# StyLLE: Style Learning and Latent Editing for Stylized Text and Speech Generation

**Jiadong Hao, Bohan Zhang , Yuchen Lu, Chengcheng Zhang, Kunda Yang**
`{haojd, bohanzh, yuchl, chengchz, kunda}@umich.edu`

## 1 Abstract

Large language models (LLMs) excel in NLP tasks but lack intrinsic style adaptability. To address this, we refine a representation editing approach called DRESS, which uses steering vectors to shift model activations for controllable style transfer. DRESS first identifies style-related attention heads via linear probing, then extracts a style subspace using SVD, and finally injects style-aware modifications into model activations during generation. We improve DRESS in three directions: (1) we automatically determine the optimal rank of each attention head's style subspace using BIC; (2) we propose a KNN-based local editing direction for fine-grained control; and (3) we accelerate inference by applying KV caching to both base and stylized forward passes. Beyond text, we extend this method to text-to-speech (TTS) synthesis using Spark-TTS, demonstrating that steering-based style editing is also effective for speech generation. Our empirical results show that these enhancements yield more efficient, accurate, and stylized outputs in both text and speech domains.

## 2 Introduction

Large language models (LLMs), such as GPT-4 [1] and LLaMA-3 [2], have demonstrated exceptional performance in natural language processing (NLP) tasks. However, these models lack the intrinsic ability to adapt their language style, which encompasses aspects such as tone, personality, emotion, and historical context [3].

Stylized responses are essential in numerous applications where user engagement and immersion rely heavily on linguistic personality. For example, role-playing NPCs in video games must maintain consistency with their designated dialect and tone, while virtual assistants in professional settings may need to adjust their level of formality dynamically. Without effective style adaptation, LLM responses risk sounding generic or out of place.

Currently, two primary approaches are used to generate stylized responses: prompting with few-shot demonstrations and fine-tuning. However, both methods have significant limitations. Prompting [4] often struggles to fully capture the nuances of a specific style, while fine-tuning [5] is computationally expensive. To overcome these challenges, DRESS-LLM [6] is proposed to control the style of LLM responses using representation editing [7], adding a steering vector to the LLM activations to shift the representations to another language style, which has shown great effectiveness and efficiency over other SOTA methods.

To fully explore the power of the steering vector editing method proposed by DRESS, our project is separated into two phases. In the first stage, we refine the representation editing pipeline through adaptive subspace selection, local direction estimation, and faster inference via KV caching. In the second stage, we extend this framework to the domain of text-to-speech (TTS) by integrating it with Spark-TTS [8], demonstrating that steering-based style editing can also effectively control vocal attributes such as pitch, speed, and timbre. Specifically, we define two contrasting vocal styles using Spark-TTS attribute settings and apply activation-based editing to shift model outputs from the original style toward the target style during inference, achieving fine-grained control without retraining. [1]

## 3 Related Work

### 3.1 Large Language Model Editing

Representation editing techniques have recently emerged as a lightweight, train-free approach for instilling specific behaviors and attributes into large language models (LLMs) by operating directly on their hidden representations during inference. Unlike conventional fine-tuning or prompt-based methods, representation editing avoids expensive parameter updates or long prompts by computing steering vectors that shift activations along attribute-relevant directions. In the context of stylized question answering, DRESS [6] exemplifies this paradigm by isolating a style subspace within attention head activations through attention head filtering and subspace denoising via singular value decomposition. For a given target style, DRESS calculates steering vectors in the identified subspace and injects them adaptively during generation, thus achieving flexible style control while preserving semantic integrity. This model editing framework will be detailed further in the following section. This approach leverages over-parameterization in LLMs to assume orthogonality between style and semantic components, enabling precise disentanglement with minimal interference [6].

### 3.2 Style Control of Text-to-Speech Generation

Existing approaches to style control in TTS can be broadly categorized into the following paradigms:

- **Reference-based methods**, which rely on exemplar audio to transfer prosody and timbre characteristics. These methods, such as Llasa [9], often suffer from instability due to sensitivity to the quality and content of the reference input.

[1]Phase 1 codes are available at: `https://github.com/rederyang/StyLLE`. Phase 2 codes are available at: `https://github.com/IUboyfriend/Sparktts_stylized`.

- **Attribute-based methods**, which explicitly modify prosodic features like pitch and speaking rate using flow-matching mechanisms or post-filtering techniques. For instance, CosyVoice2 [10] adopts flow-matching for prosody prediction, but this often leads to complex training pipelines and potential degradation in naturalness.

In contrast, our approach—integrating the DRESS framework with SparkTTS—provides a unified, lightweight solution. By injecting steering vectors into model activations along a learned style subspace, we enable fine-grained, dynamic style control without retraining. This zero-shot, modular framework is more interpretable than black-box methods and avoids reliance on reference audio, flow matching, or complex pipelines.

## 4 Method: Improved Style Editing Framework

### 4.1 Style Editing Framework of DRESS

The DRESS framework [6] aims to modify a pre-trained Large Language Model (LLM) $M$ to respond to a user query $q$ in a specific target style $S$, producing a stylized response $M'(q)$ that adheres to $S$ while preserving the original semantic meaning of $M(q)$. The target style $S$ is implicitly defined by a dataset of stylized examples, typically question-answer pairs $\{q_i, a_i^+\}_{i=1}^n$ where $a_i^+ \sim S$.

To achieve this without retraining, DRESS employs representation editing during inference. The core idea relies on constructing a paired dataset $D = \{q_i, a_i^+, a_i^-\}_{i=1}^N$, where $a_i^-$ represents the ordinary-style response to $q_i$ that is semantically equivalent to the target-style response $a_i^+$. This dataset is crucial for identifying style-specific activation patterns. The editing process involves adding calculated steering vectors to the internal activations within the Multi-Head Attention (MHA) blocks of the transformer architecture. The original DRESS framework follows a three-stage pipeline:

1. **Attention Head Filtering:** This stage identifies which attention heads are most relevant to the target style. For each head $h$ in layer $l$, the activation $u^{(h,l)} = \text{Attn}_h(x^{(l)})$ (the output of the attention mechanism before the output projection $W_h^O$) is extracted for both ordinary ($a_i^-$) and stylized ($a_i^+$) responses from the dataset $D$. A linear probing classifier $p(u^{(h,l)}) = \text{Sigmoid}(\langle \theta^{(h,l)}, u^{(h,l)} \rangle)$ is trained for each head to distinguish between the two styles based on $u^{(h,l)}$. The heads exhibiting the highest classification accuracy (top-$H$) are selected as style-relevant, forming a set $A$ of head-layer pairs $A = \{(h,l) | \text{head } h \text{ in layer } l \text{ is selected}\}$. Editing operations are only focused on these heads.
2. **Style Subspace Filtering:** Within the selected heads $(h,l) \in A$, DRESS aims to isolate a low-rank subspace that captures style variations while minimizing semantic interference. This is achieved by analyzing the activation differences between paired stylized and ordinary responses: $\delta u_i^{(h,l)} = u_i^{(h,l)+} - u_i^{(h,l)-}$ for each sample $i$ in the dataset $D$. These difference vectors are collected into a matrix $\Delta U^{(h,l)} = [\delta u_1^{(h,l)}, \dots, \delta u_N^{(h,l)}]^T \in \mathbb{R}^{N \times d}$, where $d$ is the activation dimension. Singular Value Decomposition (SVD) is applied to this matrix: $\Delta U^{(h,l)} = S^{(h,l)} \Sigma^{(h,l)} (V^{(h,l)})^T$. The top-$R$ right singular vectors $\{v_1^{(h,l)}, \dots, v_R^{(h,l)}\} \subset \mathbb{R}^d$, corresponding to the largest singular values, are chosen to form an orthogonal basis for the style subspace of head $(h,l)$, where $R$ denotes the rank of the subspace. The original DRESS heuristically sets the rank $R = 16$ for all selected heads.
3. **Adaptive Editing Strength:** Instead of applying a fixed steering vector, DRESS calculates an adaptive strength $\alpha_r^{(h,l)}$ for each basis vector $v_r^{(h,l)}$ (where $r$ indexes the basis vectors from 1 to $R$) at each generation step (i.e., for the current token's activation $u^{(h,l)}$). This strength is determined by:
   - A *global strength component* $\beta_r^{(h,l)}$, calculated as the projection of the mean activation difference $\overline{\delta u}^{(h,l)} = \frac{1}{N} \sum_{j=1}^N \delta u_j^{(h,l)}$ onto the basis vector $v_r^{(h,l)}$:
   $$\beta_r^{(h,l)} = \langle \overline{\delta u}^{(h,l)}, v_r^{(h,l)} \rangle$$
   - An *adaptive scaling factor* $\gamma_r^{(h,l)}$, which depends on the current token's activation $u^{(h,l)}$ and the mean activation of the target style samples $\overline{u}^{(h,l)+} = \frac{1}{N} \sum_{j=1}^N u_j^{(h,l)+}$:
   $$\gamma_r^{(h,l)} = \cos(\overline{u}^{(h,l)+} - u^{(h,l)}, v_r^{(h,l)})$$
   - An overall scaling hyperparameter $\lambda$.

   The final editing strength applied along basis vector $v_r^{(h,l)}$ is:
   $$\alpha_r^{(h,l)} = \lambda(1 + \gamma_r^{(h,l)})\beta_r^{(h,l)}$$

The complete steering vector for a selected head $(h,l) \in A$ at the current generation step is then $s^{(h,l)} = \sum_{r=1}^R \alpha_r^{(h,l)} v_r^{(h,l)}$. For heads not in $A$, $s^{(h,l)} = \mathbf{0}$. The final edited activation for the MHA block output in layer $l$ is computed as:

$$x^{(l+1)} = \text{MLP}\left( \bigoplus_{h=1}^{H_{total}} W_h^O \left( \text{Attn}_h(x^{(l)}) + s^{(h,l)} \right) \right)$$

where $H_{total}$ is the total number of heads in the layer. This process is applied dynamically during token generation, allowing the model $M'$ to produce responses in the target style $S$.

### 4.2 Head-wise Adaptive Style Subspace Determination

The DRESS framework posits that the stylistic variations captured by an attention head reside within a low-rank subspace of its activation space [6]. Editing is confined to this subspace, derived via SVD from the activation difference matrix $\Delta U^{(h,l)}$ (Sec. 4.1). Consequently, determining the appropriate dimensionality, or rank $R$, of this style subspace for each head is critical to the efficacy of the editing process.

The original DRESS framework employs a fixed, heuristically chosen rank (e.g., $Rank = 16$) uniformly across all selected attention heads $(h,l) \in A$. This assumes homogeneity in how different heads encode stylistic features. However, this assumption may be overly simplistic; attention heads likely exhibit heterogeneity in their functional specialization [11], suggesting that the intrinsic dimensionality required to capture style might vary significantly across heads. A fixed rank $R$ risks underfitting for heads encoding complex stylistic nuances (requiring $R > 16$) or overfitting for heads where style is represented more sparsely (requiring $R < 16$), potentially incorporating noise or irrelevant dimensions that could negatively impact semantic fidelity or fluency. This sensitivity motivates the need for a more principled approach to rank selection. To address this limitation, we propose an automatic, data-driven method to determine the optimal style subspace rank $R_{opt}^{(h,l)}$ individually for each selected attention head $(h, l) \in A$. Our approach leverages the Bayesian Information Criterion (BIC) [12], a widely used statistical criterion for model selection that balances model fit against model complexity.

Specifically, for each head $(h, l) \in A$, we first compute the SVD of its activation difference matrix: $\Delta U^{(h,l)} = S^{(h,l)}\Sigma^{(h,l)}(V^{(h,l)})^T$, where $\Sigma^{(h,l)}$ contains the singular values $\sigma_i^{(h,l)}$ in descending order. We then define a search range $[R_{min}, R_{max}]$ for the optimal rank. $R_{min}$ is set as the minimum rank $r'$ such that the cumulative explained variance ratio (CEVR) $\sum_{i=1}^{r'}(\sigma_i^{(h,l)})^2 / \sum_{j=1}^{d}(\sigma_j^{(h,l)})^2$ meets a threshold (e.g., $\geq 0.5$), ensuring a minimum level of variance capture. $R_{max}$ is set to the original dimension $d$ of the activation space. The optimal rank $R_{opt}^{(h,l)}$ for head $(h, l)$ is then determined by minimizing the BIC score within this range:

$$R_{opt}^{(h,l)} = \underset{r \in [R_{min}, R_{max}]}{\arg\min} \left( Nd \cdot \log(\text{MSE}_r) + r \cdot (N + d + 1) \cdot \log(Nd) \right) \tag{1}$$

where $N$ is the number of samples, $d$ is the activation dimension, $r$ is the candidate rank, $\text{MSE}_r = \frac{1}{Nd}\|\Delta U^{(h,l)} - \Delta U_r^{(h,l)}\|_F^2$ is the mean squared reconstruction error using the rank-$r$ approximation $\Delta U_r^{(h,l)} = S_{:,1:r}^{(h,l)}\Sigma_{1:r,1:r}^{(h,l)}(V_{:,1:r}^{(h,l)})^T$, and the second term within the minimization penalizes model complexity based on the candidate rank $r$.

This procedure effectively selects the rank that provides the best trade-off between accurately reconstructing the style-related activation differences (low $\text{MSE}_r$) and maintaining a parsimonious representation (low $r$). By applying this optimization head-wise, we allow the dimensionality of the style subspace to adapt to the specific characteristics of each attention head. This calculation is performed offline during a preprocessing phase, determining the optimal $R_{opt}^{(h,l)}$ for each relevant head. These optimized ranks are then used during the online inference stage of the modified DRESS framework, replacing the fixed rank previously used. We hypothesize that this adaptive determination leads to improved overall performance by tailoring the intervention dimensionality more precisely to each head's contribution to style encoding.

### 4.3 K-Nearest Neighbor Stylization

The original DRESS framework determines the steering vector's direction based on the *global* average displacement between stylized ($u^{(h,l)+}$) and ordinary ($u^{(h,l)-}$) activations across the entire dataset $D$, projected onto the style subspace basis vectors $\{v_r^{(h,l)}\}_{r=1}^{R_{opt}}$. Specifically, the core directional component is derived from $\overline{\delta u}^{(h,l)} = \frac{1}{N}\sum_{j=1}^{N}(u_j^{(h,l)+} - u_j^{(h,l)-})$. While DRESS incorporates adaptive *strength* scaling ($\gamma_r^{(h,l)}$) based on the current activation $u^{(h,l)}$, the underlying steering *direction* remains fixed for a given head, determined by this global average difference. This "global style direction" aims to guide the current activation towards the overall center of the target style distribution within the subspace.

However, a single global direction might not capture the potentially complex or multi-modal structure of the style manifold within the subspace. To enable more localized and potentially adaptive steering directions, we propose a complementary approach termed K-Nearest Neighbor (KNN) Stylization. Instead of relying on the global average displacement, we dynamically determine the style direction based on the neighborhood of the current activation within the style subspace.

Formally, let $u^{(h,l)}$ be the current activation of head $(h, l)$ during inference (by the original, unedited model). We project the stored ordinary-style activations from the dataset, $\{u_j^{(h,l)-}\}_{j=1}^{N}$, onto the head's style subspace basis $V^{(h,l)} = [v_1^{(h,l)}, \ldots, v_{R_{opt}}^{(h,l)}]$. Let the projected activations be $\tilde{u}_j^{(h,l)-} = (V^{(h,l)})^T u_j^{(h,l)-}$. Similarly, project the current activation: $\tilde{u}^{(h,l)} = (V^{(h,l)})^T u^{(h,l)}$. We then identify the set $\mathcal{N}_K(u^{(h,l)})$ containing the indices of the $K$ nearest neighbors of $\tilde{u}^{(h,l)}$ among $\{\tilde{u}_j^{(h,l)-}\}_{j=1}^{N}$ based on Euclidean distance within the style subspace.

Instead of using the global average difference $\overline{\delta u}^{(h,l)}$, we compute a *local* average difference based only on these $K$ neighbors:

$$\overline{\delta u}_{KNN}^{(h,l)} = \frac{1}{K}\sum_{j \in \mathcal{N}_K(u^{(h,l)})}(u_j^{(h,l)+} - u_j^{(h,l)-}) \tag{2}$$

This local difference vector $\overline{\delta u}_{KNN}^{(h,l)}$ replaces the global $\overline{\delta u}^{(h,l)}$ in the calculation of the global strength components $\beta_r^{(h,l)}$ (Sec. 4.1):

$$\beta_{r,KNN}^{(h,l)} = \langle \overline{\delta u}_{KNN}^{(h,l)}, v_r^{(h,l)} \rangle \tag{3}$$

The adaptive scaling factor $\gamma_r^{(h,l)}$ and the final editing strength $\alpha_r^{(h,l)} = \lambda(1 + \gamma_r^{(h,l)})\beta_{r,KNN}^{(h,l)}$ are calculated as before, but using these neighbor-derived $\beta$ values. The intuition is that the style direction derived from nearby examples in the activation subspace might provide a more relevant and nuanced steering signal compared to the single global average, potentially improving adaptation to local variations within the style manifold. The number of neighbors $K$ becomes a hyperparameter controlling the locality of the direction estimate.

### 4.4 Acceleration via Decoupled KV Caching

The DRESS framework operates via an online editing regime: for each token generated, the model's internal steering vectors (biases) are dynamically adjusted based on the activations from a preceding "base" forward pass over the current sequence. This adaptive nature, while central to its mechanism, introduces significant computational overhead during inference. The generation of each token requires two effective forward passes through relevant layers: one to compute the base activations $u^{(h,l)}$ needed for calculating the adaptive strengths $\alpha_r^{(h,l)}$, and a second "style" forward pass using the edited model to predict the next token. This sequential dependency and repeated computation hinder throughput, limiting practical applicability.

Observing the DRESS online process, we can conceptualize the per-token generation as involving two distinct phases:

1. **Base Forward Pass:** The original, unedited model $M$ processes the current sequence prefix $T_q$ to obtain the head activations $A_{base} = \{u^{(h,l)}\}$ required for calculating the adaptive steering strengths.

2. **Style Forward Pass:** An edited version of the model $M'$, incorporating the calculated steering vectors $s^{(h,l)}$ based on $A_{base}$, processes the sequence prefix $T_{qa}$ (typically $T_q$ plus any QA formatting) to predict the next token $t_{style}$.

Standard transformer inference acceleration relies heavily on KV caching to avoid recomputing key and value representations for previous tokens. However, the dynamic editing in DRESS complicates direct application.

We propose methods to integrate KV caching by decoupling the caching mechanisms for the two forward passes, as outlined in Algorithm 1:

**Base KV Caching:** Applying KV caching to the Base Forward pass is straightforward. Since the model $M$ remains unchanged (biases are reset before this pass), the standard KV cache ($KV_{base}$) can be maintained and updated incrementally, reducing the complexity of this step from quadratic to linear in sequence length ($O(N^2) \to O(N)$). This optimization ( Base KV in Alg. 1) preserves the DRESS logic entirely, offering acceleration without approximation – essentially a "free lunch".

**Style KV Caching:** Applying KV caching to the Style Forward pass ( Base+Style KV in Alg. 1) introduces a nuance. The model $M'$ used in this pass is edited based on $A_{base}$, meaning its parameters (specifically, the implicit biases added post-attention) change at each step $i$. Therefore, the KV cache ($KV_{style}$) computed at step $i-1$ was generated by a slightly different model than the one used at step $i$. Reusing $KV_{style}$ is thus an approximation, as the cached keys and values are not strictly aligned with the current step's edited model. However, we hypothesize that since the edits are additions to activations post-attention, the impact on the keys and values themselves might be minor, making the cached values a sufficiently good approximation for efficient generation. Our experimental results (Sec. 6.4) validate this hypothesis, showing significant speed-up with minimal quality degradation.

By employing KV caching for both passes (Base+Style KV), we aim to reduce the complexity of both critical steps to $O(N)$, maximizing inference throughput for the adaptive editing framework.

**Algorithm 1** Accelerated Generation

```
Require: Model $M$, question tokens $T_q$, QA prefix tokens $T_{qa}$, pre-computed editing parameters $\theta$, max length $L$
Ensure: Generated answer tokens $T_{ans}$
 1: $T_{ans} \leftarrow [], KV_{base} \leftarrow [], KV_{style} \leftarrow []$
 2: for $i = 1$ to $L$ do
 3:     $M \leftarrow$ ResetModelBias($M$)
 4:     Base Forward step differs:
 5:     DRESS:          $t_{base}, A_{base}, KV_{base} \leftarrow$ ForwardPass($M, T_q$)                ▷ O($N^2$)
 6:     Base KV:        $t_{base}, A_{base}, KV_{base} \leftarrow$ ForwardPass($M, T_q, KV_{base}$)     ▷ O($N$)
 7:     Base+Style KV:  $t_{base}, A_{base}, KV_{base} \leftarrow$ ForwardPass($M, T_q, KV_{base}$)     ▷ O($N$)
 8:     $M \leftarrow$ EditModelBias($M, A_{base}, \theta$)
 9:     Style Forward step differs:
10:     DRESS:          $t_{style}, A_{style}, KV_{style} \leftarrow$ ForwardPass($M, T_{qa}$)              ▷ O($N^2$)
11:     Base KV:        $t_{style}, A_{style}, KV_{style} \leftarrow$ ForwardPass($M, T_{qa}$)              ▷ O($N^2$)
12:     Base+Style KV:  $t_{style}, A_{style}, KV_{style} \leftarrow$ ForwardPass($M, T_{qa}, KV_{style}$)  ▷ O($N$)
13:     $T_q, T_{qa}, T_{ans} \leftarrow T_q \oplus t_{style}, T_{qa} \oplus t_{style}, T_{ans} \oplus t_{style}$
14:     if IsStopToken($t_{style}$) then
15:         break
16:     end if
17: end for
18: return $T_{ans}$
```

## 5 Method: Adaptation of DRESS to Text-to-Speech Generation

### 5.1 "Text+Attributes"-to-Speech Framework of SparkTTS

The architecture for "text+attribute"-based speech generation in SparkTTS is illustrated in Figure 1. SparkTTS uses a decoder-only transformer, Qwen2.5-0.5B [13], as its speech language model. Given a transcript and three style attributes—gender (male, female), pitch, and speed (each with five levels: very low, low, moderate, high, very high)—tokenizers convert these into corresponding text and attribute tokens. During inference, the model produces two types of outputs: global tokens (a fixed-length sequence of 32 tokens encoding speaker characteristics and style) and semantic tokens (generated at 50 tokens per second to represent linguistic content). These tokens are then passed to a pretrained BiCodec [8] for detokenization, resulting in a final speech waveform in `.wav` format.

### 5.2 Single-Dressed SparkTTS

Our goal is to demonstrate that steering vector-based style transformation methods, such as DRESS, can also be applied to speech generation. To this end, we define two distinct attribute configurations: the original style as {gender: female, pitch: high, speed: high} and the target style as {gender: male, pitch: low, speed: low}. The objective is to apply DRESS to the activations corresponding to the original style, such that the generated speech exhibits the characteristics of the target style.

Our baseline method directly follows the original DRESS pipeline. For a set of 50 scripts, we generate activations using both the original and target attribute settings. From each pair, we extract the activations at the final token (<|*im_end*|>), as in the DRESS procedure. We then perform linear probing to identify the top-$K$ most style-relevant attention heads. Next, singular value decomposition (SVD) is applied to the activation differences from these heads to isolate a low-rank subspace that captures style variation while suppressing noise.

During inference, we provide the model with the input transcript and the original style attributes. Steering vectors are computed dynamically at each generation step and injected into the model activations, thereby guiding both the global and semantic token outputs toward the target style.

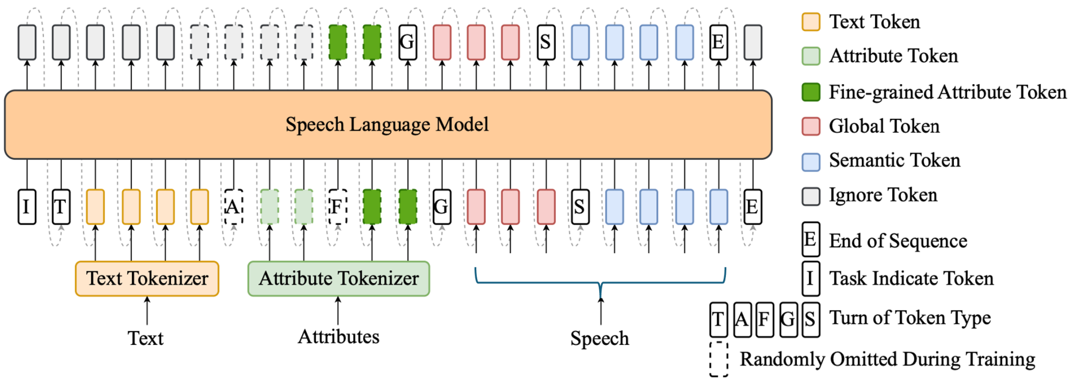

Figure 1: Spark-TTS speech generation process

### 5.3 Double-Dressed SparkTTS

Due to the distinct design of SparkTTS's output format—where fixed-length global tokens represent style information and variable-length semantic tokens encode linguistic content—directly applying DRESS based on the final end token alone proves insufficient. In our experiments, we observed frequent failure cases where no global tokens were generated. This is likely because the intrinsic differences between global and semantic tokens, making it difficult to determine a single global editing direction and strength that are simultaneously effective for both.

To address this limitation, we propose a *Double-Dress* method. During the activation extraction phase, we collect two separate sets of activations: those at the end of the global token sequence (<|*end_global_token*|>, denoted *ACTIVATIONS_GLOBAL*) and those at the end of the semantic token sequence (<|*im_end*|>, denoted *ACTIVATIONS_SEMANTIC*). During inference, we compute and inject steering vectors based on *ACTIVATIONS_GLOBAL* for global token generation and on *ACTIVATIONS_SEMANTIC* for semantic token generation. This separation allows for independent control over the two token streams, enabling more precise and stable stylization across both style and content dimensions.

## 6 Results of Improved Style Editing Framework

### 6.1 Experiment Settings

We evaluate our proposed improvements using the same stylized Question-Answering (QA) benchmarks introduced by [6]: *Shakespeare-style* (English) and *Dream of the Red Chamber-style* (Chinese). These datasets comprise paired ordinary-style and target-style responses with aligned semantics, constructed using both source texts and GPT-4 based data augmentation. The curation process, including the use of general QA datasets (MOSS for Chinese, Alpaca for English) and the train/test splits, follows the methodology detailed in the original DRESS paper.

To assess the quality of the generated stylized responses, we adopt the same set of metrics used in [6]. These include: (1) **Style Intensity (SI)**, measured by a separately trained BERT-based style classifier assessing the alignment with the target style; (2) **Semantic Preservation (SP)**, quantified by the averaged cosine similarity between the BGE embeddings [14] of the original and stylized responses; (3) **Fluency Score (FS)**, calculated as $1/(1 + \log \text{PPL})$ based on the perplexity assigned by the original base LLM; and (4) **Overall Assessment (OA)**, an objective composite score defined as $\text{OA} = \text{SI} \times \text{SP} \times \text{FS}$. Additionally, we report the **GPT-4 Rating** (0-10 scale) as a subjective measure of overall quality, following the evaluation protocol from the original study.

All experiments are conducted using the Qwen-1.5-14B-Chat model [15] as the base LLM, consistent with the original DRESS experiments. Computations are performed on a machine equipped with an NVIDIA A100 GPU. Key hyperparameters for the original DRESS components (e.g., number of initially selected heads $H$, overall editing strength $\lambda$) are retained from the original paper unless specified otherwise. For our proposed improvements, the CEVR threshold for $R_{min}$ determination is set to 0.5. Generation employs deterministic decoding (temperature set to 0) to ensure stable metric evaluation. Further implementation details mirror those described in [6].

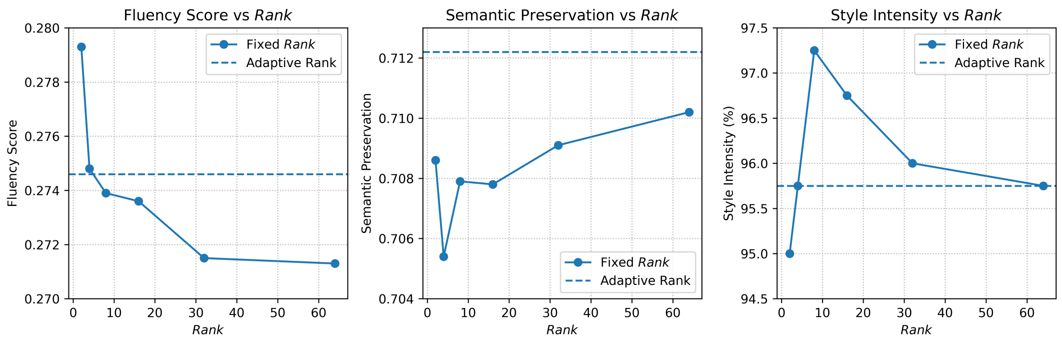

Figure 2: Impact of style subspace rank selection on generation quality.

### 6.2 Impact of Adaptive Style Subspace Determination

While the original DRESS framework investigated the sensitivity to the number of selected heads ($H$) and the global editing strength ($\lambda$), the influence of the style subspace rank ($R$) was underexplored, with a fixed value ($R = 16$) being heuristically adopted. As illustrated in Figure 2, our analysis reveals that the choice of rank impacts generation quality across multiple metrics. Notably, the fixed rank of 16 used in the original DRESS is suboptimal compared to ranks identified through broader exploration. This finding supports our motivation that a uniform rank assignment is insufficient given head heterogeneity. Our proposed adaptive rank determination method, leveraging BIC, automatically selects a potentially different $R_{opt}^{(h,l)}$ for each head. As shown in Table 1 and 2 (Adaptive Rank row), this automated approach achieves performance comparable or slightly superior (particularly in semantic preservation) to the manually tuned baseline DRESS, while obviating the need for heuristic rank selection.

| **Method** | **Fluency (FS)** | **Semantic (SP)** | **Style (SI)** | **Speed (token/s)** |
|---|---|---|---|---|
| DRESS | 0.2736 | <u>0.7078</u> | <u>96.75</u> | 2.51 |
| Adaptive Rank | 0.2746 | **0.7122** | 95.75 | 2.51 |
| Accelerated (Base+Style KV) | <u>0.2785</u> | 0.6979 | **97.00** | **7.32** |
| Adaptive Rank + Accelerated | **0.2789** | 0.6995 | **97.00** | **7.32** |

Table 1: Performance comparison of DRESS variants on the *Dream of the Red Chamber* benchmark.

| **Method** | **Fluency (FS)** | **Semantic (SP)** | **Style (SI)** | **Speed (token/s)** |
|---|---|---|---|---|
| DRESS | 0.2237 | **0.6485** | <u>99.75</u> | 2.51 |
| Adaptive Rank | 0.2280 | <u>0.6446</u> | **100.00** | 2.51 |
| Accelerated (Base+Style KV) | <u>0.2355</u> | 0.6259 | **100.00** | **7.32** |
| Adaptive Rank + Accelerated | **0.2387** | 0.6293 | **100.00** | **7.32** |

Table 2: Performance comparison of DRESS variants on the *Shakespeare* benchmark.

## 6.3 Analysis of KNN Stylization

Figure 3 demonstrates the effect of varying the number of neighbors ($K$) in our proposed KNN Stylization method. A clear trade-off emerges between style intensity (SI) and semantic preservation (SP). As $K$ decreases, SI generally increases while SP tends to decrease. This trend saturates at very low $K$ values (e.g., $K \approx 2000$ in this experiment), where style intensity reaches its peak, suggesting the editing process becomes heavily dominated by stylistic considerations. We hypothesize that using fewer, closer neighbors provides a more localized and potentially more accurate estimate of the immediate style direction required for the current activation, leading to stronger style manipulation. Conversely, increasing $K$ incorporates information from a broader set of reference samples. This averaging effect might mitigate the influence of semantic "noise" potentially entangled within the style subspace, even after the initial filtering steps, thereby better preserving semantics at the cost of slightly diluted style steering. This analysis not only validates KNN Stylization as an effective mechanism but also highlights that $K$ serves as an intuitive hyperparameter for explicitly controlling the balance between stylistic fidelity and semantic integrity. Furthermore, the observed trade-off reinforces the notion that the style subspaces identified, despite denoising efforts, may still retain residual semantic information.

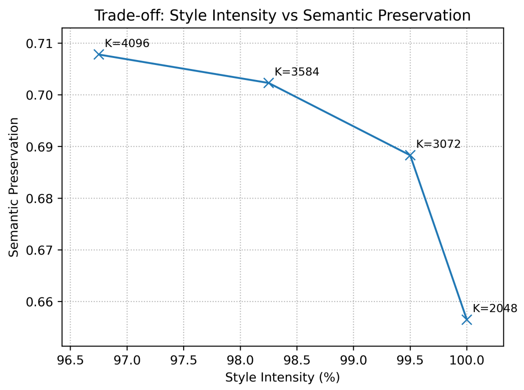

Figure 3: Trade-off between semantic preservation (SP) and style intensity (SI) as the number of neighbors ($K$) varies in KNN Stylization.

## 6.4 Evaluation of Accelerated Generation

Table 1 and 2 presents the results for our proposed generation acceleration technique utilizing KV caching for both the base and style forward passes (Base+Style KV). This approach achieves a substantial speed-up, reaching approximately 7.32 tokens/second—nearly a 3-fold increase compared to the original DRESS inference speed (2.51 tokens/second). Crucially, this acceleration is attained with minimal impact on generation quality. In fact, we observe improvements in fluency (FS) and style intensity (SI), with only a decrease in semantic preservation (SP). This outcome strongly suggests that the approximation introduced by reusing potentially "stale" KV cache values in the style forward pass (where biases change per step) is acceptable. It implies that recomputing the full attention context from scratch after each step's edit might be redundant for maintaining effective style control, paving the way for significantly more efficient stylized generation in practical applications. Furthermore, qualitative examples comparing the outputs of different methods are provided in Table 6.4.

| DRESS | Adaptive Rank | Accelerated |
|---|---|---|
| [*Shakespeare*-style] ***Q****: What is the time complexity of this linear search algorithm?* | | |
| Ver queth the time complexity of this linear search, yea, as tis in the bigode of algorhythms. The time complexity of this, is O(n), where n is the number of items in the tabele, or list, that you search. This means that as the list grows, so does the time, because each item must be searched, one by one, until the prize is found or not. Therefore, the linear search is of a snail's pace in terms of time, and not very efficient for a large tabele. | Ver queth the time complexity of this linear search, yea, as tis in the bigonometric measure. The algorithm, which we call 'thou', hase a time complexity of O(n). This means that as the size of the list (n) grows, so does the time it takes to find the eld (for it must seach through all elds in the list). So, the time it taketh to run the algorithm is in direct proportion to the size of the list. | Ver queth the time complexity of this linear search, yea, as it were, the run-time of the algorhythme. In this case, it shall be O(n), where n is the number of elements in the array. This means that as the elements increase, the time it shall take to find the element, will also increase, linearly. So, if the element is in the last, it shall take n-1, or else, n, to find it. Therefore, the time complexity of this linear search is of a proportionate verity, to the number of elements in the array. |
| [*Dream of the Red Chamber*-style] ***Q****:* 我想了解一下GANs生成像的原理。 | | |
| GANs，全名Generative Adversarial Networks，翻译过来就是生成对抗网络。这玩意儿的原理就是通过两个神经网络，一个叫生成器，一个叫判别器，这两者你来我往的斗智斗勇，最终生成的图像是跟真的似的，判别器也分不出真假来了，这就算是成了。生成器是负责生成假的，判别器是负责辨识真假，这两者一搭手，生成的图象就越来越逼真了。不过这玩意儿也有个毛病，那就是生成的图象有时候也会跑偏，生成出些奇奇怪怪的图象来，这就需要人工调调参数，或者换换模型，才能生成出好的图象来。 | GANs，全名Generative Adversarial Networks，翻译过来就是生成对抗网络。这玩意儿的原理就是通过两个神经网络，一个叫生成器，一个叫判别器，这两者你来我往的斗智斗勇，最后生成器把假的图儿生成得越来越像真的，判别器也把真的图儿辨认得越来越准。简单来说，就是个"猫儿偷鱼，狗儿守窝"的游戏，猫儿偷鱼偷得多了，狗儿也辨认得出是猫儿不是狗儿了，最后生成的图儿也就跟真的差不多了。 | GANs，全名Generative Adversarial Networks，翻译过来就是生成对抗网络。这玩意儿是两块儿，一块儿是生成器，一块儿是判别器。生成器是拿一堆儿乱七八糟的数据，给它整出一张新图儿来，判别器是拿这张新图儿，给它判别是不是真的。两块儿就在这儿玩儿猫儿腻，生成器想方设法整出真图儿来，判别器想方设法别让生成器整出假图儿来。这么玩儿来玩儿去，生成器就整出一张越来越真的图儿来，判别器也就越来越辨不出真假来了。等到判别器辨不出真假了，这图儿就算是生成出来了。这就是GANs生成像的原理。 |

Table 3: Qualitative examples: Stylized generation results for different methods.

## 7 Results of DRESS to Text-to-Speech Generation

### 7.1 Experiment Settings

As we adopt the "Text + Attributes" mode of speech generation, no explicit speech dataset is required. Instead, we prepare 50 training scripts, and get pairwise activations for both the original style attributes ({gender: female, pitch: high, speed: high}) and the target style attributes ({gender: male, pitch: low, speed: low}). Additionally, we construct a test set of 100 scripts and generate corresponding ground-truth stylized speeches using the target attributes. The objective is to apply the DRESS algorithm to the activations of these 100 test scripts—conditioned on the original style attributes—and edit them to produce speech that reflects the target style.

### 7.2 Evaluation metrics

**Speaker Similarity (SIM)**[16]: Speaker similarity evaluates whether the generated speech retains the identity of the reference speaker. Specifically, we utilize a pre-trained ECAPA-TDNN model to extract speaker embeddings from both reference speech $x_{\text{ref}}$ and generated speech $x_{\text{gen}}$. Then, we compute the cosine similarity between these embeddings: $\text{SIM}(x_{\text{ref}}, x_{\text{gen}}) = \frac{e_{\text{ref}} \cdot e_{\text{gen}}}{\|e_{\text{ref}}\| \|e_{\text{gen}}\|} \in [0, 1]$ where $e_{\text{ref}}$ and $e_{\text{gen}}$ are the speaker embeddings of reference and generated audio, respectively. A higher similarity score indicates stronger preservation of speaker identity.

**Universal Mean Opinion Score (UTMOS)**[17]: UTMOS provides an automatic estimation of speech naturalness based on UTMOS model trained on human ratings. Given a synthesized audio sample $x_{\text{gen}}$, UTMOS predicts a Mean Opinion Score (MOS): $\text{UTMOS}(x_{\text{gen}}) \approx \text{MOS} \in [1, 5]$ Here, 1 corresponds to extremely poor naturalness, and 5 corresponds to excellent naturalness. Higher scores reflect better perceptual speech quality.

**Short-Time Objective Intelligibility (STOI)**[18]: STOI measures the intelligibility of speech by comparing short-time spectral representations of reference and generated speech signals. The STOI score between reference speech $x_{\text{ref}}$ and generated speech $x_{\text{gen}}$ is calculated as follows: $\text{STOI}(x_{\text{ref}}, x_{\text{gen}}) \in [0, 1]$ A score close to 1 indicates high intelligibility, whereas scores approaching 0 suggest severely impaired intelligibility.

**Word Error Rate (WER)**: The WER measures the accuracy of speech transcription by comparing the Automatic Speech Recognition (ASR) transcriptions of reference speech and generated speech. It is defined by the normalized edit distance (Levenshtein distance) between the reference transcription $T_{\text{ref}}$ and hypothesis transcription $T_{\text{gen}}$: $\text{WER}(T_{\text{ref}}, T_{\text{gen}}) = \frac{S+D+I}{N}$ where $S$, $D$, and $I$ represent the number of substitutions, deletions, and insertions respectively, and $N$ denotes the total number of words in the reference transcript. Lower WER values indicate higher linguistic accuracy.

**EOS Miss**: During generation, we observe that applying DRESS can sometimes introduce overly aggressive modifications, causing the model not to output the End-of-Sequence (EOS) token. As a result, generation continues until the maximum sequence length is reached, leading to abnormally long or incomplete outputs. To quantify this issue, we record the number of such occurrences out of the 100 test samples as an indicator of generation stability.

### 7.3 Hyperparameter Setting

We mainly tune three hyperparameters, which are no.of heads, rank and $\lambda$. No.of heads is the number of most style-relevant heads selected by linear probing. Rank is the number of singular vectors used in SVD in the style subspace filtering. $\lambda$ is the overall scaling hyperparameter when injecting the steering vector.

According to Sections 5.2 and 5.3, we design three experimental configurations, summarized in Table 7.3. Method 1 uses the Single-DRESS SparkTTS setup, keeping all hyperparameters consistent with the original DRESS implementation, except for reducing the global steering strength $\lambda$ from 3 to 1.3 to prevent the model from failing to generate global tokens. Method 2 also uses Single-DRESS SparkTTS, but with all hyperparameters fine-tuned for better performance. Method 3 employs the Double-DRESS SparkTTS approach, with independently fine-tuned parameters for both global and semantic token editing.

| **Method** | **no. of heads** | **rank** | $\lambda$ |
|---|---|---|---|
| 1(Single-DRESS) | 64 | 16 | 1.3 |
| 2(Single-DRESS) | 20 | 5 | 1.3 |
| 3(Double-DRESS) | 128 for gloabl, 32 for semantic | 12 | 0.1 for global, 1.25 for semantic |

Table 4: Hyperparameter settings of Sparktts with DRESS

## 7.4 Quantitative Results

The evaluation results are presented in Table 7.4. The Target style output and Original style output refer to the speech samples generated by the original SparkTTS using the "text + attributes" mode and are included for reference.

The SIM and STOI metrics primarily reflect the degree of stylization. As shown, Method 2 achieves the strongest stylization effect, with both Method 2 and Method 3 showing clear increase over the Original Style Output. This supports the effectiveness of applying DRESS-based editing for speech stylization. It is worth noting that both metrics are designed for comparing speeches of the same length. Since speaking speed—one of our stylized attributes—alters the duration of generated speech, scores may be far lower than 1 even when the perceptual quality remains high, as confirmed by human evaluations.

The UTMOS metric assesses the naturalness and fidelity of the generated speech. Both Method 2 and Method 3 yield high UTMOS scores, indicating good consistency and perceptual quality, while Method 1 underperforms significantly in this regard.

WER and EOS Miss are used to measure generation stability. Comparing Methods 1 and 2, we observe that using too many attention heads or a high subspace rank can lead to overly aggressive editing, increasing the risk of degenerate outputs. Method 3 achieves the lowest WER and EOS Miss count, thanks to its separation of global and semantic token editing. By using <|*end_global_token*|> as the reference for global tokens—rather than the shared <|*im_end*|> token as in Methods 1 and 2—it provides more appropriate and targeted steering vectors, resulting in more stable and coherent generation.

Currently, the two sets of hyperparameters for global and semantic tokens are manually tuned using heuristic methods, which is both inefficient and suboptimal. As future work, we plan to incorporate adaptive $K$ selection into the Double-DRESS framework for automatic rank determination, and conduct a more systematic search for the optimal number of heads and overall scaling hyperparameters.

| **Experiment** | **SIM** $\uparrow$ | **UTMOS** $\uparrow$ | **STOI** $\uparrow$ | **WER** $\downarrow$ | **EOS Miss** $\downarrow$ |
|---|---|---|---|---|---|
| Target style output | 1.0000 | 3.6368 | 1.0000 | 0.0000 | - |
| Original style output | 0.1011 | 3.6724 | 0.1764 | 0.0605 | - |
| Method 1(Single-DRESS) | 0.1521 | 2.9617 | 0.1609 | <u>0.2795</u> | 16 |
| Method 2(Single-DRESS) | **0.2215** | **3.2000** | **0.1905** | 0.2821 | <u>11</u> |
| Method 3(Double-DRESS) | <u>0.1551</u> | <u>3.1860</u> | <u>0.1842</u> | **0.1946** | **2** |

Table 5: Quantitative Measurement on Sparktts with DRESS

## 7.5 Qualitative Results

Qualitative examples demonstrate that Method 2 and Method 3 consistently produce high-quality outputs closely matching the target style, whereas Method 1 yields unstable or unclear results.[2]

## 8 Conclusion and Future Work

In the first part of our work, we presented three improvements to the DRESS style editing framework. We introduced head-wise adaptive rank determination via BIC for enhanced robustness, KNN Stylization offering localized steering and explicit control over the style-semantic trade-off, and a decoupled KV caching strategy leading to faster inference with minimal quality degradation. These refinements enhance the controllability, efficiency, and robustness of representation editing for stylized generation. The trade-off observed with KNN Stylization, however, suggests residual semantic entanglement even after subspace filtering. Therefore, a key direction for future work is exploring more advanced disentanglement techniques (e.g., non-linear methods or adversarial approaches) to achieve a cleaner separation between style and content representations, potentially further improving semantic preservation with strong stylistic interventions.

In the second phase of our project, we successfully extended the DRESS style editing framework to the domain of speech generation using SparkTTS. We demonstrated that steering vector-based editing, originally developed for text, is also effective for modulating vocal attributes such as pitch, speed, and timbre in a zero-shot, text-to-speech setting. Our experiments showed that naive application of Single-DRESS can lead to unstable generation, especially due to the separate nature of global and semantic token streams in SparkTTS. To address this, we proposed a novel Double-DRESS strategy that independently edits the activations associated with global and semantic tokens, which achieves a significantly improved generation stability and perceptual quality. For future work, we plan to incorporate all Stage 1 enhancements—including adaptive subspace rank selection using BIC, KNN-based localized steering, and accelerated inference via KV caching—into the speech generation setting. Additionally, we aim to move beyond discrete attribute labels by introducing continuous attribute conditioning, and to automate the tuning of editing hyperparameters for each token stream, making the editing process more robust and scalable across diverse speaking styles.

[2]Audio samples of the above three methods together with the original and target style output are available at: `https://drive.google.com/drive/folders/1XFX9BCVQEwwyMXdDgl90v0htZYc4z4M4?usp=sharing`

## Author contribution

**Kunda Yang**:

In phase one: Re-implemented the DRESS framework from scratch. Conceived, implemented, experimented with (>70 experiments), and iterated on the improved style editing methodologies (Adaptive Rank, KNN Stylization, Acceleration). Provided the foundational codebase for phase two.

In phase two: Guided the selection of the appropriate TTS model architecture; proposed the Double-Dressed SparkTTS approach; assisted the team member with the implementation of the TTS adaptation.

For final report writing: Contributed to the writing of Sections 4, 5, 6, and 8 of the final report.

**Jiadong Hao**:

In phase one: Worked on the Adaptive k selection method and implemented two evaluation metrics: SP and FS.

In phase two: Implemented the whole code to transfer the DRESS to Sparktts, including Single and Double- Dressed Spark-TTS. Fine-tuned Method 1 and Method 3 mentioned in the report.

For final report writing: Responsible for section 2, 5, 7, 8.

**Yuchen Lu**:

In phase one: Fine-tuned the pretrained bert-based models based on the SP and DRC dataset and implemented two evaluation metrics: SI and GPT-rating.

In phase two: Explored the potential of applying the DRESS method to TTS models such as Llasa, Vits and Oute-TTS. Generated results with method 1. Designed four evaluation metrics to assess stylized TTS generation, implemented the corresponding evaluation scripts.

For final report writing: responsible for section 3, 7. Constructed demonstration examples to showcase the performance of the different DRESS methods on TTS models.

**Chengcheng Zhang**:

Investigate multiple tts models, such as llasa, OuteTTS, and speecht5, and ultimately determine the feasibility of sparktts and implement steering vector method.

Implement TTS stylized transfer to a scratch target (without knowing target attribute tokens). Based on Method 1, achieved Method 2.

For final report writing: responsible for section 5, 7.

**Bohan Zhang**:

Investigated several TTS models—including Llasa, OutTETTS, and SpeechT5—and ultimately confirmed the feasibility of Spark-TTS.

Generated paired Spark-TTS outputs to serve as the baseline and the target.

For final report writing: responsible for section 3.

## References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[3] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205, 2022.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[5] Xinyu Ma, Xu Chu, Zhibang Yang, Yang Lin, Xin Gao, and Junfeng Zhao. Parameter efficient quasi-orthogonal fine-tuning via givens rotation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 33686–33729. PMLR, July 2024.

[6] Xinyu Ma, Yifeng Xu, Yang Lin, Tianlong Wang, Xu Chu, Xin Gao, Junfeng Zhao, and Yasha Wang. Dressing up llm: Efficient stylized question-answering via style subspace editing. *arXiv preprint arXiv:2501.14371*, 2025.

[7] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] Xinsheng Wang, Mingqi Jiang, Ziyang Ma, Ziyu Zhang, Songxiang Liu, Linqin Li, Zheng Liang, Qixi Zheng, Rui Wang, Xiaoqin Feng, Weizhen Bian, Zhen Ye, Sitong Cheng, Ruibin Yuan, Zhixian Zhao, Xinfa Zhu, Jiahao Pan, Liumeng Xue, Pengcheng Zhu, Yunlin Chen, Zhifei Li, Xie Chen, Lei Xie, Yike Guo, and Wei Xue. Spark-tts: An efficient llm-based text-to-speech model with single-stream decoupled speech tokens, 2025.

[9] Zhen Ye, Xinfa Zhu, Chi min Chan, Xinsheng Wang, Xu Tan, Jiahe Lei, Yi Peng, Haohe Liu, Yizhu Jin, Zheqi Dai, Hongzhan Lin, Jianyi Chen, Xingjian Du, Liumeng Xue, Yunlin Chen, Zhifei Li, Lei Xie, Qiuqiang Kong, Yi-Ting Guo, and Wei Xue. Llasa: Scaling train-time and inference-time compute for llama-based speech synthesis. 2025.

[10] X. Du and et al. Cosyvoice2: A flow matching approach for acoustic feature prediction in tts. In *ICASSP 2024 – IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024.

[11] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[12] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[13] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[14] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[15] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[16] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. 10 2020.

[17] Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. Utmos: Utokyo-sarulab system for voicemos challenge 2022, 2022.

[18] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.