

4JO4 Research Project Report

Qianyue Wang

December 12, 2024

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Background Knowledge and Related Work researching | 2 |
| 2.1 | Non-Autoregressive Transformers (NAT) | 2 |
| 2.1.1 | RNN Model (Recurrent Neural Networks) | 2 |
| 2.1.2 | Transformer Model..... | 3 |
| 2.1.3 | NAT Model | 4 |
| 2.1.4 | Paper Analysis: Revisiting NAT for Efficient Image Synthesis | 5 |
| 2.2 | GLARE..... | 6 |
| 2.2.1 | Stage II | 6 |
| 2.2.2 | Stage III | 7 |
| 3 | Model Reproduction Results and Analysis | 8 |
| 3.1 | NAT Implementation | 8 |
| 3.1.1 | Understanding the NAT Implementation | 8 |
| 3.1.2 | Experimental Results on GLARE Dataset | 9 |
| 3.1.3 | Experimental Results on Low Light Enhancement Dataset | 10 |
| 3.2 | GLARE Implementation | 11 |
| 3.2.1 | Stage2..... | 11 |
| 3.2.2 | Stage3..... | 12 |
| 4 | New Model Results and Analysis | 13 |
| 4.1 | Model Modification and Rationale | 13 |
| 4.2 | Results of GLARE with NAT..... | 14 |
| 5 | Future Work | 15 |
| 6 | Conclusion | 16 |
| 7 | Time Schedule Gantt chart | 17 |

1 Introduction

This report summarizes my learning and exploration in the course, focusing on two state-of-the-art models: Non-Autoregressive Transformers (NAT) [5] for efficient image synthesis and GLARE [8] for low-light image enhancement. By reproducing and analyzing these models, I gained insights into their respective strengths, limitations, and potential synergies, while exploring the feasibility of integrating the two.

NAT, as introduced in [5], adopts a non-autoregressive framework for image generation, emphasizing efficiency and scalability. By predicting multiple image components simultaneously and leveraging random mask-based mechanisms, NAT achieves competitive image quality with reduced computational complexity, making it suitable for diverse applications. However, its reliance on input image quality can limit its performance in scenarios involving low-light or severely degraded images. On the other hand, GLARE, proposed in [8], addresses the challenges of low-light image enhancement through a generative latent feature framework combined with codebook retrieval. This approach excels in recovering fine details and improving visibility in adverse lighting conditions, though it may lack the structural coherence achieved by NAT's random masking mechanism.

The complementary strengths of NAT and GLARE highlight the potential for integration. NAT excels in generating coherent and high-quality images from a global perspective, while GLARE's strengths lie in enhancing fine details under challenging lighting conditions. Combining these models allows for comprehensive improvements: achieving clear and coherent image reconstruction in low-light environments, addressing detail and consistency issues in degraded images, and expanding the adaptability and robustness of the models in complex scenarios.

In the following sections, I will outline my investigation and learning of the background knowledge related to the models, provide a detailed description of the reproduction experiments and the integration of NAT and GLARE, and present the results of their combination. Finally, I will propose directions for further advancements in workflows for image synthesis and enhancement.

2 Background Knowledge and Related Work researching

2.1 Non-Autoregressive Transformers (NAT)

NAT models enable parallel decoding by relaxing the sequential dependency of traditional transformers. While efficient, NAT may introduce artifacts or blur in generated images due to its reliance on non-sequential processes, making it less effective for low-light enhancement tasks. Since I was not familiar with NAT initially, I began by exploring foundational concepts in image processing, such as RNN and Transformer models, to build a comprehensive understanding.

2.1.1 RNN Model (Recurrent Neural Networks)

RNN is a Recurrent Neural Network mainly used for processing sequential data, relying on the state at each time step in a time series. Each node in an RNN combines the current input and the hidden state from the previous time step to generate a new hidden state. RNN has the following characteristics: 1) Sequential processing: RNN processes

input sequences in order, and the current time step's calculation depends on the states from previous time steps, making it suitable for tasks with temporal dependencies. The parameters u, w, v are shared across different time steps 2) Long-term dependency issue: Due to the interdependence of states across time steps, RNN struggles to effectively capture long-term dependencies, leading to issues such as vanishing gradients or exploding gradients. 3) Variants: To address the long-term dependency issue, variants like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) were developed, which can better capture long-term dependencies.

In RNN, as gradients are back propagated through multiple time steps, they are continuously multiplied by numbers smaller than 1 (leading to vanishing gradients) or greater than 1 (leading to exploding gradients). This phenomenon makes it difficult to learn from long sequences, as the network cannot effectively update weights, leading to a failure in the training process. The reasons for vanishing gradients in RNNs are as follows: 1) Dependency between time steps: RNN's backpropagation occurs step-by-step along the time axis, and the gradient is propagated backward to the previous time steps. For short sequences, this propagation is manageable, but for longer sequences, the gradient diminishes as time steps increase, eventually approaching zero. 2) Impact of activation functions: The commonly used activation functions in RNNs (e.g., sigmoid or tanh) constrain outputs to a specific range (sigmoid ranges from 0 to 1, and tanh ranges from -1 to 1). During multi-layer chain backpropagation, the derivatives of these activation functions are very small, which further diminishes the gradient. The residual structure, due to the nature of the "1+" term, The formulary proving that it is greater than one is shown in Figure 1. It ensures that the parameters do not become zero.

$$X_{Din} = X_{Aout} + C(B(X_{Aout}))$$

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} \frac{\partial X_{Din}}{\partial X_{Aout}}$$

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} [1 + \frac{\partial X_{Din}}{\partial X_C} \frac{\partial X_C}{\partial X_B} \frac{\partial X_B}{\partial X_{Aout}}]$$

Figure 1: Illustration of the vanishing gradient problem in RNNs as gradients diminish over long time steps.

2.1.2 Transformer Model

The core of the Transformer is the *self-attention mechanism*, which can directly capture global dependencies between different positions when processing sequences without relying on sequential order. The Transformer architecture is composed of an encoder and a decoder, where the encoder maps the input into context vectors, and the decoder generates the output. The overall structure of the Transformer, including the encoder and decoder.

****Improvements in Transformer Models:***

The Transformer introduced significant improvements to the attention mechanism, enabling it to handle sequential data more efficiently while also processing long-range dependencies. This accelerated training and addressed the limitations of traditional recurrent neural networks (e.g., RNNs). The key contributions include the elimination of

recurrent structures and the introduction of multi-head self-attention mechanisms. The attention mechanism can be mathematically expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right) V$$

where Q , K , and V represent the query, key, and value matrices, respectively, and d_k is the dimensionality of the key vectors.

****Key Components of the Transformer:***

- **Cross-Attention:** This mechanism integrates information from two sequences. It is primarily used in the decoder to combine the encoder’s output with the current state of the decoder, ensuring the decoder references the input sequence while generating the output.
- **Multi-Head Self-Attention:** This mechanism enables the model to capture different contextual information from multiple “attention heads” simultaneously. Each attention head computes attention weights independently in different subspaces, allowing the model to focus on various parts of the sequence at multiple levels.
- **Feed-Forward Neural Network (FFN):** The FFN further processes the attention output non-linearly, enhancing its representational capacity. The processing steps can be summarized as: Input \rightarrow Attention Mechanism \rightarrow Feed-Forward Neural Network.

2.1.3 NAT Model

****Characteristics and Definition:***

Non-Autoregressive Transformers (NAT) do not rely on the previous outputs, and each time step’s output is independent, meaning that the model can generate the entire sequence in parallel. This greatly accelerates the generation speed. Characteristics of non-autoregressive models include: 1) ***Parallel Generation*:** Non-autoregressive models do not depend on previous time steps, allowing the model to generate all positions in the sequence simultaneously. 2) ***Faster Speed*:** Since the generation at each time step is independent, all tokens can be generated at once, making non-autoregressive models much faster than autoregressive models when generating long sequences. Autoregressive and non-autoregressive models are primarily used in the decoder part.

****Process:***

- ***Input Processing*:** The image to be reconstructed is divided into a series of small patches, which are then mapped into the embedding space. The embedding operation can be done using a pre-trained VQ-VAE (Vector Quantized Variational AutoEncoder) or a similar model, generating discrete visual tokens.
- ***Encoding Stage (Encoder)*:** In the Transformer’s encoder part, the input embedded patches are encoded through multiple self-attention layers (Multi-Head Attention). This step captures global dependencies and spatial information between patches, laying the foundation for parallel decoding in the next stage.
- ***NAT Decoding Stage (Decoder with NAT)*:**

- *Masked Attention:* Initially, the outputs of all positions are masked. NAT can decode tokens at multiple positions in parallel, generating multiple visual tokens at each step and updating each token’s state through the self-attention mechanism.
- *Parallel Decoding:* NAT progressively generates the complete sequence of image patches through parallel decoding. Each step’s generation not only depends on previously generated tokens but also filters out unreliable predictions using a re-masking mechanism and repeats this process until all patches are generated.

***Decoder Process:**

NAT’s decoder process is parallel and does not rely on previously generated results.

1. At the beginning, all tokens at all positions are set to [MASK], indicating that the tokens at each position have not yet been generated. This fully masked input is processed by the Transformer encoder to obtain contextual information for each position. Then, parallel prediction is performed. NAT predicts tokens for all positions in parallel. In other words, the model generates an initial token prediction for each masked position in parallel, rather than generating one token at a time. The generated tokens may not be completely accurate and may contain some noise or errors.
2. *Re-masking Mechanism:* To improve generation quality, NAT performs re-masking after parallel generation. The model evaluates the confidence level of the generated tokens at each position. If certain positions have low confidence (i.e., the model is uncertain about the generation at these positions), the tokens at these positions will be re-masked, indicating that they need to be regenerated. The remaining high-confidence tokens will be retained and will not be modified.
3. *Multiple Iterations:* This process is repeated for multiple iterations. In each generation round, NAT generates tokens for all masked positions in parallel, then re-masks and regenerates tokens for low-confidence positions. Through several iterations, the model can gradually produce a high-quality complete sequence.

2.1.4 Paper Analysis: Revisiting NAT for Efficient Image Synthesis

The paper delves into how Non-Autoregressive Transformers (NAT) can efficiently handle image synthesis tasks by utilizing parallel decoding mechanisms.

***Parallel Decoder:**

The NAT decoder operates in a parallel manner, allowing simultaneous generation of tokens at all positions during each iteration. The decoding process adheres to the following equations [5]:

$$\hat{V}_i^{(t)} = \begin{cases} \sim \hat{p}_{\pi(t)}(V_i \mid \mathbf{V}^{(t-1)}), & \text{if } V_i^{(t-1)} = [\text{MASK}]; \\ V_i^{(t-1)}, & \text{otherwise.} \end{cases} \quad (1)$$

$$C_i^{(t)} = \begin{cases} \log \hat{p}(V_i = \hat{V}_i^{(t)} \mid \mathbf{V}^{(t-1)}), & \text{if } V_i^{(t-1)} = [\text{MASK}]; \\ +\infty, & \text{otherwise.} \end{cases} \quad (2)$$

The above formulas are at the heart of NAT’s parallel decoding process. Equation (1) describes how the model generates predictions for masked tokens, while Equation (2) explains how the confidence scores for each prediction are computed.

Equation (1): Masked Token Prediction. For every position i in the token sequence, the model checks whether the token is masked. If $V_i^{(t-1)} = [\text{MASK}]$, the model predicts the token value $\hat{V}_i^{(t)}$ by sampling from a probability distribution conditioned on the previously generated sequence $\mathbf{V}^{(t-1)}$. If $V_i^{(t-1)}$ is already determined, it remains unchanged. This approach allows the model to handle masked and unmasked tokens simultaneously during decoding. *Equation (2): Confidence Score Calculation.* The confidence score $C_i^{(t)}$ measures how reliable the prediction for each token is. For masked tokens, the confidence score is calculated as the logarithm of the predicted probability of the token. High-confidence tokens are retained, while low-confidence tokens are re-masked and re-generated in subsequent iterations. This iterative approach ensures that the final token sequence has high accuracy.

Significance of the Process: This process eliminates the need for sequential generation, significantly accelerating the image synthesis pipeline. By combining parallel decoding with iterative refinement through re-masking, NAT achieves both speed and accuracy, making it a promising choice for large-scale image generation tasks.

2.2 GLARE

GLARE (*Generative Latent Feature-based Codebook Retrieval for Low-Light Image Enhancement*) is a novel framework designed to address the challenges of enhancing low-light images. Unlike conventional methods that often rely on heuristic priors or simple end-to-end training, GLARE introduces a three-stage process that leverages a pre-learned normal-light codebook as guidance. This innovative approach enables the model to effectively align low-light features with their corresponding normal-light representations while preserving both structural integrity and fine-grained details. By combining generative latent feature learning and adaptive feature refinement, GLARE provides a robust and flexible solution for low-light image enhancement tasks.

In the GLARE framework, the image enhancement process is divided into three distinct stages: 1) *Stage I: Normal-Light Codebook Learning*, 2) *Stage II: Generative Latent Feature Learning*, and 3) *Stage III: Adaptive Feature Transformation*. Each stage is designed to address specific challenges in low-light image enhancement while leveraging a pre-learned normal-light prior to guide the transformation process. However, during the reproduction process, I directly executed Stage II and Stage III, skipping Stage I. Upon further investigation, it seems that Stage I is inherently incorporated into Stage II, as the normal-light codebook is pre-trained during earlier training phases and subsequently frozen for later stages. This design ensures that the codebook acts as a robust and consistent prior, significantly simplifying subsequent processing.

2.2.1 Stage II

In *Stage II*, the framework focuses on learning latent representations from low-light images and aligning them with pre-learned normal-light features. The input low-light image I_l is first encoded into a latent feature representation c_l through a conditional encoder (E_c), which extracts both local and global features using deep convolutional layers. This conditional encoder is designed to leverage the pre-learned normal-light codebook as

guidance, ensuring that the extracted features are conditioned to align with the structure and semantics of the normal-light domain.

The encoded features are then processed by an *invertible latent normalizing flow* (ILNF) module, which plays a critical role in aligning the low-light features c_l with the pre-learned normal-light space. The ILNF employs a bijective transformation f_θ to map the latent representation into a distribution that matches the normal-light codebook. This design ensures that the encoded features not only maintain structural consistency but also semantically match their corresponding normal-light counterparts. The invertible nature of the flow allows for both forward and backward transformations, improving the stability and efficiency of the training process.

Invertible Latent Normalizing Flow (ILNF) Once the latent representation c_l is generated, it is processed by the ILNF module, which maps the low-light features to the normal-light feature space. Key aspects of ILNF include:

- **Bijective Transformation:** ILNF employs a bijective mapping f_θ , ensuring that the encoded features can be transformed bidirectionally between the low-light and normal-light domains.
- **Structural Consistency:** By aligning low-light features with the normal-light space, ILNF maintains both structural and semantic fidelity.
- **Robust Distribution Learning:** ILNF models the complex distribution relationships between low-light and normal-light features, avoiding inconsistencies and feature loss.

By combining the strengths of the conditional encoder and ILNF, Stage II establishes a solid foundation for subsequent feature refinement and decoding. It ensures that the input features are structurally and semantically aligned, enabling high-quality image enhancement in later stages.

2.2.2 Stage III

In [Stage III](#), the framework introduces adaptive feature transformation, where the aligned features are further refined to optimize detail recovery and overall image quality. This stage employs two key mechanisms: the adaptive mix-up blocks (AMB) and deformable convolution layers (DeConv).

The adaptive mix-up blocks (AMB) play a crucial role in dynamically integrating features from different layers or stages. By adaptively blending multiple sources of feature information, AMBs balance global structural consistency with fine-grained detail enhancement. This dynamic combination ensures that both high-level semantic information and localized details are effectively retained, reducing artifacts and suppressing noise. AMBs act as an intermediary between coarse global alignment and detailed refinement, improving feature expressiveness across varying spatial resolutions.

The deformable convolution layers (DeConv) further enhance the refinement process by providing spatial flexibility. Unlike standard convolutional layers, DeConv layers allow the sampling grid to adapt based on the input feature map, enabling the model to focus on regions that require more complex or targeted adjustments. This mechanism is particularly beneficial for capturing intricate textures and addressing uneven illumination, as it ensures spatially-aware feature transformations that are contextually relevant.

To implement these transformations, Stage III integrates two decoders: the Multi-Feature Decoder (MF-Decoder) and the Normal-Light Decoder (NL-Decoder). The MF-Decoder aggregates and processes multi-scale feature information from earlier stages, ensuring that critical details and structural coherence are preserved. Meanwhile, the NL-Decoder utilizes the pre-learned normal-light codebook to align low-light features with their normal-light counterparts, serving as a reference for restoring natural color and brightness.

The overall process of Stage III operates as follows: initially, features from Stage II are passed through AMBs, where they are adaptively fused and refined. These intermediate features are then processed by DeConv layers, enabling precise, region-specific adjustments. The refined features are decoded through the MF-Decoder and NL-Decoder, working in tandem to generate the final enhanced image. This multi-stage integration ensures that the output image demonstrates significant improvements in detail recovery, brightness, and overall visual quality.

Compared to state-of-the-art models like EnlightenGAN [3], RetinexNet [6], and KinD++ [7], GLARE’s Stage III introduces a unique advantage by combining AMBs and DeConv layers with its codebook-based design. While other methods focus on adversarial learning, decomposition, or heuristic approaches, GLARE leverages these adaptive mechanisms to provide both precision and flexibility. The integration of the MF-Decoder and NL-Decoder ensures robust structural alignment and realistic detail restoration, making GLARE particularly effective for extreme low-light conditions.

In summary, the integration of adaptive mix-up blocks, deformable convolution layers, and dual decoders in Stage III enables GLARE to outperform existing methods by offering both fine-grained detail enhancement and global structural coherence. These components work synergistically to adapt to diverse enhancement requirements, setting GLARE apart as a powerful and versatile low-light enhancement framework.

3 Model Reproduction Results and Analysis

3.1 NAT Implementation

3.1.1 Understanding the NAT Implementation

To gain a deeper understanding of the Non-Autoregressive Transformer (NAT) used in image generation tasks, we constructed a process flowchart based on the implementation code, as shown in Figure 2. This flowchart illustrates the main steps of the NAT process from input image to final image generation.

First, the input image is transformed into discrete image tokens using a pre-trained VQ-GAN encoder, with the quantization layer completing the embedding of these tokens for subsequent processing. Next, the generated tokens are masked to simulate missing information, enabling the non-autoregressive parallel decoding process. Following this, the NAT scheduling module iteratively predicts the original values of the masked tokens. To ensure generation accuracy, this module re-masks unreliable tokens based on their confidence scores. This process is repeated multiple times until a complete sequence of tokens is generated. Finally, the token sequence is mapped back to the pixel space via the VQ-GAN decoder to produce the final image.

The core characteristics of this workflow include parallel processing and an iterative refinement masking strategy, which allows NAT to achieve high efficiency in image gen-

eration.

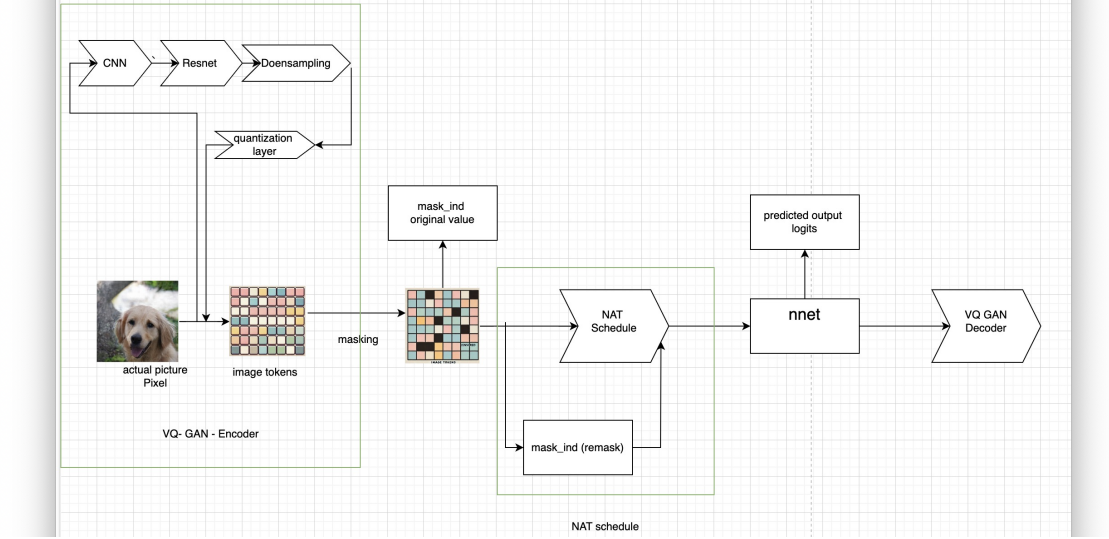


Figure 2: The process flowchart of NAT implementation. The flowchart outlines the key steps from input image processing to final image generation.

3.1.2 Experimental Results on GLARE Dataset

Using the GLARE dataset and the checkpoint provided in the official GitHub repository, the AutoNAT-S model was tested to evaluate its performance. The resulting FID score was approximately 8.47, showing a notable gap compared to the FID of 4.30 reported in the paper for ImageNet-256 [5]. However, despite this difference, the obtained result remains competitive when compared to other methods mentioned in the paper, I tested these methods with the code referenced in the paper [5] by using GLARE dataset. Table 1 presents a comparative analysis of FID scores achieved by different methods, highlighting the relative efficiency of AutoNAT-S.

| Method | Model Type | FID (GLARE) | Reported FID (ImageNet256) [5] |
|-----------------------|--------------------|-------------|--------------------------------|
| VQGAN [1]. | Autoregressive | 30.26 | 28.86 |
| VQ-Diffusion [2]. | Diffusion | 19.74 | 13.86 |
| Draft-and-revise [4]. | Non-Autoregressive | 12.18 | 9.65 |
| AutoNAT-S [5] | Non-Autoregressive | 8.47 | 2.80 |

Table 1: Comparison of FID scores across different methods on the GLARE dataset, including results reported.

The difference in FID scores between this experiment and the results reported in the paper [5] can be attributed to several factors:

- *Batch Size Adjustment:* Due to computational constraints, the batch size in the experiment was reduced from 128, as mentioned in the paper, to 64. This adjustment likely caused less stable gradient updates during model training and inference, which negatively impacted the quality of the generated images.
- *Dataset Characteristics:* Differences between the GLARE and ImageNet-256 datasets could also contribute to the performance gap. The GLARE dataset may have different distributional properties, which might lead to suboptimal results for a model fine-tuned or optimized for ImageNet-256.

- *Environmental and Hyperparameter Differences:* Variations in hardware environments, hyperparameter settings, or data preprocessing methods could further influence the outcomes. Even minor discrepancies in these aspects may affect the model’s performance.

Despite these challenges, the AutoNAT-S model still achieves competitive results on the GLARE dataset compared to other state-of-the-art methods, retaining its advantages in computational efficiency and generation quality. Therefore, the incorporation of NAT into GLARE methods demonstrates its potential to enhance generation efficiency and quality. As GLARE is a method introduced later, the integration of NAT into GLARE holds significant importance, which will be further discussed in the following sections.

3.1.3 Experimental Results on Low Light Enhancement Dataset

We conducted experiments on the Low-Light (LOL) enhancement dataset to evaluate the performance of our method. The results are shown in Figure 3, where we tested three different categories of images: original images (FID = 8.47), LOL-enhanced images with enhancement factors 0.3 (FID = 9.89) and 0.7 (FID = 9.38). Each row of the table in the figure corresponds to a different enhancement factor, displaying the LOL graph (enhanced image) and the final result produced by the model.







| LOL | FID | LOL graph | final result |
|-----|------|---|--|
| 0.3 | 9.89 |  |  |
| 0.7 | 9.38 |  |  |
| 1.0 | 8.47 |  |  |

Figure 3: Experimental results on the LOL dataset with different enhancement factors. The table shows the FID scores, LOL graphs (enhanced images), and final results for each enhancement factor (0.3, 0.7, and original).

From the FID scores, it is clear that the LOL-enhanced images (0.3 and 0.7) result in higher FID scores compared to the original images. This indicates that the current method has limited ability to handle LOL-based enhancement effectively, especially without training. The original image achieves the lowest FID score (8.47), while the LOL-enhanced images show worse performance with higher FID scores (9.89 and 9.38). This could be attributed to two key reasons:

- The LOL dataset lacks sufficient training data. my experiments were conducted solely in a testing scenario without fine-tuning the model, limiting its ability to generalize well.
- The enhancement method applied to the LOL images is suboptimal in capturing the underlying data distribution required for better generalization.

To address these limitations, I propose integrating the GLARE method with the current approach. By combining GLARE with LOL enhancement, we aim to improve the visual quality of the generated images and reduce the FID scores.

3.2 GLARE Implementation

3.2.1 Stage2

Through different training checkpoints, we analyze the trends of Adjust_PSNR, Adjust_LPIPS, Adjust_SSIM, and the original PSNR, LPIPS, SSIM. These metrics effectively reflect the model’s recovery quality and perceptual performance in image reconstruction tasks. The experimental results are shown in Table 2, and more detailed data can be found at [GitHub link](#).

| Checkpoint | Adjust_PSNR | Adjust_LPIPS | Adjust_SSIM | PSNR | LPIPS | SSIM |
|------------|-------------|--------------|-------------|---------|--------|--------|
| 1000 | 21.4671 | 0.4220 | 0.7272 | 18.1524 | 0.4284 | 0.7138 |
| 10000 | 24.5305 | 0.1855 | 0.8297 | 21.4741 | 0.1890 | 0.8201 |
| 30000 | 25.5241 | 0.1299 | 0.8462 | 20.8933 | 0.1416 | 0.8217 |
| 45400 | 25.9406 | 0.1335 | 0.8395 | 22.5916 | 0.1370 | 0.8250 |
| 60150 | 26.3710 | 0.1240 | 0.8493 | 23.0591 | 0.1291 | 0.8326 |

Table 2: Performance Metrics Across Different Training Checkpoints

**Data Analysis and Trend Interpretation*

- **Adjust_PSNR:** Adjust_PSNR improves significantly throughout the training, rising from 21.4671 (1000_G.pth) to 26.3710 (60150_G.pth). This indicates that the model continuously enhances its ability to reconstruct image details and reduces the error with target images.
- **Adjust_LPIPS:** Adjust_LPIPS decreases from 0.4220 (1000_G.pth) to 0.1240 (60150_G.pth). The lower perceptual loss demonstrates that the generated images are perceptually closer to real images, with improved feature capture.
- **Adjust_SSIM:** Adjust_SSIM steadily increases from 0.7272 (1000_G.pth) to 0.8493 (60150_G.pth), reflecting improvements in pixel-level quality and structural similarity.
- **Original Metrics:** Similar trends are observed in the original metrics. PSNR increases from 18.1524 to 23.0591, LPIPS decreases from 0.4284 to 0.1291, and SSIM improves from 0.7138 to 0.8326. These trends align with the adjusted metrics, further confirming the enhanced reconstruction performance.

**Trend Summary*

- The continuous rise of Adjust_PSNR and Adjust_SSIM, along with the notable decline of Adjust_LPIPS, highlights the model’s substantial performance improvements during training. In later checkpoints, the metrics stabilize, indicating the model’s near-optimal state.
- The final results align closely with the ideal ranges mentioned in the GLARE paper (PSNR 26.2–26.5, SSIM 0.855, LPIPS 0.11), confirming the success of the replication process.
- After 30000_G.pth, the improvement in metrics slows down, suggesting that the model has captured most key features. This observation could inform optimization of training duration to avoid unnecessary resource consumption.

****Impact of Learning Rate and Issues Encountered***

The impact of different learning rates (lr) on model performance and training stability was analyzed. Table 3 summarizes the results for lr values of 2×10^{-4} , 4×10^{-4} , and 5×10^{-4} . The choice of learning rate significantly influences both the quality of the reconstructed images and the stability of the training process.

| LR | Adjust_PSNR | Adjust_SSIM | Adjust_LPIPS | PSNR | SSIM | LPIPS |
|--------------------|-------------|-------------|--------------|---------|--------|--------|
| 2×10^{-4} | 24.8153 | 0.8879 | 0.1331 | 20.6928 | 0.8546 | 0.1406 |
| 4×10^{-4} | 26.3885 | 0.8488 | 0.1285 | 22.9404 | 0.8320 | 0.1335 |
| 5×10^{-4} | NaN | NaN | NaN | NaN | NaN | NaN |

Table 3: Performance Metrics Across Different Learning Rates (Checkpoint: 60000_G.pth)

The data indicate that a lower learning rate of 2×10^{-4} results in suboptimal performance compared to 4×10^{-4} . For instance, Adjust_PSNR is limited to 24.8153, and PSNR reaches only 20.6928, which are significantly lower than the metrics achieved with 4×10^{-4} . However, the training process remains stable at this lower rate, and no anomalies, such as NaN or Inf values, are observed.

At 4×10^{-4} , the model achieves the best performance, with Adjust_PSNR reaching 26.3885 and Adjust_LPIPS decreasing to 0.1285, indicating higher reconstruction quality and perceptual similarity. These results are closely aligned with the benchmarks reported in the GLARE paper, demonstrating the effectiveness of this learning rate for the given configuration.

On the other hand, using a higher learning rate of 5×10^{-4} causes significant instability, leading to NaN or Inf values in the input tensors. This instability is likely due to gradient explosion or numerical issues during optimization, which can disrupt the convergence process. The occurrence of such issues suggests that an excessively high learning rate pushes the model’s parameter updates beyond stable limits, resulting in divergence.

This analysis underscores the critical role of learning rate selection in balancing training performance and stability. While a lower learning rate ensures stability, it may fail to fully optimize the model’s potential. Conversely, an excessively high learning rate risks numerical instability and training failure. Therefore, selecting a moderate learning rate, such as 4×10^{-4} , provides a balance between achieving optimal results and maintaining computational stability.

3.2.2 Stage3

****Data Analysis and Trends in Stage 3***

Stage 3 introduces the Adaptive Feature Transformation module, which integrates both global and local features through the combination of VQ, MF Decoder, and NL Decoder. This architecture enhances the model’s ability to reconstruct high-quality images, and the trends in the experimental data strongly reflect this improvement. Table 4 presents a detailed comparison of Stage 2 and Stage 3 metrics across multiple checkpoints, highlighting the significant gains achieved in this stage. In addition, more detailed data can be found at [GitHub link](#).

The Adjust PSNR metric consistently improves in Stage 3, with values increasing from 26.1090 at Checkpoint 1000 to 27.8090 at Checkpoint 60150, indicating a substantial enhancement in detail reconstruction. Similarly, Adjust SSIM rises steadily from 0.8425 to 0.8792, reflecting better preservation of structural similarity in the reconstructed images. In contrast, the Adjust LPIPS metric shows a sharp decline, reducing from 0.1431 to 0.0802 across the same range of checkpoints, underscoring the model’s ability to generate images that are perceptually closer to real ones.

These trends can be attributed to the specific design of Stage 3. The VQ module quantizes the latent features, enabling more effective representation, while the MF Decoder emphasizes local details. Concurrently, the NL Decoder enhances global feature integration, which is critical for maintaining overall structural integrity. The combination of DecConv and AMB modules further refines feature transformation, ensuring that both pixel-level and perceptual quality improve significantly.

Table 4 demonstrates these improvements clearly, providing quantitative evidence of Stage 3’s superior performance. These results indicate that the Adaptive Feature Transformation module effectively balances local and global feature representation, ensuring consistent performance gains throughout the training process.

In summary, Stage 3 leverages its unique architecture to substantially enhance the quality of image reconstruction. The experimental data confirm its success in achieving higher Adjust PSNR and Adjust SSIM while reducing Adjust LPIPS, underscoring its ability to generate detailed and perceptually accurate reconstructions.

| Checkpoint | S2 PSNR | S2 SSIM | S2 LPIPS | S3 PSNR | S3 SSIM | S3 LPIPS |
|------------|---------|---------|----------|-----------|----------|----------|
| 1000 | 21.4671 | 0.7272 | 0.4220 | 26.1090 ↑ | 0.8425 ↑ | 0.1431 ↓ |
| 10000 | 24.5305 | 0.8297 | 0.1855 | 27.0253 ↑ | 0.8676 ↑ | 0.0903 ↓ |
| 30000 | 25.5241 | 0.8462 | 0.1299 | 27.4739 ↑ | 0.8745 ↑ | 0.0811 ↓ |
| 45400 | 25.9406 | 0.8395 | 0.1335 | 27.6838 ↑ | 0.8800 ↑ | 0.0843 ↓ |
| 60150 | 26.3710 | 0.8493 | 0.1240 | 27.8090 ↑ | 0.8792 ↑ | 0.0802 ↓ |

Table 4: Comparison of Performance Metrics Between Stage 2 and Stage 3 Across Different Checkpoints

4 New Model Results and Analysis

4.1 Model Modification and Rationale

In Stage 2 of the model, significant improvements were made to the generative process to enhance reconstruction quality and capture fine-grained details. The original framework in Stage 2 centers on leveraging an invertible latent normalizing flow to process and invert representations in the latent space. However, to improve the model’s ability to represent complex features and handle uncertainty in specific regions, we introduced two key components after the flow operation: the **mask token** and the NAT (Noise-Aware Transformer) module.

The **mask token** was incorporated to explicitly handle incomplete or uncertain information in the latent space. This token serves as a special indicator to highlight regions of interest or ambiguity within the latent space, guiding the model to prioritize these areas during reconstruction. By doing so, the model achieves more focused optimization on local features, enabling better restoration of intricate scenarios.

Following this, the NAT module was integrated to process regions marked by the mask token. The NAT module leverages the global modeling capabilities of Transformers, effectively capturing long-range dependencies while addressing localized uncertainty. Unlike traditional local filtering methods, NAT provides a more flexible approach to modeling and extracts finer details from the latent space. This enhancement improves the model’s robustness in handling noisy or complex regions, ultimately leading to higher-quality image reconstructions.

During training, the NAT module was incorporated seamlessly into the optimization workflow. After the latent space representations are inverted, the NAT module further processes them in conjunction with the mask token markers, refining the conditioned features to achieve high-fidelity reconstructions. This design significantly enhances the model’s adaptability to complex samples during training and exhibits superior stability during inference.

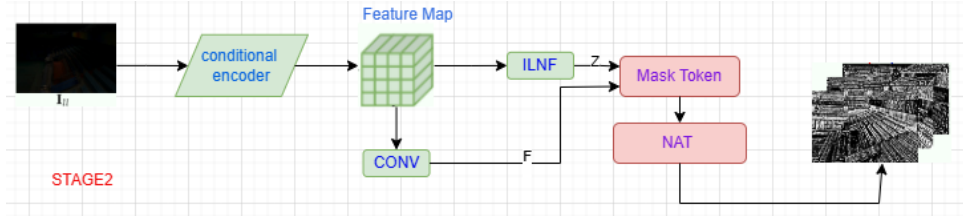


Figure 4: Illustration of the proposed workflow in Stage 2. Green components represent unchanged parts, while red components indicate the newly introduced mask token and NAT module. This figure is adapted and modified based on the workflow presented in [5] to illustrate the updates in our model.

As shown in Figure 4, the workflow depicts the path from the latent space to the final reconstructed outputs. The modifications emphasize the integration of the mask token and NAT module within the overall system. These additions are highlighted in red to distinguish them from the original, unchanged parts of the model, which remain in green. This visual representation provides a clear overview of how the proposed changes are incorporated into the original framework to enhance the reconstruction process. In addition, the detailed code can be seen in the following link: [Google Drive Link](#).

4.2 Results of GLARE with NAT

To evaluate the effect of NAT in the GLARE model, we compared the performance of the original Stage 2 model and the Stage 2 model with NAT scheduling across multiple checkpoints. The results, as shown in Table 5, focus on three key metrics: PSNR, SSIM, and LPIPS, which measure reconstruction quality, structural similarity, and perceptual fidelity, respectively. In addition, more detailed data can be found at [GitHub link](#).

The results show that PSNR improves consistently with the inclusion of NAT across all checkpoints. In the first 30,000 iterations, the increase in PSNR is modest; for instance, it rises from 21.4671 to 21.5137 at checkpoint 1000 and from 24.5305 to 24.6506 at checkpoint 10,000. However, after 30,000 iterations, the improvement becomes more significant, reaching 27.0827 at checkpoint 60,150. This value approaches the performance

| Checkpoint | S2 PSNR | S2 SSIM | S2 LPIPS | S2 NAT PSNR | S2 NAT SSIM | S2 NAT LPIPS |
|------------|---------|---------|----------|-------------|-------------|--------------|
| 1000 | 21.4671 | 0.7272 | 0.4220 | 21.5137 ↑ | 0.7342 ↑ | 0.4103 ↓ |
| 10000 | 24.5305 | 0.8297 | 0.1855 | 24.6506 ↑ | 0.8336 ↑ | 0.1828 ↓ |
| 30000 | 25.5241 | 0.8462 | 0.1299 | 26.0662 ↑ | 0.8532 ↑ | 0.1217 ↓ |
| 45400 | 25.9406 | 0.8395 | 0.1335 | 26.5969 ↑ | 0.8573 ↑ | 0.1247 ↓ |
| 60150 | 27.0827 | 0.8493 | 0.1240 | 26.5969 ↑ | 0.8678 ↑ | 0.1162 ↓ |

Table 5: Comparison of Performance Metrics Between Stage 2 of the Original Model and the Model with NAT Scheduling Across Different Checkpoints

of Stage 3 in the original model. It is evident that the model’s results improve more noticeably as training progresses, indicating that the NAT module better aligns latent representations in later stages of training. This suggests that the NAT module enhances the model’s ability to recover high-frequency details in reconstructed images during the later training phases.

For SSIM, a stable improvement trend is observed as well. Across all checkpoints, the SSIM metric increases consistently, such as from 0.7272 to 0.7342 at checkpoint 1000 and from 0.8493 to 0.8678 at checkpoint 60,150. The steady improvement in SSIM indicates that the NAT module significantly enhances the structural similarity, particularly in preserving geometric and contextual features.

In contrast, LPIPS decreases slightly with the addition of NAT, indicating that reconstructed images become closer to “real images” in terms of human visual perception. The reduction in LPIPS is relatively small, such as from 0.4220 to 0.4103 at checkpoint 1000 and from 0.1240 to 0.1162 at checkpoint 60,150. This slight decrease suggests that NAT optimizes latent features to improve perceptual fidelity while maintaining a balance between structure and context, without introducing significant artifacts or degrading overall reconstruction quality. The decreasing LPIPS can be attributed to the NAT module’s fine-grained optimization of latent features, which enhances perceptual quality.

In summary, the inclusion of NAT significantly improves reconstruction quality, reflected in higher PSNR and SSIM scores and a slight reduction in LPIPS. These enhancements demonstrate the effectiveness of NAT in aligning latent representations and improving both perceptual and structural fidelity. Furthermore, the total training time for Stage 2 with NAT increased from 10 hours to 18 hours. This additional training time is primarily due to the extra computational overhead introduced by mask token integration and latent feature alignment in the NAT module. However, this increase in training time is a reasonable trade-off for the observed performance gains.

5 Future Work

The integration of NAT into the GLARE model has demonstrated a positive impact on reconstruction performance, as evidenced by the upward trends in key metrics. This indicates that introducing NAT in Stage 2 is an effective approach to improving the quality of reconstructed images. However, the observed performance improvement is not as significant as anticipated, suggesting that there is room for further optimization.

One potential avenue for improvement is to integrate NAT into Stage 3. For instance, NAT can be applied during the “Adaptive Feature Transformation” phase to further refine the latent features before final reconstruction. If NAT is introduced in both Stage 2 and Stage 3, it may enhance the overall feature alignment and transformation processes, leading to better reconstruction performance and more robust results.

In addition to expanding NAT’s application, introducing new evaluation metrics can

provide a more comprehensive assessment of the model’s performance. While the current metrics (PSNR, SSIM, and LPIPS) effectively evaluate image quality and perceptual fidelity, they may not fully capture subtle differences in reconstruction quality, such as high-frequency details or specific artifacts.

- **FID:** FID evaluates the quality of generated images by calculating the distributional difference between generated and real images. It combines pixel-level distribution characteristics with perceptual semantic information, making it effective in assessing image realism and naturalness. Introducing FID can quantify how closely reconstructed images align with high-quality standards in human visual perception, particularly in terms of detail and distribution consistency.
- **Structural Similarity Index for Multi-Scale (MS-SSIM):** MS-SSIM is a multi-scale extension of SSIM, capable of capturing structural similarity at various scales. Compared to single-scale SSIM, MS-SSIM is more comprehensive as it evaluates both low-frequency and high-frequency information. For images with intricate structures and global features, such as complex patterns and fine details, MS-SSIM can provide a more accurate evaluation.

By introducing these new metrics, future research can more comprehensively evaluate the performance improvements brought by GLARE. FID measures image realism directly, while MS-SSIM captures detail and structural information at multiple scales, providing a more holistic assessment of model performance.

6 Conclusion

Through this course, I have deepened my understanding of low-light enhancement and image restoration techniques by reproducing and experimenting with the NAT and Glare models. During the NAT model reproduction, I learned about its advantage of using randomly generated masks, which performed well on normal images but showed limited effectiveness for low-light enhancement. On the other hand, the Glare model demonstrated better performance in low-light scenarios, making it a promising approach for further exploration.

In my attempt to combine the NAT and Glare models, I gained insights into the fundamentals of model integration. Although the experimental results showed only slight improvements, this effort highlighted the strengths and weaknesses of each model and emphasized the importance of selecting suitable models or weight assignments for specific tasks. I also realized that the potential of this combination could be better unlocked by optimizing additional parameters or structures, such as integrating NAT modules into multiple parts of the Glare model or experimenting with alternative combinations.

Through this study, I not only gained a deeper theoretical understanding of image processing but also acquired practical experience in running and evaluating models (e.g., FID data comparison and PSNR, SSIM, LPIPS analysis). These experiences have laid a solid foundation for undertaking more complex research projects in the future.

7 Time Schedule Gantt chart

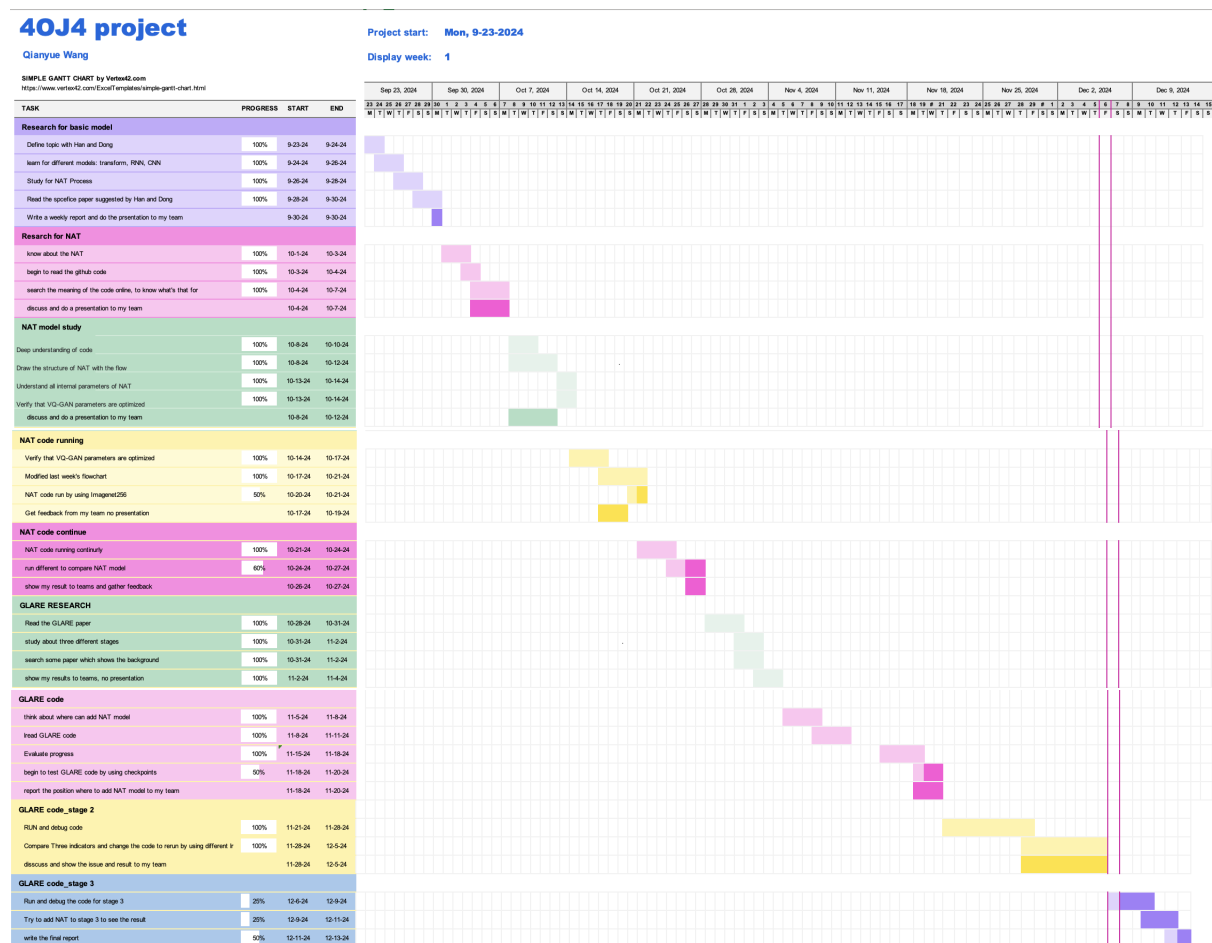


Figure 5: Time shcedule.

References

- [1] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [2] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [3] Yifan Jiang, Xinyu Gong, Ding Liu, Ziqian Cheng, Hao Fang, Xin Shen, and Jianchao Yang. Enlightengan: Deep light enhancement without paired supervision. In *ACM International Conference on Multimedia (MM)*, 2021.
- [4] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Draft-and-revise: Effective image generation with contextual rq-transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

- [5] Z. Ni, J. Han, X. Wu, et al. Revisiting non-autoregressive transformers for efficient image synthesis. In *Proceedings of the CVPR 2024*, 2024.
- [6] Cheng Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *British Machine Vision Conference (BMVC)*, 2018.
- [7] Yingkun Zhang, Jiayi Zhang, and Xiaojie Guo. Kindling the darkness: A practical low-light image enhancer. *IEEE Transactions on Image Processing*, 29:4249–4264, 2020.
- [8] H. Zhou, W. Dong, X. Liu, S. Liu, X. Min, G. Zhai, and J. Chen. Glare: Low light image enhancement via generative latent feature-based codebook retrieval. In *Proceedings of the ECCV 2024*, 2024.