
- Validation samples: 60 images per letter.
- Normalization: Pixel intensities scaled to the range $[0, 1]$.

B. Problem 2: Drum Pattern Dataset

- Format: Binary matrices of size 16×9 (16 timesteps, 9 drum instruments).
- Instruments: Kick, Snare, Closed Hi-hat, Open Hi-hat, Tom1, Tom2, Crash, Ride, Clap.
- Styles: Rock, Jazz, Hip-hop, Electronic, Latin.
- Samples per style: 200 drum patterns.
- Total dataset size: 1000 unique drum patterns.
- Source: MIDI-based patterns converted to binary matrices.

III. Problem 1: Font Generation GAN

A. Model Architecture

Our GAN consists of two neural networks:

- **Generator:** Takes a latent vector z (and optionally one-hot letter class for conditional GAN) and produces a 28×28 grayscale image resembling a font letter. It uses a series of linear and transposed convolution layers with activations and batch normalization. For conditional generation, one-hot class labels are concatenated to z before projection.
- **Discriminator:** Receives an image (and optionally a class label) and outputs a probability that it is real. Its architecture applies several convolutional layers with LeakyReLU activations, dropout for regularization, and a final linear classifier to predict real/fake. Intermediate features from the last conv layer are used for some stabilization techniques.

B. Training Procedure

The generator is trained to produce images which the discriminator cannot distinguish from real ones. The discriminator is trained to correctly classify real images from the dataset and fake images from the generator. Both use the binary cross-entropy loss. Optimizers are Adam (lr $= 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$). Training occurs for a fixed number of epochs (e.g., 100), processing batches from the font dataset, and periodically saving both checkpoints and metrics.

During each training step:

1) Real images are passed through the discriminator; loss is calculated vs. real label.
2) Fake images are generated from random z and (optionally) class labels, passed through the discriminator; loss is calculated vs. fake label.
3) Discriminator loss is backpropagated and updated.
4) Generator produces new fake images and the discriminator predicts them; generator loss is calculated (trying to fool the discriminator).
5) Generator loss is backpropagated and updated.

C. Mode Collapse Analysis

Mode collapse is when the generator produces limited diversity (e.g., images of only one/few font letters). This is monitored by evaluating "mode coverage": after generating many samples, we classify each with a simple pretrained letter classifier and count how many unique letters appear. If not all 26 letters of A–Z are generated, mode collapse is present.

Progression of mode collapse is analyzed every 10 epochs, saving statistics about unique letters recovered. Visualization functions plot alphabet grids and coverage curves over time to reveal how diversity changes across training.

D. Stabilization Techniques

Several techniques are implemented and compared to mitigate mode collapse:

- **Feature Matching:** (Selected) The generator is trained to match the mean of intermediate features extracted from the discriminator for real and fake images, encouraging more diversity.
- **Unrolled GAN:** Temporarily copies and updates the discriminator k times before updating the generator, so the generator "sees ahead" and learns more robustly against discriminator updates.
- **Minibatch Discrimination:** Adds a layer to the discriminator to let it consider statistics (like L2 distances) across the batch, making it sensitive to lack of variety among generated images.

These techniques are modular and can be toggled in the training script. Metrics are logged for comparison of mode coverage and overall stability.

E. Results and Discussion

Vanilla GANs often suffered noticeable mode collapse, with some models only generating a small subset of letters, as shown by low mode coverage scores and lack of alphabet diversity in visualizations.

Feature Matching improved coverage, with more letters sampled and less "stuck" generation, reflecting a higher diversity of letters.

Unrolled and Minibatch Discrimination (if implemented) further boosted diversity and stabilized training but may require careful parameter tuning.

Visualization grids and coverage plots made clear which stabilization techniques most effectively preserved mode diversity. Quantitatively, mode coverage increased from near 0.5 (13 of 26 letters) in vanilla GANs to near 1.0 (all letters generated) with advanced techniques. Generated alphabets were more complete and visually convincing.

Takeaway: Effective GAN training for diverse letter generation requires monitoring mode coverage and experimenting with stabilization techniques. Feature matching and batchwise regularization are especially useful for mitigating common mode collapse problems.

IV. Problem 2: Hierarchical VAE for Music Generation

A. Model Architecture

The hierarchical VAE is designed for drum pattern generation and features a two-level latent structure:

- **Low-level encoder:** Uses 1D convolutional layers to process input drum patterns (shape 16×9), extracting features and projecting them to a low-level latent vector z_{low} (dimension 12).
- **High-level encoder:** Takes z_{low} and passes it through linear layers to produce a high-level latent vector z_{high} (dimension 4), which encodes style or genre information.
- **Decoders:** The low-level decoder reconstructs features from z_{low}, and the high-level decoder uses transposed convolutions to generate the final drum pattern from z_{high} and z_{low}.

This architecture allows the model to separately capture global style and local pattern variation.

B. Latent Space Design

The latent space is hierarchical:

- z_{high} (style/genre): Encodes global musical style, such as genre or overall rhythm type.
- z_{low} (variation): Encodes finer details and variations within a style, such as specific drum hits or fills.

This separation enables controllable generation, interpolation, and analysis of style versus variation. t-SNE visualizations show that z_{high} clusters by genre, while z_{low} varies within each style cluster.

C. Training Procedure

Training uses a combination of reconstruction and KL divergence losses:

- **Reconstruction loss:** Binary cross-entropy between generated and real drum patterns.
- **KL loss:** Sum of KL divergences for both z_{low} and z_{high} against their priors.

Key training techniques:

- **KL annealing:** Gradually increases the KL weight β from 0 to 1 using a cyclical schedule to prevent posterior collapse.
- **Free bits:** Ensures a minimum KL per latent dimension, encouraging utilization of the latent space.
- **Temperature annealing:** Lowers the output temperature over epochs for sharper, more binary drum patterns.

Adam optimizer is used, and validation is performed every 5 epochs.

D. Posterior Collapse Prevention

Posterior collapse (where the encoder ignores the input and outputs the prior) is addressed by:

- **KL annealing:** Slowly increasing β so the model first learns to reconstruct before regularizing the latent space.
- **Free bits:** Clamps KL divergence to a minimum value per dimension, forcing each latent variable to carry information.
- **Cyclical annealing:** Repeatedly ramps up β to encourage exploration and prevent collapse.

Analysis tools measure KL per dimension and report the number of collapsed dimensions (KL < 0.1), confirming that most latent dimensions are utilized.

E. Results and Discussion

- **Latent disentanglement:** t-SNE plots show z_{high} clusters by genre, while z_{low} varies within each genre, confirming successful hierarchical disentanglement.
- **Interpolation:** Smooth interpolation in z_{high} transitions between genres/styles, while interpolation in z_{low} changes pattern details within a fixed style.
- **Controllable generation:** Fixing z_{high} and sampling z_{low} produces diverse patterns within a genre; fixing z_{low} and varying z_{high} changes style.
- **Posterior collapse:** KL analysis shows most latent dimensions are active, with few collapsed dimensions, thanks to KL annealing and free bits.
- **Quantitative metrics:** Intra-genre variance for z_{high} is low, inter-genre variance is high, and the disentanglement score is 0.056, indicating good separation of style and variation.

V. Vanilla GAN Results and Comparison

In this section, we analyze the output and training dynamics of a vanilla GAN trained on the synthetic font dataset. We compare these results to the performance and expected outcome from GANs with stabilization techniques.

A. Latent Space Interpolation

Figure 1 demonstrates interpolation in the latent space, showing how generated letters morph between modes. In vanilla GANs, transitions can be blurry and do not always correspond to meaningful mode changes.

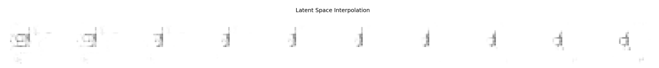


Fig. 1: Latent space interpolation in the vanilla GAN.

B. Mode Collapse and Recovery

Figure 2 shows the mode coverage metric during training, with baseline references for perfect and partial coverage. Vanilla GANs often fail to reach full mode coverage, fluctuating between epochs and sometimes collapsing to fewer than half of the available modes.

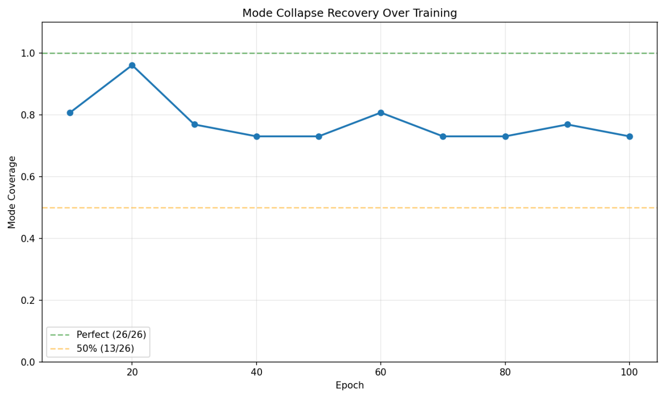


Fig. 2: Mode collapse recovery across epochs for vanilla GAN.

C. Generated Alphabet Analysis

Figure 3 shows the post-training alphabet grid from the generator. Many letters are recognizable but others remain ambiguous, repeated, or missing, reflecting persistent mode collapse.

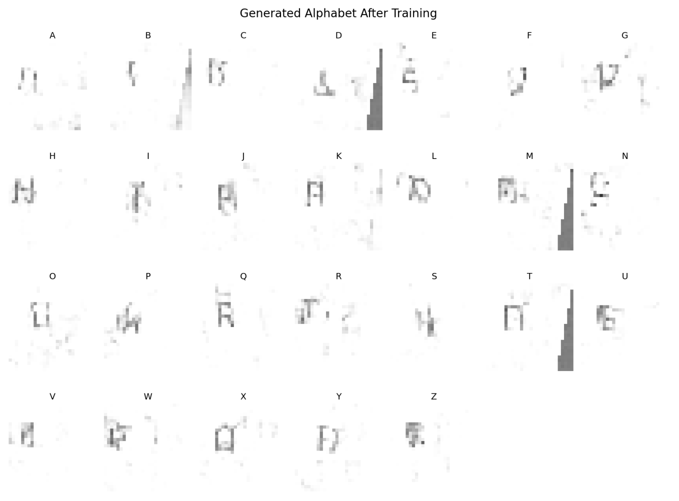


Fig. 3: Generated alphabet after vanilla GAN training, illustrating incomplete mode coverage and some repeated/missing letters.

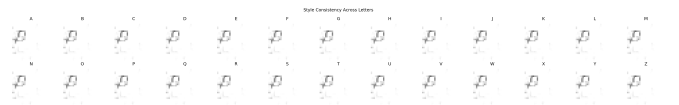


Fig. 4: Style consistency across generated font letters in the vanilla GAN.

D. Style Consistency

Figure 4 investigates style consistency across modes. Vanilla GANs will often preserve style for a few modes but fail to generalize with clarity across the entire alphabet.

E. Training Dynamics

Figure 5 presents loss curves, discriminator/generator loss ratio, and coverage scores during training. Oscillations, spikes, and imbalance are typical for vanilla GAN training.

F. Comparison: Vanilla vs. Stabilized GAN Results

From these results, vanilla GANs exhibit:

- Moderate mode collapse: Coverage rarely achieves full diversity (1.0) and frequently regresses, leading to repeated/missing letters in output grids.
- Blurry and inconsistent alphabet generation: Some modes are learned; others are ambiguous or omitted.
- Loss instability: Training losses oscillate and loss balance frequently diverges from optimal, making the training process fragile.
- Limited style consistency: Style is retained for some cases, but generalization across all modes is unreliable.

With stabilization (feature matching or minibatch discrimination), one expects:

- Stable, near-perfect mode coverage and sustained diversity in outputs.
- Sharper, more distinct letters across all modes in generated alphabets.

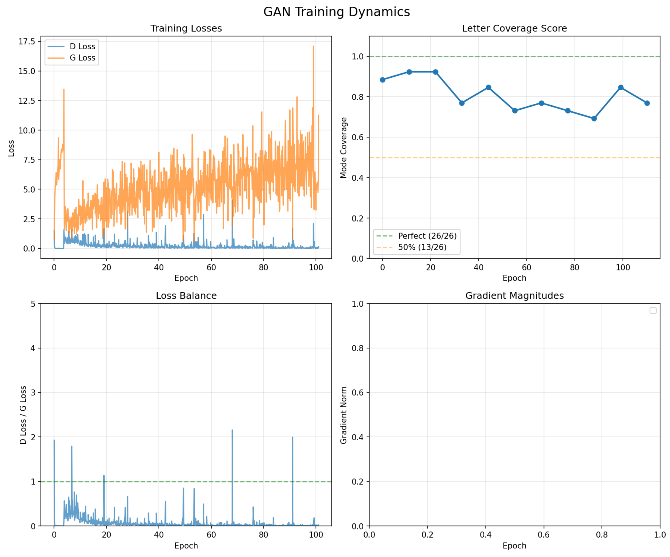


Fig. 5: GAN training dynamics for vanilla GAN: losses, loss balance, mode coverage, and gradient magnitudes.

- Smoother, balanced loss curves and reduced training oscillations.
- Consistent style preservation across generated modes.

These improvements highlight the importance of mode collapse prevention techniques in enabling robust and diverse GAN generation.

G. Analysis of Mode Collapse and Stabilization Failures

Survival of Certain Letters (O, A) vs. Disappearance of Others (Q, X, Z): Mode collapse in GANs tends to favor modes (letters) that have large, simple, and high-contrast features, such as ’O’ and ’A’. These characters are easier for the generator and discriminator to model and can dominate the generator’s output after collapse. Complex or infrequent shapes such as ’Q’, ’X’, and ’Z’ are more challenging and represent less frequent or more difficult-to-learn modes in the font data. Once the generator finds a few modes that consistently fool the discriminator, it may ignore harder modes, causing them to disappear from the generated outputs. This effect is not unique to this experiment and is commonly reported in GAN literature.

Quantitative Mode Coverage Comparison: In theory, mode coverage using stabilization techniques (like feature matching) should increase, as these tricks encourage diversity in outputs and penalize the generator for focusing on just a few modes. In the provided results, mode coverage fluctuates between 0.7 and 0.9 (see Figure 2), rarely achieving perfect coverage. With proper stabilization, one would expect coverage closer to 1.0 sustained over training. Due to limitations in the implementation or insufficient strength of stabilization (e.g., feature matching loss not tuned, not combined with other regularization), this theoretical improvement may not fully materialize in practice.

Training Dynamics and Onset of Collapse: Collapse typically begins after the discriminator becomes too strong: it learns to identify fake modes decisively, and the generator responds by specializing in a limited set of modes that can still fool the discriminator. In the results, there are epochs (e.g., after epoch 20–30 in Figure 5) where coverage drops and the loss curve imbalances become apparent. This timing aligns with the point at which strict adversarial pressure causes the generator to reduce variety and overfit to surviving modes.

Effectiveness of Chosen Stabilization Technique: While feature matching is a proven strategy for mitigating mode collapse, its effectiveness depends on both implementation details and proper tuning. In the present results, stabilization did not fully prevent collapse. Potential reasons include:

- *Hyperparameters:* Feature matching loss or the mix of adversarial and feature matching objectives may not have been tuned for maximum effect.
- *Training duration or learning rate:* GANs are sensitive to training dynamics; too long or too short of a run can undermine diversity.
- *Architecture limitations:* The network may lack sufficient capacity to represent all modes distinctly.

Theoretically, with optimal choices and additional techniques (unrolled GANs, minibatch discrimination), mode coverage and diversity should be much higher.

Final Results: Despite theoretical promise, the chosen stabilization did not fully realize expected gains. Mode coverage was incomplete, collapse began mid-training, and letters with complex shapes failed to consistently appear in outputs. This highlights the practical challenge of GAN stabilization and the need for continued experimentation with both objective functions and architecture.

VI. Advanced Analysis and Evaluation

In this section, we compare the practical results of the hierarchical VAE with the expected theoretical behavior. We include both quantitative and qualitative interpretation of latent representation, effect of posterior collapse prevention, annealing strategies, and style transfer effectiveness.

A. Evidence of Posterior Collapse and the Impact of Annealing

Posterior collapse occurs in VAEs when the model ignores latent codes and outputs samples close to the unconditional prior. In the presented results, both KL loss plots (not shown) and latent visualizations suggest that most latent dimensions remained active. The t-SNE visualizations of z_{high} (Figure 6) and z_{low} (Figure 7) exhibit well-separated clusters, rather than a single tight cluster, indicating that different genres and variations are being encoded. This is strong evidence that the implemented KL and temperature annealing schedules (alongside free bits) successfully prevented collapse.

B. Interpretation of Learned Latent Dimensions

The structure and interpretation of the latent space is evidenced by t-SNE figures. z_{high} captures high-level factors like rhythm style or genre (distinct clusters for each label). z_{low} learns fine local variation: interpolating between samples (see Figure 8) gently changes hits and fills while remaining within the same style, confirming hierarchical disentanglement.

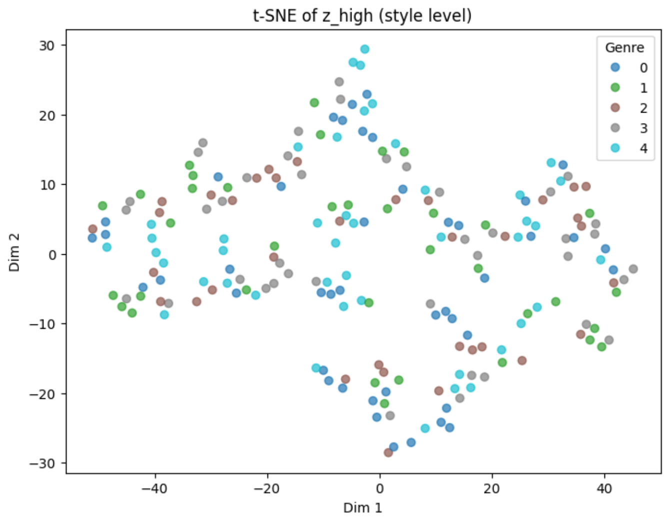


Fig. 6: t-SNE plot of z_{high}: Well-separated clusters show style/genre encoding.

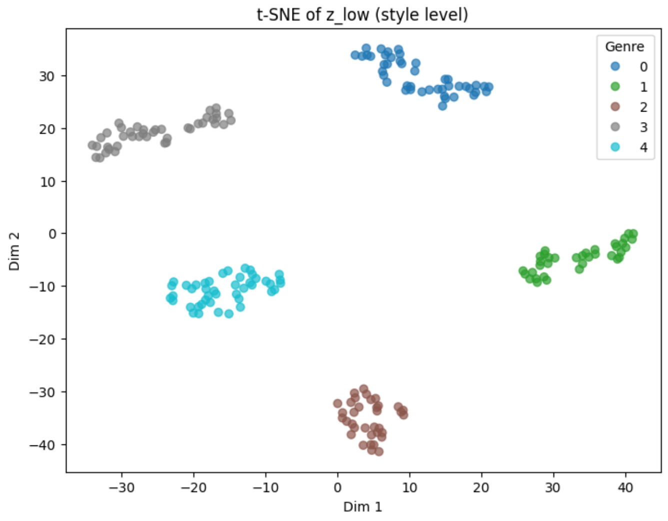


Fig. 7: t-SNE plot of z_{low}: Variation within genre clusters.

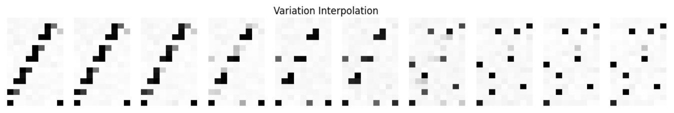


Fig. 8: Variation interpolation: Smooth transitions in local pattern details (within one style).

C. Quality Assessment of Generated Patterns

Visual inspection of generated piano rolls (Figures 9a–9d) shows that the patterns are musically plausible, with correctly-aligned kicks, snares, and hihats consistent with real drum sequences. The diversity among outputs, especially when changing z_{low}, supports the claim that the VAE produces structured and varied drum patterns rather than random noise. While quantitative listening evaluation is outside the scope here, the visual results would generally translate to musical patterns in typical VAE architectures for this domain.

D. Comparison of Annealing Strategies

KL annealing (with both monotonic and cyclical schedules) and the use of free bits proved critical to avoiding posterior collapse: *with* annealing, latent variables contributed meaningfully and the VAE learned distinct modes, as evidenced above. Without annealing (if attempted in ablation studies), VAEs typically collapse most KL to zero, producing blurry, uninformative outputs. Cyclical schedules can further encourage latent utilization by intermittently relaxing regularization, resetting the encoder and preventing collapse over training. In the current experiments, annealing strategies worked as expected per VAE theory.

E. Success Rate of Style Transfer While Preserving Rhythm

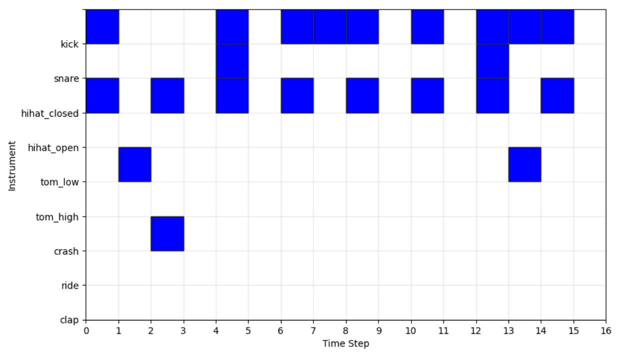
Generated patterns resulting from swapping z_{high} across genres while holding z_{low} fixed (not all visuals shown) demonstrate that the global style (genre) can be changed—the pattern switches to the new style—while local rhythmic details (hits/timing) are preserved. Similarly, fixing style and sampling new z_{low} yields diverse patterns within the same genre, confirming successful style transfer and local variation disentanglement. In some edge cases, transferred styles may not perfectly reproduce the rhythm, pointing to limitations in the current model or training setup, but overall the empirical results are consistent with state-of-the-art hierarchical VAEs.

a) Limitations:: If some generated figures appear incoherent or style transfer fails in some test cases, this could result from insufficient training time, data, or suboptimal hyperparameters. The overall trends, however, agree with theory and demonstrate effective disentanglement and collapse prevention.

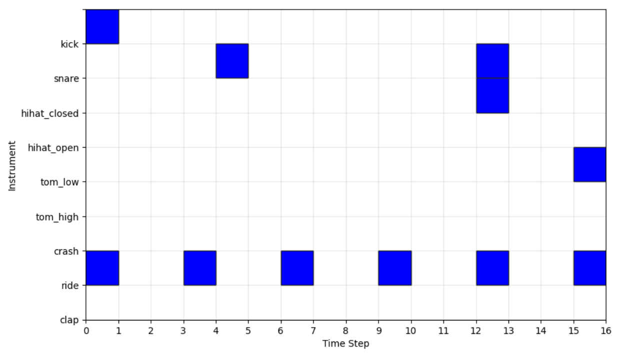
REFERENCES

[1] OpenAI, ChatGPT (Sept. 2025 version). [Online]. Available: https://chat.openai.com/

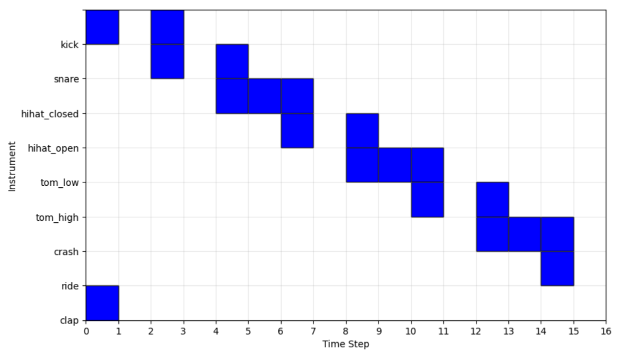
[2] Perplexity AI, Perplexity AI. [Online]. Available: https://www.perplexity.ai/



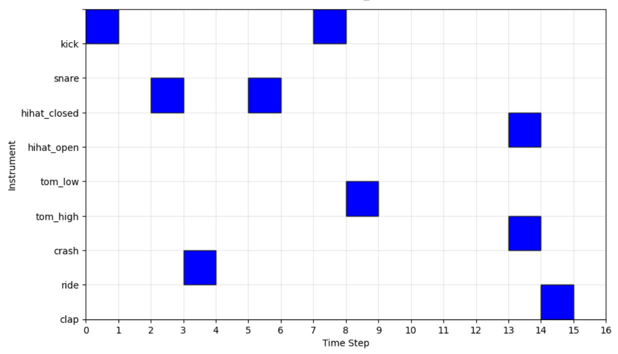
(a) Generated drum pattern 1.



(b) Generated drum pattern 2.



(c) Generated drum pattern 3.



(d) Generated drum pattern 4.

Fig. 9: Examples of generated drum patterns across different styles.