CS275P Final Project Report – Text summarization using VAE

Yumeng Zhang
University of California, Irvine
61928832
yumez14@uci.edu

Qiyu Zhang
University of California, Irvine
62367332
qzhuang4@uci.edu

Ran Ran
University of California, Irvine
30705408
ranr1@uci.edu

Zijie Li
University of California, Irvine
59608454
zijiel18@uci.edu

Zhiyan Tan
University of California, Irvine
29264828
zhiyant@uci.edu

1 Description of the problem and related Work

Automatic Text Summarization (ATS) refers to the process of condensing a long piece of text into a shorter version, while preserving its essential information and overall meaning. The goal of ATS is to create a coherent and concise summary that captures the key points and main ideas of the original text.

Chirantana Mallick (2019)(1) describes a graph-based text summarization method that captures the aboutness of a text document by using a modified TextRank algorithm, which is based on the concept of PageRank used for web pages. Samuel R. Bowman et al (2015)(2) introduced an RNN(Recurrent Neural Network)-based variational autoencoder (VAE) model that incorporates distributed sentence representations. they not only successfully applied VAE to generate well-formed text,but also explore many interesting properties of the model's latent sentence space. Liu and Liu (2019)(3) present a novel variational autoencoder (VAE) for natural text generation, utilizing a transformer-based architecture and augmenting the decoder with an LSTM (Long Short-Term Memory)networks based language model layer to better exploit latent variable information. LSTM networks were designed to overcome the limitations of traditional RNNs by effectively managing long-term dependencies through their specialized gating mechanisms. Their approach addresses training issues like KL divergence collapsing and model degradation, and experiments show that their method generates more meaningful and semantically coherent sentences.

However, due to our limited proficiency with the unsupervised VAE method, our model could not successfully generate text summaries. In our project, we explored a text-generation model LSTM-VAE, as well as a generative model based on using pre-trained Transformers as the encoder and decoder in a VAE. We studied various aspects of natural language processing, including the challenges of tokenizing and embedding, integrating different architectures, optimizing the training process, and improving the coherence and relevance of generated summaries. Our work involved experimenting with different strategies to mitigate issues such as KL divergence collapse and model degradation, ultimately aiming to enhance the quality and interpretability of the generated text. Instead, we focused on the task of generating coherent novel sentences that interpolate between known sentences.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Through this exploration, we gained insights into the strengths and limitations of hybrid models in summarizing complex texts and the potential of advanced neural architectures in improving text generation tasks.

2 Model Architecture

2.1 LSTM-VAE summarization model

2.1.1 Model Architecture

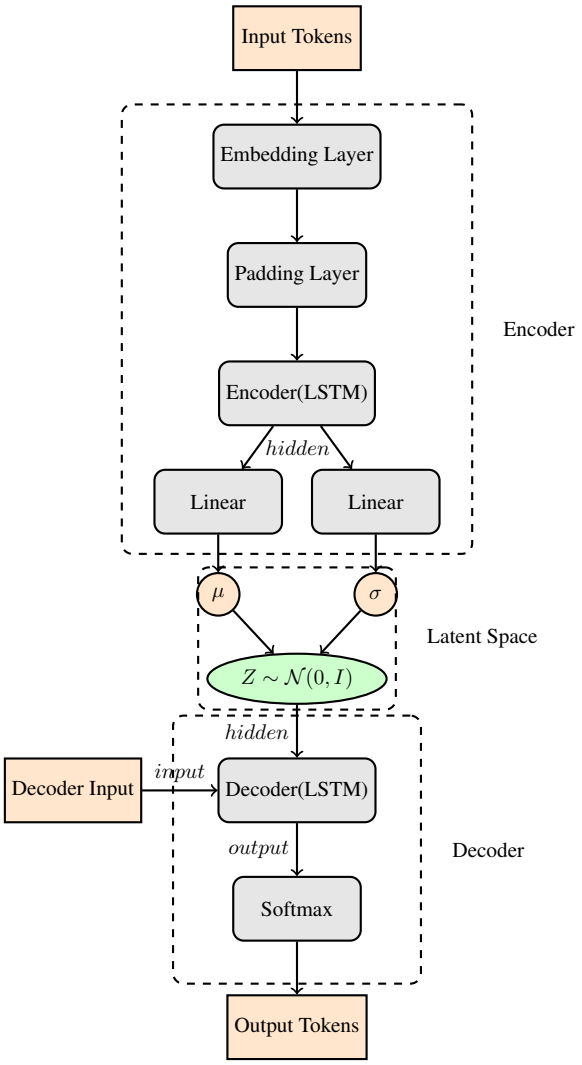

Figure 1: Model architecture of LSTM-VAE

In the training process, we split the summary and article texts into sentences, which serve as the input sequences (denoted as X_1). We train a VAE (Variational Autoencoder) model, aiming to capture global representations of connections of words in the latent space (denoted as $\mathbf{z}$). The decode sequence (denoted as X_2) is the right-shifted version of the input.

In this context:

- **Encoder**: The encoder is responsible for mapping the input sequence (X_1) into a latent representation $\mathbf{z}$. It compresses the information in X_1 into a lower-dimensional space, capturing the essential features and structure of the sentences. The approximate posterior

distribution $q(\mathbf{z} \mid X_1)$ is modeled as a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, where μ and σ are parameters derived from X_1.

$$q(Z \mid X_1) = \mathcal{N}(\mu(X_1), \sigma^2(X_1))$$

- **Decoder**: The decoder takes the latent representation (Z) and reconstructs the sequence (X_2), which is the right-shifted version of the original input(X_1) . It attempts to generate the output sequence based on the information stored in Z and (X_2), effectively learning to recreate the sequential patterns of the words.

$$p(x_t \mid \mathbf{z}) = p(x_t \mid \mathbf{z}, x_{t-1})$$

This model struggles to effectively learn global latent representations due to the VAE's tendency to ignore the latent variable Z and rely solely on the LSTM decoder, leading to a KL divergence term of zero. This phenomenon is known as posterior collapse. We use two techniques to address this issue by weakening the ability of LSTM decoder.

2.1.2 KL cost annealing

Typically, the cost function in a VAE consists of two main terms: Reconstruction Loss: Measures how well the decoder can reconstruct the input data from samples drawn from the latent space. KL Divergence: Measures the divergence between the learned latent distribution and a prior distribution (usually a standard normal distribution).

- **Reconstruction Loss**:

$$\text{NLL\_loss} = -\sum_{t=0}^{T} \log p(y_t \mid x_t) \tag{1}$$

  Where, $y_t$is the generated token for the t step, and x_t is the original token for the t step.

- **KL Divergence**:

$$\text{KL\_loss} = -\frac{1}{2}\sum_{i=1}^{D}\left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2\right) \tag{2}$$

  where, D represents the dimensional of the latent space, and μ_i and σ_i represent the mean and standard deviation of the latent distribution of thei^{th} dimension.

So we introduce a dynamic weighting of the KL divergence term within the cost function during training. Initially, this weight is set to zero, allowing the model to focus solely on maximizing information encapsulated in z. As training advances, we incrementally raise this weight, compelling the model to refine its encodings and align them with the underlying prior distribution. This gradual increase continues until the weight reaches 1, achieving alignment with the true variational lower bound in the weighted cost function.

$$\text{loss} = (\text{NLL\_loss} + \text{KL\_weight} \times \text{KL\_loss}) \tag{3}$$

And, KL_weight can be calculated as in (4):

$$\text{KL\_weight} = \frac{1}{1 + \exp(-k \cdot (\text{step} - x0))} \tag{4}$$

2.1.3 Word dropout

Another natural way to weaken the encoder is to reduce the information provided by the decoder input sequence. In other words, we can reduce the amount of conditioning information it receives during training. This can be achieved by randomly omitting a certain portion of the words that the decoder would typically use for conditioning. This process involves randomly selecting a fraction of the words that the decoder relies on and replacing them, which weakens the decoder's ability to depend on those words for generating outputs. Therefore, push the model to store more information in the latent space.

2.2 Pre-trained Transformer VAE Generation Model

To address the issue of posterior collapse and manage longer inputs such as articles, we modified the encoder and decoder of our VAE model to utilize pre-trained transformer structures. Pre-trained transformer models, such as BERT, have demonstrated exceptional performance in capturing contextual information and understanding long-range dependencies, making them suitable for our task.

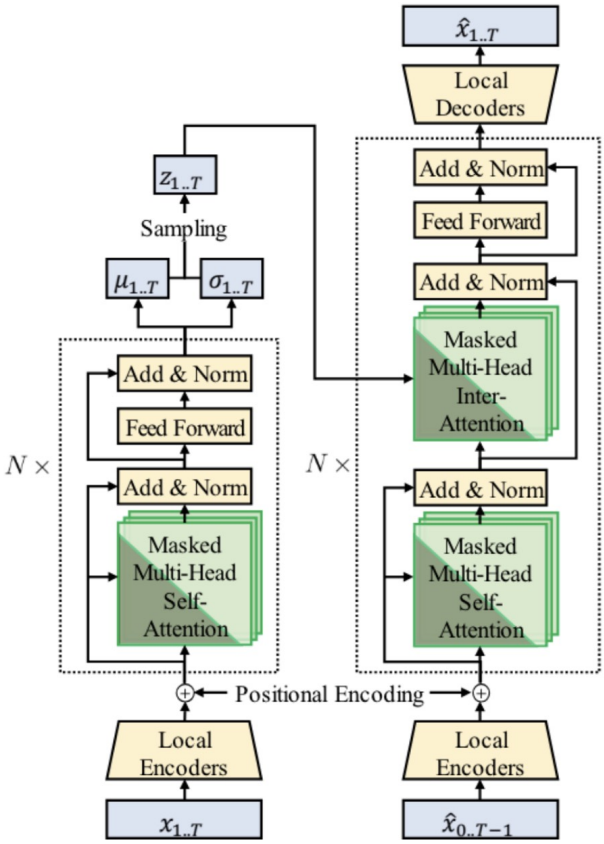

Figure 2: Model architecture of Transformer-VAE(4)

We employed the Hugging Face `EncoderDecoderModel` class as the primary architecture for our model, initializing it with pre-trained 'bert2bert' model checkpoints. This choice allows us to leverage the rich semantic representations learned from large-scale text corpora, thereby enhancing the model's ability to generate coherent and contextually relevant sequences.

- **Encoder** In our model, the encoder is based on a pre-trained BERT model. We utilize the encoder to process the input sequence, capturing its contextual information. The encoder's hidden states are then mapped to the parameters μ and σ using pooling and linear projection layers. These parameters are used to define the latent variable $\mathbf{z}$, which is reparameterized as the original "encoder hidden state".
- **Decoder** The decoder in our model is also based on a pre-trained BERT model but needs cross-attention layers and makes use of causal masking for auto-regressive generation. The original inputs for a transformer decoder are the 'encoder hidden state' and the input sequence. In our case, we substitute 'encoder hidden state' with $\mathbf{z}$ and keep everything else the same. During decoding, each token in the sequence performs cross-attention with the same $\mathbf{z}$, ensuring that the generated output maintains coherence and relevance to the input context.

Notably, rather than creating a latent variable $\mathbf{z}$ for each token, we used a global $\mathbf{z}$ to encode the semantic information for each input sequence. This approach reduces the complexity of the model and helps maintain the semantic integrity of the input sequence during generation. The model structure is illustrated in Figure 2.

In this model, we use CrossEntropy Loss instead of Negative Log-Likelihood loss.

$$\text{CrossEntropyLoss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\log p(y_{i,t} \mid x_{i,t}) \tag{5}$$

Where N is the number of samples, $y_{i,t}$ is the generated token for sample i at time step t, and $x_{i,t}$ is the original token for sample i at time step t.

3 Experiments

We have two different experient setting, using sentence level dataset and article level dataset, both LSTM-VAE and Transformer VAE are experimented under these two setting. To evaluate our model, we test our models with reconstruction and inference tasks. The whole process is shown in Figure 3. The parameters setting for the LSTM-VAE model are shown in Table 1. The parameters setting the Transformer-VAE model are shown in Table 2.

3.1 Pre-processing

In this experiment, we use the BBC news dataset from Kaggle [1] as the benchmark. The dataset provides 2224 original news articles covering five categories, and each article is accompanied by a reference summary.

Since we are using a relatively small dataset with around 2,000 articles and 40,000 sentences, it is reasonable to enhance our model's performance by cleaning the dataset. For example, many of these news articles contain numerical values that may be split into different parts. To address this, we replaced entire numerical sequences, such as "1.23 billion," with a placeholder "N" to prevent the splitting of these numbers. This preprocessing step ensures that the numerical values are treated as a single entity, which helps the model better understand and retain the contextual meaning of the text. Additionally, we removed all punctuation marks from the dataset. Cleaning the dataset in this manner helps reduce noise and improves the overall quality of the training data, leading to more accurate and reliable model performance.

Additionally, the cleaned data is further tokenized by Twitter Tokenizer and Bert Tokenizer for LSTM-VAE and Transformer-VAE respectively.

Table 1: Parameters For LSTM-VAE

Parameter	**Value**
Batch Size	32
Learning Rate	0.001
Embedding Size	300
Hidden Size	256
LSTM Layers	2
Latent Size	32
Word Dropout	0.5

Table 2: Parameters For Transformer-VAE

Parameter	**Value**
Batch Size	4
Learning Rate	0.0001
Embedding Size	512
Hidden Size	768
Latent Size	32

3.2 Training Process

We trained the VAE-LSTM for 10 epochs, and the values of training loss throughout the training process are shown in the Figure 4.

The NLL loss decreases over time, suggesting that the model is getting better at predicting the target values. The initial high values and subsequent reduction indicate the model's learning phase. The

[1] Kaggle BBC news summary: https://www.kaggle.com/datasets/pariza/bbc-news-summary

KL loss starts high and decreases rapidly, which is expected as the KL weight starts from zero and gradually increases. This shows that the model is learning to align the latent space with the prior distribution effectively over time. The plot confirms the gradual increase in the KL weight from zero to one, with a smooth increase indicating a well-implemented annealing function. This function helps in balancing the focus between NLL and KL losses during training.

The Transformer-VAE is also trained for 10 epochs, the training KL divergence and cross-entropy loss is in Figure 5 6.The KL divergence plot suggests convergence after initial fluctuations, while the entropy plot shows a steady decline, implying increasing model certainty and improved latent space representation over time.

3.3 Inference

The inference process for both the Transformer-VAE and LSTM-VAE models is identical. We begin by randomly initializing the latent variable z and feeding it to the decoder along with a start token ([101]). The entire generation process operates in a loop, generating tokens until either a maximum length (e.g., 512 tokens) is reached or an end-of-sequence token is encountered. For each token, the model predicts the next token, applies a softmax function to obtain probabilities for each possible token, samples the next token based on these probabilities, and appends it to the generated sequence. Finally, the generated sequence is decoded into a string, representing the generated article.

3.4 Reconstruction

The reconstruction process for both the Transformer-VAE and LSTM-VAE models is as follows. We begin by feeding the input sequence into the model. The encoder processes this input sequence and encodes it into the latent variable z. This latent variable z is used to generate the hidden state of the decoder. The input to the decoder is a version of the input that has been partially dropped out.

We conducted two comparative experiments: document-level and sentence-level generation. In the document-level experiment, the input is the entire article, and the output is the generated article. In the sentence-level experiment, the input is individual sentences, and the model generates sentences which are then combined to form the article. The reconstruction results are in Table 3.

4 Result Analysis

4.1 LSTM-VAE

4.1.1 Inference

As shown in the Table 3, the sentence-level model outperforms the document-level model and is able to generate grammatically correct sentences.

Table 3: Sentence-level Samples for LSTM VAE

Sentence
it is not a good job . <eos>
the n is not a fan for the <unk> . <eos>
the lib dems say the lib dems would be a parody of the election and the tories have been asked . <eos>
the company said the organizational was not intended to be a huge factor in the us . <eos>
the prime minister said the government was not a good issue of the leadership and the <unk> . <eos>

4.1.2 Reconstruction

We compared the reconstructed article with the original ones and calculated the rouge scores between them. The results are shown in Table 4.

Here is a reconstruction sample of the sentence in BBC dataset from the sentence-level LSTM-VAE model.

Table 4: ROUGE Scores for Document and Sentence Level Generation in LSTM-VAE

Generation Type	Metric	Precision	Recall	F-measure
Document(LSTM)	rouge1	0.184	**0.171**	0.162
	rouge2	0.021	0.018	0.017
	rougeL	0.166	**0.153**	0.146
Sentence(LSTM)	rouge1	**0.318**	0.141	**0.194**
	rouge2	**0.074**	**0.035**	**0.047**
	rougeL	**0.293**	0.130	**0.179**

The input sentence: while the majority of any salary costs may be covered by the government statutory pay recruitment costs advertising costs retraining costs and the strain on the company will not be he said.

The reconstructed sentence: the company said the organizational industry would be a gentle nudge to the company and the company said it would be a flavour.

While the reconstructed sentence could benefit from improvements in coherence and relevance, its language style does share some similarities with the original dataset. These similarities are evident in the formal tone, the organizational context, the introduction of multiple concepts, and the use of attribution. Due to the inferior results and excessive length of the document-based LSTM-VAE model, the detailed results are available in the GitHub repository.

4.2 Transformer-VAE

4.2.1 Inference

Similar to the LSTM-VAE model, the Transformer-VAE trained on a sentence-level dataset yields promising results. The grammar is noticeably better when the model processes individual sentences rather than entire articles. Although the Transformer-based VAE did not fully meet our expectation of perfectly handling whole articles as single inputs, we observed that the pre-trained encoder-decoder structure produces sentences with better grammar and more consistent meaning compared to the non-pre-trained LSTM-VAE.

The reason why the LSTM still struggles with article-length inputs is likely due to the input sequence being too long; the maximum length of each article exceeds the longest sequence that BERT can handle. Also, a largar dataset might be needed to finetune our model with this long sentences. The results are shown in Table 5.

Table 5: Sentence-level Samples for Transformer VAE

Sentence
in the second half we did not play with enough speed. <eos>
indoor parking may be costly but it is not always secure. <eos>
singer sam endicott said he felt great about coming top of the sound of n list. <eos>
a spokesman said they have no comment to make. <eos>
currently the private sector was only spending n a year on seeking an inoculation and the market needed boosting mr brown said. <eos>

4.2.2 Reconstruction

Here is a sample for Transformer-VAE model.

The input sentence: the orderly reversal of the deficit is a major challenge for policy makers in both the united states and other economies it noted.

The reconstructed sentence: during his time in charge of the national side he never surprised me that he would have a chance in the major and national side.

While the model excels at generating grammatically correct sentences from a randomly sampled $\mathbf{z}$, it struggles to accurately reconstruct sentences fed into the encoder as shown in the Table 6. This discrepancy highlights a key limitation in the current implementation of the Transformer-VAE.

One possible reason for this issue is the inherent complexity of the text data and the variability in sentence structures. The model might be learning to generate plausible sentences that fit the general distribution of the training data, but it may not be effectively capturing the detailed nuances required for precise sentence reconstruction.

Another factor could be the information bottleneck introduced by the latent variable $\mathbf{z}$. Although $\mathbf{z}$ encodes semantic information, it may not retain sufficient detail to reconstruct specific input sentences accurately. This suggests that the latent space might be too compressed, causing the loss of crucial information necessary for sentence reconstruction.

Additionally, the pre-training of the encoder-decoder structure primarily focuses on generating meaningful and contextually accurate text, which might not directly translate to the ability to reconstruct specific inputs. The model's training process may need to be adjusted to place more emphasis on reconstruction accuracy, possibly by incorporating additional loss functions or training strategies that prioritize this aspect.

Overall, while the Transformer-VAE demonstrates strong generative capabilities, further refinement is needed to enhance its performance in sentence reconstruction tasks. This could involve experimenting with different model architectures, tuning hyperparameters, or employing more sophisticated training techniques to improve the fidelity of sentence reconstruction.

Table 6: ROUGE Scores for Document and Sentence Level Generation in Tranformer-VAE

Generation Type	Metric	Precision	Recall	F-measure
Document(Transformer)	rouge1	0.218	0.203	0.295
	rouge2	**0.03**	0.02	0.025
	rougeL	**0.38**	**0.28**	0.178
Sentence(Transformer)	rouge1	**0.250**	**0.213**	**0.310**
	rouge2	0.028	**0.025**	0.024
	rougeL	0.223	0.234	**0.228**

As shown in Table 4 6, the results from sentence-level generation are noticeably better than those from document-level generation. This observation can be explained by the limitations in the model's ability and the scale of the dataset. Specifically, the LSTM-VAE approach struggles to capture contextual information from relatively long articles, and the Transformer-VAE also faces challenges with limited size datasets. In contrast, sentence-level generation allows the model to more easily learn and utilize information from the inputs.

5 Implementation

Our project is written in Python using the Pytorch structure.

The GitHub repositories we referenced are listed below:

`https://github.com/timbmg/Sentence-VAE`
`https://github.com/rohithreddy024/VAE-Text-Generation`

In the **VAE-LSTM** model, we reused code from the repository and modified it to create two models: one for the article level and another for the sentence level. We implemented the Transformer Encoder-Decoder based on code from the Hugging Face repository to encode the input sequence. All codes are in our own GitHub repository.

`https://github.com/SlowlyRan/CS275textsummary`
`https://github.com/zickli/Sentence-VAE`

For the **Transformer-VAE**, we based our model on the structure proposed in (4), which is relatively simple and easy to implement. Given our limited computing resources and the fact that we are dealing

with text data, we adapted the fine-tuning method of large language models for the transformer-based VAE, as discussed in (3). Furthermore, the entire codebase for the Transformer-VAE was developed from scratch.

6 Conclusion

In our initial experiments, we developed a standard LSTM-VAE model where the encoder processed an entire article, and the decoder reconstructed the article based on the previous time step x_{t-1} to generate the current time step x_t. However, as training progressed, we encountered a challenge: the KLD loss quickly decreased to 0, indicating that the latent variable z was unable to extract the semantic information from the article. This resulted in the model behaving more like a conventional LSTM language model, lacking the ability to effectively encode the full content of sentences and abstract global features in z.

To address this issue, we introduced variations in the kld_weight and $word\_dropout$ parameters to mitigate posterior collapse and encourage z to encode more semantic information. Despite these efforts, we found that directly using an entire article as input made it difficult for the model to learn complex semantic patterns within the article.

To enhance the model's ability to capture complex patterns, we adopted a new approach: we segmented the article into sentences and developed a model that takes individual sentences as input and generates corresponding sentences. By reconstructing the article from these generated sentences, we aimed to enable the model to capture and reproduce the intricate semantic patterns present in the original article.

Another approach we explored was using a Transformer-based VAE, leveraging the ability of the pre-trained Transformer to process long sequences, capture contextual information. Compared to the LSTM-VAE, the Transformer-VAE performed better on sentence-level data, producing sentences with superior grammar and consistency. However, both models faced challenges with article-length inputs due to BERT's sequence length limitations. The pre-trained encoder-decoder structure improved the grammatical quality of generated sentences, but enhancements are needed for better reconstruction fidelity. Future work will focus on refining model architectures, tuning hyperparameters, and adopting advanced training techniques.

7 Contribution

Yumeng Zhang: Did research on transformer-based VAE modelling, posterior collapse. Implemented an Transformer-based VAE model.

Qiyu Zhang: Run comprehensive experiments of transformer-based VAE model, analized and visualized experiment results.

Ran Ran: Managed the project's documentation and reporting. and documented the LSTM-VAE model, detailing the model structure, methodology, experiment design.

Zijie Li: Implemented a sentence-level LSTM-VAE and completed experiments.

Zhiyan Tan: Implemented an article-level LSTM-VAE and completed experiments.

References

[1] T. Wang and X. Wan, "T-cvae: Transformer-based conditioned variational autoencoder for story completion," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 5233–5239, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[3] D. Liu and G. Liu, "A transformer-based variational autoencoder for sentence generation," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, 2019.

[4] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 516–520, IEEE, 2020.

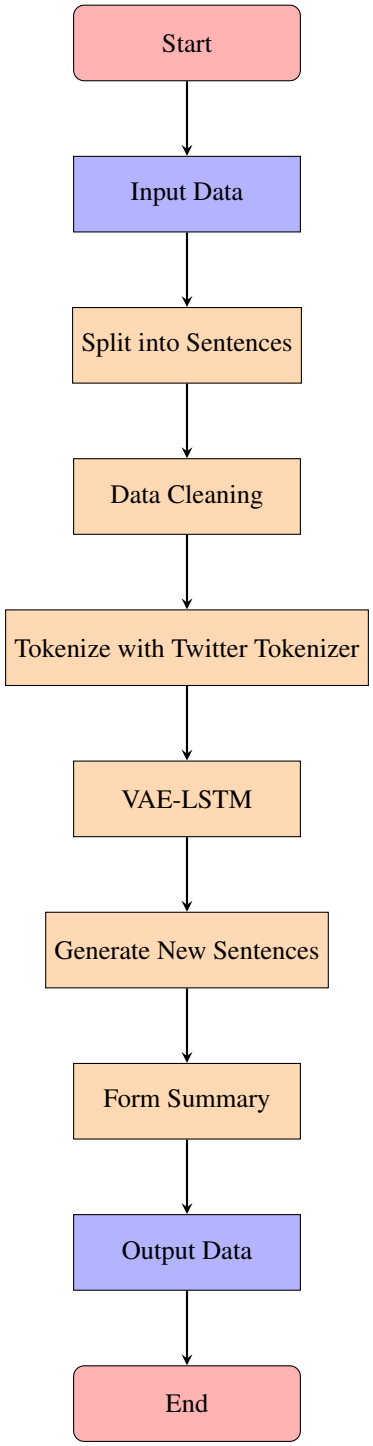

Figure 3: Flow chart of the reconstruction process with LSTM-VAE model

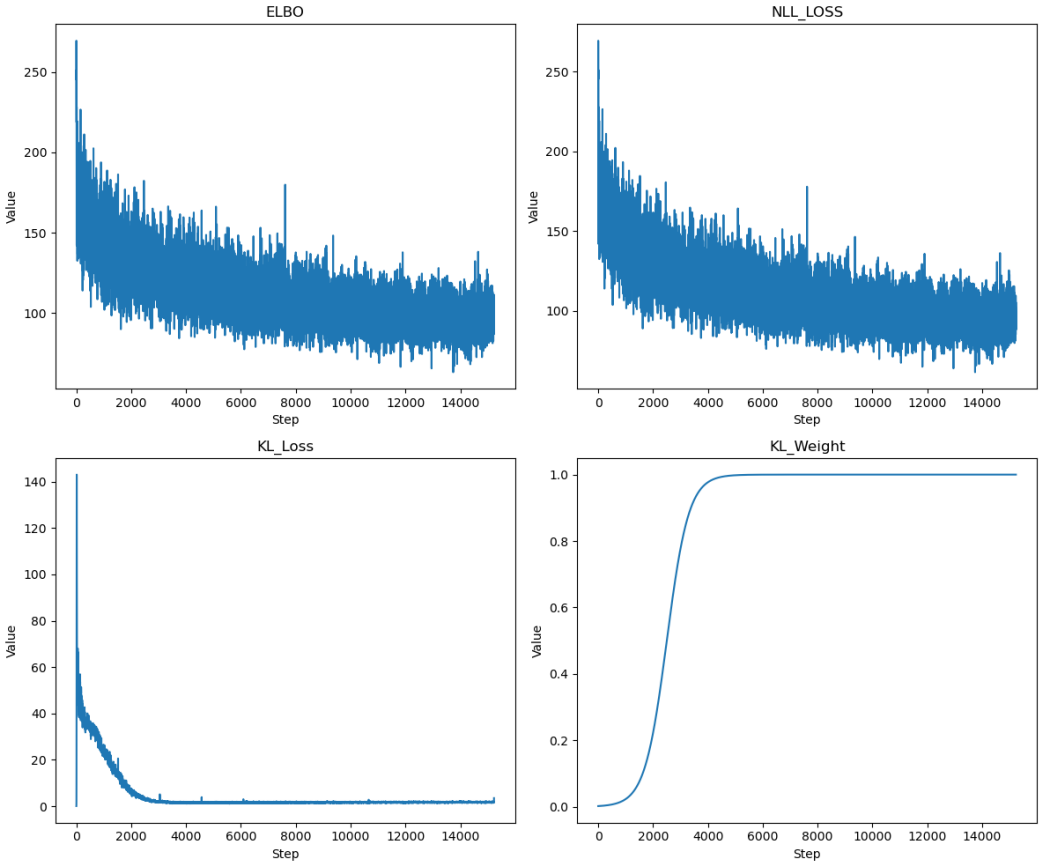

Figure 4: Training loss variation over steps (LSTM-VAE)

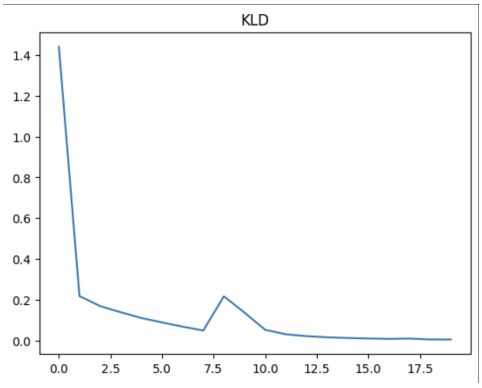

Figure 5: KLD in Transformer VAE

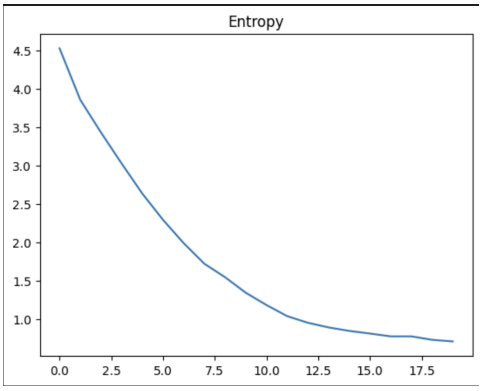

Figure 6: Cross Entropy in Transformer VAE