Submitted to *the INFORMS 2024 Data Challenge Competition*

FOR INTERNAL USE ONLY (leave blank)
Submission Number:
Team Number:

Deep Learning for TikTok Video Popularity Prediction: An Variational Inference Perspective

Jie Wang (**Corresponding Author**)
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, jwang3163@gatech.edu

Yongchun Li
Department of Industrial and Systems Engineering, University of Tennessee Knoxville, Knoxville, TN 37996, ycli@utk.edu

We propose a deep learning framework for predicting the popularity of TikTok videos, as illustrated in Figure 1. Our approach jointly trains a recognition model (encoder) that maps video features into a deep latent space, and a generative model (decoder) that reconstructs popularity metrics from the latent variables. Both the encoder and decoder are implemented using deep neural networks. Numerical experiments demonstrate the effectiveness of our method, which secured first place during the first three weeks of the testing phase and second place in the final week.

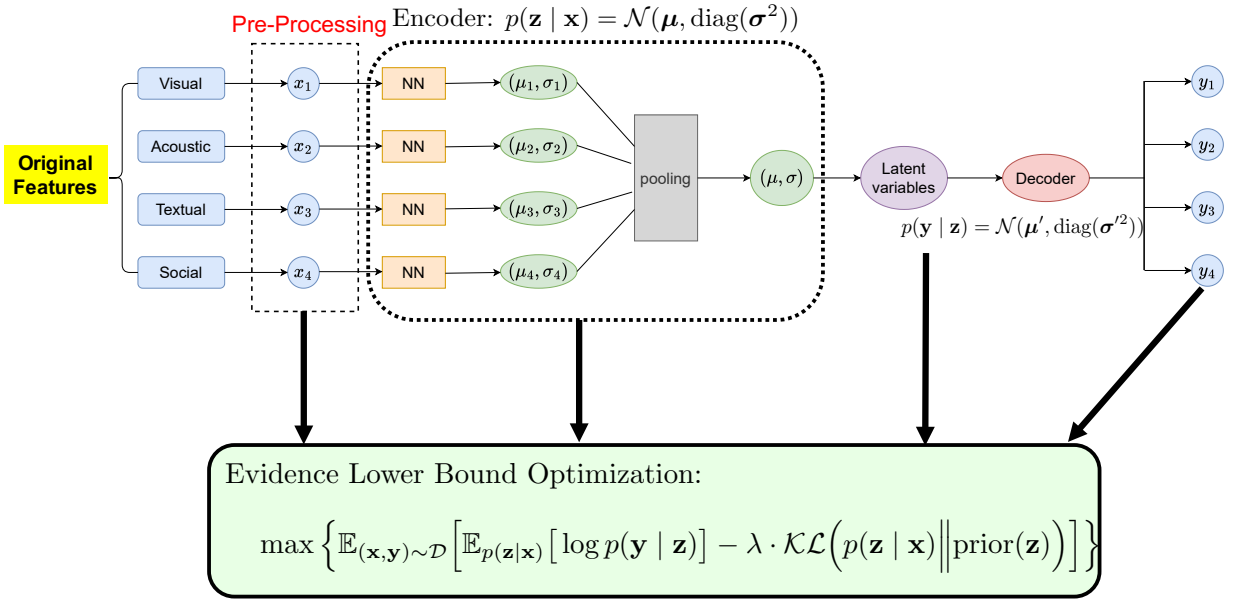

Figure 1 **Diagram of our framework, which learns stochastic mappings between an observed feature space (x-space), latent z-space, and observed label space (y-space).**

1. Introduction

Would you determine TikTok video popularity by tossing a coin? Replacing deterministic metrics with a random decision may seem irrational, yet introducing stochastic outputs for popularity prediction offers several advantages. First, feature components that are irrelevant to the prediction task can be treated as stochastic noise. Second, stochastic outputs are valuable for quantifying the uncertainty of predictions. Motivated by these insights, we adopt a variational inference approach combined with deep learning to provide stochastic estimators for the prediction task. The key ideas are illustrated in Figure 1.

2. Methodology

Let $\mathcal{D}$ represent our training dataset, given by $\big\{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^n, \mathbf{y}^n)\big\}$, where $\mathbf{x}^i$ denotes the features associated with the i-th video, and $\mathbf{y}^i \in \mathbb{R}^4$ represents the corresponding popularity metrics (views, likes, comments, and shares). The feature set $\mathbf{x}^i$ is divided into four components: (1) visual content, (2) acoustic content, (3) textual description, and (4) social influence, denoted as $\mathbf{x}^i = (\mathbf{x}^i_j)_{j=1}^4$. We preprocess these features, with detailed implementation described in Section 2.2. Given $\mathcal{D}$, we utilize the variational auto-encoder framework [2] to learn a latent variable $\mathbf{z}$ that aids in predicting the popularity metrics. The relationship among $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ is modeled as $\mathbf{x} \to \mathbf{z} \to \mathbf{y}$. By the rules of probability, this gives $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{z} \mid \mathbf{x})p(\mathbf{y} \mid \mathbf{z})$. Here, we define $p(\mathbf{z} \mid \mathbf{x})$ as the *encoder*, which maps features to the latent space, and $p(\mathbf{y} \mid \mathbf{z})$ as the *decoder*, which maps the latent variable to popularity metrics, approximating the ground truth. Both probability models are parameterized using neural networks with parameters θ and ϕ, respectively.

2.1. Formulation

A direct but intractable goal for our problem is to maximize the regularized log-likelihood model:

$$\max_{\theta,\phi} \ \Big\{\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\log p_{\theta,\phi}(\mathbf{x},\mathbf{y}) - \lambda \cdot \mathcal{KL}\Big(p_\theta(\mathbf{z} \mid \mathbf{x})\Big\|p_0(\mathbf{z})\Big)\Big]\Big\}, \tag{Ideal}$$

where the second term represents the KL-divergence between the encoder and the *uninformative prior* $p_0(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. It peanlizes over too complex encoder mapping. Unfortunately, the

log-likelihood function for the first part is challenging to compute, as evaluating the marginal distribution $p_{\theta,\phi}(\mathbf{x},\mathbf{y}) = \int_z p_{\theta,\phi}(\mathbf{x},\mathbf{y},\mathbf{z})\mathrm{d}z$ requires high-dimensional integeration. To tackle this challenge and inspired from conventional Bayesian inference literature, we replace the optimization objective with the *evidence lower bound*:

$$\max_{\theta,\phi} \ \Big\{\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\big[\log p_\phi(\mathbf{y}\mid\mathbf{z})\big] - \lambda\cdot\mathcal{KL}\Big(p_\theta(\mathbf{z}\mid\mathbf{x})\Big\|p_0(\mathbf{z})\Big)\Big]\Big\}. \tag{1}$$

In the following, we discuss key components of our optimization problem.

Neural Network Encoder. We build the encoder model $p_\theta(\mathbf{z}\mid\mathbf{x}) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu},\mathrm{diag}(\boldsymbol{\sigma}^2))$, where the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ depend on feature $\mathbf{x}$ and weight θ. Let us assume the *conditional independence* of features from distinct parts, then $p_\theta(\mathbf{z}\mid\mathbf{x}) \propto p_0(z)\prod_{j=1}^4 p_\theta(\mathbf{z}\mid\mathbf{x}_j)$, and consequently, we build

$$p_\theta(\mathbf{z}\mid\mathbf{x}_j) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}_j,\mathrm{diag}(\boldsymbol{\sigma}_j^2)), \qquad (\boldsymbol{\mu}_j,\log\boldsymbol{\sigma}_j) = \mathrm{EncoderNeuralNet}_\theta(\mathbf{x}_j). \tag{2}$$

We recover $p_\theta(\mathbf{z}\mid\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu},\mathrm{diag}(\boldsymbol{\sigma}^2))$ with

$$\boldsymbol{\mu} = \sum_{j=1}^4 \Big[\frac{\boldsymbol{\mu}_j \odot \frac{1}{\boldsymbol{\sigma}_j^2}}{\sum_{j=1}^4 \frac{1}{\boldsymbol{\sigma}_j^2}}\Big], \qquad \boldsymbol{\sigma}^2 = \frac{1}{\sum_{j=1}^4 \frac{1}{\boldsymbol{\sigma}_j^2}}, \tag{3}$$

where $\odot$ and $\frac{1}{\boldsymbol{\sigma}}$ are operated element-wisely, and we name the operation above as *pooling*.

Neural Network Decoder. Similar to the encoder setting, we model the decoder $p_\phi(\mathbf{y}\mid\mathbf{z}) = \mathcal{N}(\mathbf{y};\boldsymbol{\mu}',\mathrm{diag}(\boldsymbol{\sigma}'^2))$, where the parameters $(\boldsymbol{\mu}',\log\boldsymbol{\sigma}') = \mathrm{DecoderNeuralNet}_\phi(\mathbf{z})$.

Optimization. Iteratively, we obtain the stochastic gradient estimator of (1) with respect to (w.r.t.) θ and ϕ, and then perform the gradient update. We terminate the algorithm until convergence. Let us take the first term in (1) to discuss how to compute its gradient as an example, as the second term can be handled similarly. The unbiased gradient w.r.t. ϕ can be obtained easily because

$$\nabla_\phi\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\big[\log p_\phi(\mathbf{y}\mid\mathbf{z})\big]\Big] = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\big[\nabla_\phi\log p_\phi(\mathbf{y}\mid\mathbf{z})\big]\Big].$$

However, the unbiased gradient w.r.t. θ is challenging to derive as $\nabla_\theta\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\cdot] \neq \mathbb{E}_{\nabla_\theta p_\theta(\mathbf{z}|\mathbf{x})}[\cdot]$. W apply the reparameterization trick to express $\mathbf{z}\sim p_\theta(\mathbf{z}\mid\mathbf{x})$ as $\mathbf{z} = g(\boldsymbol{\epsilon},\theta,\mathbf{x})$, where the distribution of random variable $\boldsymbol{\epsilon}$ is independent of $\mathbf{x}$ and θ. In detail,

$$\mathbf{z} = g(\boldsymbol{\epsilon},\theta,\mathbf{x}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\odot\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}).$$

Then we can estimate the gradient w.r.t. θ in the following:

$$\nabla_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\big[\log p_\phi(\mathbf{y}\mid\mathbf{z})\big]\Big] = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\nabla_\theta \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\big[\log p_\phi(\mathbf{y}\mid\mathbf{z})\big]\Big]$$
$$= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\nabla_\theta \mathbb{E}_{\boldsymbol{\epsilon}}\big[\log p_\phi(\mathbf{y}\mid g(\boldsymbol{\epsilon},\theta,\mathbf{x}))\big]\Big] = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\Big[\mathbb{E}_{\boldsymbol{\epsilon}}\big[\nabla_\theta \log p_\phi(\mathbf{y}\mid g(\boldsymbol{\epsilon},\theta,\mathbf{x}))\big]\Big].$$

Prediction and Uncertainty Quantification. After training, for a given new feature vector $\mathbf{x}$, we first pass it through the encoder $p_\theta(\mathbf{z}\mid\mathbf{x})$ to obtain the latent variable $\mathbf{z}$. This latent variable is then mapped through the decoder $p_\phi(\mathbf{y}\mid\mathbf{z})$ to generate the predicted output $\mathbf{y}$. We repeat this process for 10^3 independent trials, using the sample mean of the outputs as our point estimate, and construct error bars for the predicted output based on a 95% confidence interval.

2.2. Data Pre-Processing

Visual Content. For a given visual content, we first perform downsampling to extract 10 frames at fixed time intervals. From each frame, we extract a feature vector using the ResNet50 convolutional neural network, pretrained on the ImageNet dataset, which has achieved notable success in computer vision tasks. The final feature output is obtained by combining these 10 feature vectors.

Acoustic Content. Acoustic content provides complementary information to the visual component of the video. For feature representation, we use the widely adopted Mel-Frequency Cepstral Coefficients (MFCC) [3]. Specifically, we compute a 48-dimensional acoustic feature vector, consisting of a 24-dimensional mean vector and a 24-dimensional variance vector of the MFCC features.

Textual Content. Users typically attach brief descriptions to their videos. We collect the text from all videos and apply contrastive learning [4] to capture the latent space of the textual content. This process results in a 20-dimensional textual feature vector.

Social Content. The popularity of each video is significantly influenced by both the publisher and its release date. To capture these factors, we construct the following features for the social content: author ID, number of followers, number of people the author is following, total number of hearts (likes), total number of videos, and the time elapsed since the release date.

Post-Processing. For continuous-valued features, we normalize them to have zero mean and unit variance. We then extract the necessary features from four components to serve as input to our framework.

3. Numerical Results

Both the encoder and decoder networks are implemented as LeakyReLU neural networks with seven hidden layers, and training was completed over 144 RTX 4090 GPU hours. The following plots present point and interval estimates for the popularity metrics of each video in the testing dataset. Each data point is normalized by subtracting the predicted output from the true output, where red data points closer to the origin indicate better prediction performance. Purple shaded areas represent the constructed error bars, where the large area means we are less certain regarding the prediction outputs.

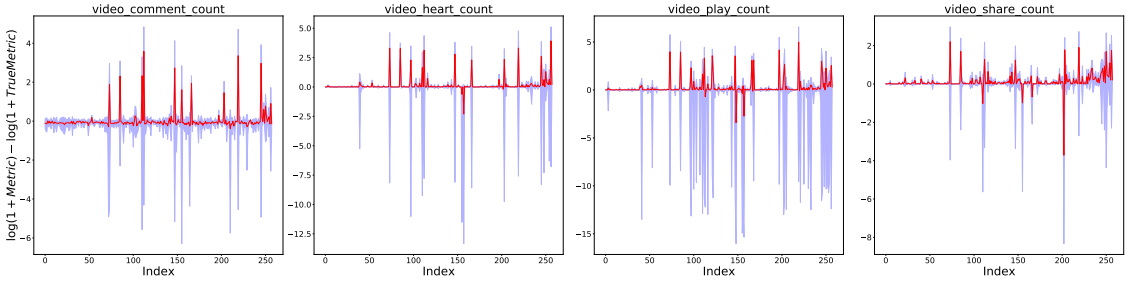

Figure 2 **Experiment results on testing dataset. Here the y-axis is normalizd to be $\log(1+\texttt{Metric})-\log(1+\texttt{Metric}_{\texttt{True}})$, where $\texttt{Metric}_{\texttt{True}}$ denotes the true output, and the x-axis denotes the index of data samples.**

4. Conclusion

We develop a deep learning-based variational inference approach for TikTok video popularity prediction, which not only yields superior performance, but also provides uncertainty quantification that estimates the prediction error. Its pratical performance on testing dataset has the second-highest rank, as validated by official rankings.

5. Team Members

- Jie Wang (Corresponding Author), School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, jwang3163@gatech.edu
- Yongchun Li, Department of Industrial and Systems Engineering, University of Tennessee Knoxville, Knoxville, TN 37996, ycli@utk.edu

References

[1] INFORMS. 2024. Submission template example. *Journal of XYZ*. 1(1).
[2] Kingma, Diederik P., and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12.4 (2019): 307-392.
[3] Li, Zhonghua, et al. Non-reference audio quality assessment for online live music recordings. *Proceedings of the 21st ACM international conference on Multimedia*. 2013.
[4] Qiu, Yao, Jinchao Zhang, and Jie Zhou. Improving gradient-based adversarial training for text classification by contrastive learning and auto-encoder. *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. 2021