ArtEdge: Real-Time Neural Style Transfer for Mobile Devices

Nayeem Mohammad
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
nayeem@email.sc.edu

Nitin Gupta
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
niting@email.sc.edu

***Abstract*—Neural style transfer (NST) offers powerful tools for artistic image transformation, but real-time execution on resource-constrained mobile devices presents significant challenges. This paper introduces *ArtEdge*, a mobile-first system performing NST directly on-device via edge computing. *ArtEdge* enhances user privacy and achieves low latency by eliminating cloud dependency. We implemented and adapted four diverse NST models: a custom MobileNet AdaIN, Fast Style Transfer, the transformer-based StyTr^2, and the mobile-optimized AesFA. All models are deployed on iOS using Core ML for hardware acceleration. Evaluations on an iPhone 15 Pro demonstrated the feasibility of near real-time performance, with inference times significantly reduced compared to CPU execution, reaching millisecond levels for lighter models. Our results highlight a clear trade-off: lightweight models like MobileNet AdaIN and Fast Style Transfer offer superior speed, while advanced architectures like AesFA and StyTr^2 provide higher visual fidelity at increased computational cost. *ArtEdge* successfully demonstrates the viability of deploying sophisticated NST models on mobile platforms, paving the way for accessible and responsive creative tools. All code files for the app and models can be found at the following link: github.com/g-nitin/ArtEdge**

***Index Terms*—Neural Style Transfer, Edge Computing, Real-Time Image Processing, Mobile Computing**

I. Introduction

Neural style transfer (NST) has emerged as a transformative technique at the intersection of computer vision and digital art, enabling the reimagining of conventional imagery through the application of distinct artistic styles. Since the seminal work by Gatys et al. [2] established the foundational framework for NST, subsequent advancements have significantly enhanced both its visual quality and computational efficiency. Despite these improvements, the practical deployment of NST on resource-constrained platforms remains a formidable challenge, particularly when real-time performance and local data privacy are paramount.

Our proposed solution, *ArtEdge*, addresses these challenges by developing a mobile-first solution for real-time neural style transfer. Unlike traditional approaches that rely on cloud-based processing, *ArtEdge* leverages edge computing to process images directly on mobile devices. This approach not only reduces latency from seconds to milliseconds but also enhances privacy by ensuring that sensitive images remain on-device. By emphasizing real-time performance and memory efficiency, *ArtEdge* aims to democratize access to sophisticated artistic tools and promote more inclusive visual communication technologies.

Building upon a robust body of literature, from the initial CNN-based techniques and their iterative optimization challenges to the emergence of transformer-based and diffusion models, *ArtEdge* situates itself within a continuum of research dedicated to balancing computational demands with artistic fidelity. Recent mobile-optimized frameworks have demonstrated that domain-specific innovations, such as frequency-based decomposition and compact network architectures, can achieve impressive inference speeds without sacrificing quality. *ArtEdge* synthesizes these insights to overcome critical constraints associated with processing speed and memory limitations inherent in mobile environments.

The remainder of this paper is organized as follows. We begin with a detailed review of existing NST methodologies and their evolution, highlighting both foundational approaches and recent innovations tailored for mobile platforms. We then describe the architecture and implementation of *ArtEdge*, outlining the design decisions that enable real-time processing under mobile constraints. Finally, we evaluate the system's performance relative to current state-of-the-art methods and discuss its implications for future research in both NST and edge computing domains.

II. Literature Review

The evolution of neural style transfer has undergone significant advancements, driven by computational efficiency, visual quality, and adaptability to resource-constrained platforms like mobile devices. Adopting the structured review approach outlined in the survey paper by Zhou et al. [14], this section presents a chronological progression of key NST methodologies, some common evaluation metrics, and applications of NST, with a specific focus on mobile-based development.

A. Foundational CNN-Based Approaches

The field of neural style transfer was pioneered by Gatys et al. in 2016, who introduced a novel framework using convolutional neural networks (CNNs) to separate and recombine content and style features. By optimizing a noise image through iterative minimization of content and style losses, the

latter computed via Gram matrices to capture texture statistics, their method achieved groundbreaking and visually appealing artistic results. However, the computational intensity of this approach, requiring up to 51 seconds on average to stylize a 512×512 image [5], rendered it impractical for real-time applications. Despite this limitation, Gatys et al. established the conceptual foundation for feature disentanglement, a principle later refined for efficiency.

Addressing the speed limitations of the optimization-based approach, Johnson et al. [6] introduced Fast Style Transfer (FST). This method trains a dedicated feed-forward convolutional neural network for each specific artistic style. Instead of iterative optimization at inference time, the network directly transforms a content image into a stylized version in a single pass. This is achieved by training the network using perceptual loss functions similar to those proposed by Gatys et al. The primary advantage of FST was its dramatically reduced inference time, enabling near real-time stylization. However, its main drawback is the inflexibility inherent in requiring a separate, pre-trained network for every desired style, unlike methods that aim for arbitrary style transfer.

Building on the goal of real-time performance but aiming for greater flexibility, Huang and Belongie proposed Adaptive Instance Normalization (AdaIN) in 2017 [3], a landmark advancement enabling real-time arbitrary style transfer. By aligning the mean and variance of content features with those of style features through a lightweight affine transformation, AdaIN reduced inference times to 95 milliseconds on average for 512×512 images [5]. This efficiency made it a cornerstone for early mobile implementations. However, AdaIN's reliance on fixed VGG encoders limited its ability to handle complex or abstract styles, and its global feature alignment often sacrificed fine-grained local details critical for high-quality artistic outputs. Subsequent efforts, such as Dynamic Instance Normalization (Jing et al., 2020) [4], introduced learnable transformations to enhance flexibility but struggled with memory constraints on mobile hardware.

B. Transformer-Based Architectures

The advent of transformers [12] marked a paradigm shift in NST, leveraging self-attention mechanisms to model long-range dependencies and global context. Deng et al. (2022) introduced StyTr2 [1], a dual-path transformer architecture that separately encodes content and style sequences using domain-specific encoders, followed by a multi-layer decoder for stylization. This approach mitigated the content leakage problem prevalent in CNN-based methods, where style features inadvertently overwrote structural details. However, Deng et al.'s approach consumes more time than other approaches due to the large parameters of Transformers, making it challenging to deploy on mobile devices.

C. Diffusion Models and Scalability

Diffusion models have emerged as a powerful alternative for high-fidelity style transfer, exemplified by D²Styler, proposed by Susladkar et al. in 2023 [11]. This framework employs a discrete diffusion process to iteratively denoise latent representations conditioned on style and content features, achieving photorealistic results with nuanced texture preservation. While diffusion models often outperform CNNs and transformers in visual quality, their iterative nature imposes prohibitive computational costs, requiring significant processing time and memory. Due to these substantial computational demands and the limited time and resources available for this project, the same constraints that restricted the training duration of our custom models (Section III-C1), we did not include diffusion models in our experimental evaluation within *ArtEdge*. Their optimization for real-time mobile deployment remains an active area of research, but was beyond the scope of our current investigation.

D. Mobile-Optimized NST Frameworks

Recent advancements prioritize mobile constraints without compromising aesthetics. Kwon et al. proposed AesFA in 2024 [7], a lightweight framework that disentangles style and content in the frequency domain. By discarding pre-trained VGG encoders and introducing an aesthetic contrast loss, AesFA reduced inference times to 20 ms on average for 1024×1024 images [14] while boasting a low memory footprint. This approach demonstrated that domain-specific optimizations, such as frequency-based decomposition and dynamic convolution, could achieve mobile-compatible speeds without sacrificing style expressiveness. Similarly, MicroAST, by Wang et al. in 2023 [13], replaced traditional encoders with compact, custom-designed networks, achieving 522 ms inference times for 4K-resolution images with a memory trace of less than 2 MB [14], setting a critical benchmark for high-resolution mobile applications. These works underscore the importance of architectural innovations, such as depthwise separable convolutions and quantization-aware training, in bridging the gap between desktop-grade quality and mobile practicality.

These diverse approaches highlight the ongoing trade-offs between visual quality, computational efficiency, and flexibility, setting the stage for our proposed work, which aims to navigate these challenges in a mobile context.

III. Proposed Work

The core objective of the *ArtEdge* project is to develop and deploy a robust system for real-time neural style transfer (NST) directly on mobile devices. This necessitates careful selection and adaptation of NST models to balance artistic quality with the rigid computational and memory constraints of edge computing. Our proposed work involves the integration, adaptation, and evaluation of four distinct NST models, representing different architectural paradigms, into a unified mobile application framework. This section details the underlying technologies, the specific model implementations, and the overall system architecture designed for *ArtEdge*.

A. Overall System Architecture

The *ArtEdge* system is designed with a mobile-first philosophy. The workflow, illustrated in Figure 1, involves developing

TABLE I
SUMMARY OF NEURAL STYLE TRANSFER MODELS IMPLEMENTED IN *ArtEdge*

Model Name	**Base Architecture**	**Key Innovation/Principle**	**Adaptations in *ArtEdge***
MobileNet AdaIN	CNN (MobileNetV2 Encoder) + AdaIN	Custom design using lightweight encoder with AdaIN for real-time arbitrary style transfer	Designed from scratch; VGG used only for perceptual loss during training; Trained for 20 epochs
Fast Style Transfer	CNN (Feed-forward)	Perceptual loss for real-time, single-style transfer	Integrated pre-trained models; Clamped output layer activation to [0, 1] for numerical stability
StyTr^2	Transformer	Dual transformer encoders/decoder, Content-Aware Positional Encoding (CAPE)	Adapted existing codebase to resolve deprecated library functions
AesFA	CNN (Custom Lightweight)	Frequency-domain disentanglement, aesthetic contrast loss, no VGG at inference	Reworked network components and loss calculation for mobile compatibility and performance improvements

and training models using standard deep learning frameworks (primarily PyTorch) on conventional hardware. Subsequently, these trained models are converted into a mobile-optimized format, specifically Apple's Core ML (`.mlpackage`), which allows for efficient inference on iOS devices leveraging hardware acceleration like the Neural Engine. This conversion process ensures that the computationally intensive stylization task occurs entirely on the user's device, preserving privacy and enabling real-time performance.

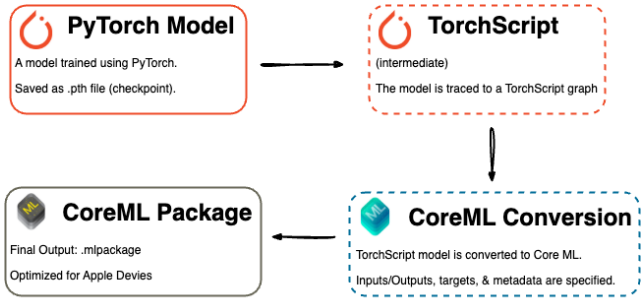

Fig. 1. System workflow for *ArtEdge*, showing the transition from PyTorch model development and training to Core ML conversion and deployment within the mobile application.

The user interaction flow within the deployed ArtEdge application follows three simple steps, as illustrated in Figure 2: selecting a style transfer model, choosing content and style images, and initiating the stylization process.

B. Core Architectural Components

Our selection of models leverages several key technologies foundational to modern computer vision and neural style transfer:

- **Convolutional Neural Networks (CNNs):** As highlighted in the literature review, CNNs, particularly architectures like VGG [2], are fundamental to NST for extracting hierarchical features representing content and style. While powerful, their computational cost often necessitates optimization for mobile use. We utilize VGG primarily within perceptual loss functions.
- **MobileNetV2:** To address the computational demands of CNNs on mobile devices, we incorporate MobileNetV2 [9]. This architecture employs depthwise separable convolutions and linear bottlenecks, significantly reducing parameter count and computational load while maintaining strong feature extraction capabilities, making it ideal for edge deployment.
- **Adaptive Instance Normalization (AdaIN):** Proposed by Huang and Belongie [3], AdaIN provides an efficient mechanism for real-time style transfer. By aligning the mean and variance of content feature maps with those of style feature maps, it allows a single feed-forward network to apply arbitrary styles without retraining, forming a core component of one of our custom models.
- **Transformers:** Moving beyond CNNs, Transformers [12] utilize self-attention mechanisms to capture long-range dependencies and global context within images. Architectures like StyTr^2 [1] adapt Transformers for NST, potentially offering improved handling of global stylistic elements and content preservation compared to purely convolutional approaches.

C. Model Implementations for ArtEdge

We implemented and adapted four distinct NST models within the *ArtEdge* framework, each chosen to explore different trade-offs between style quality, speed, and architectural complexity on mobile platforms. A summary of these models is presented in Table I.

1) MobileNet AdaIN: Recognizing the need for a highly optimized yet flexible model, we designed a novel architecture combining MobileNetV2 and AdaIN. Note that, due to brevity, this model is referred to as AdaIN for the remainder of the text.

- **Architecture:** It employs a truncated MobileNetV2 (up to layer 13) as a lightweight feature encoder, significantly reducing computational cost compared to VGG-based encoders. The core stylization mechanism is the AdaIN

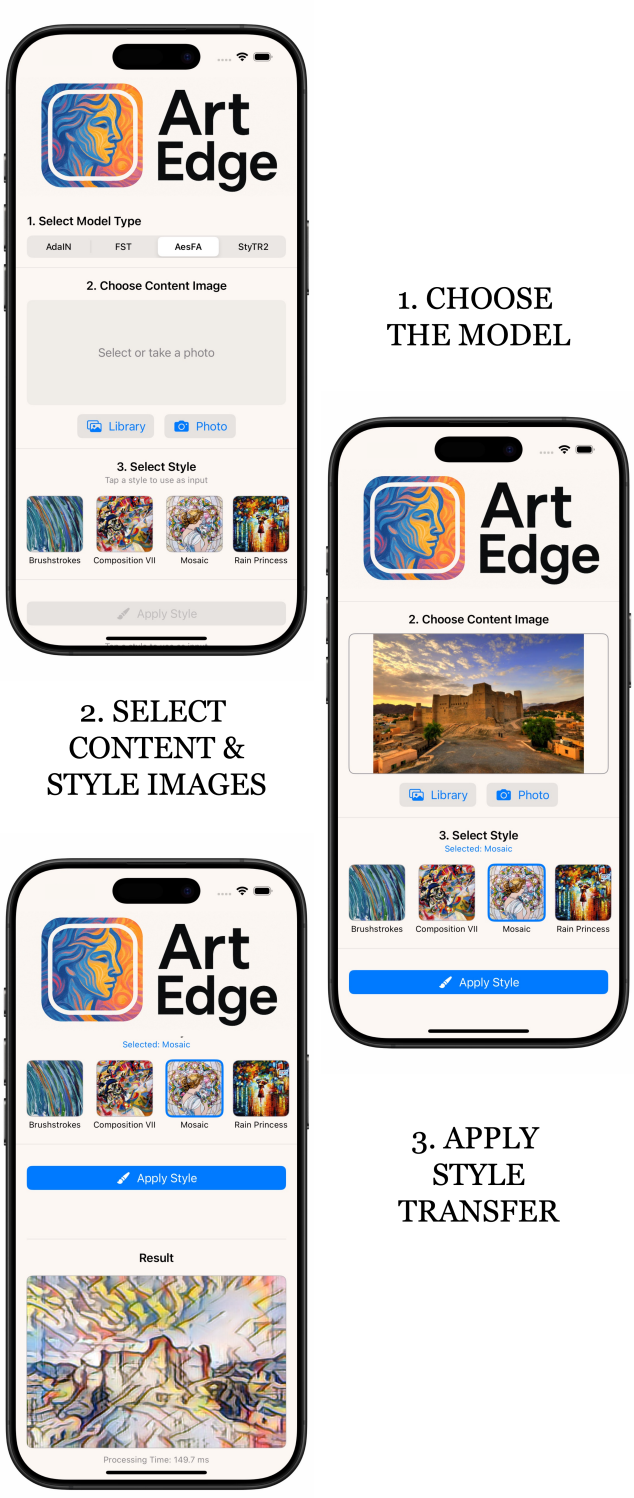

Fig. 2. User workflow in the *ArtEdge* application. (1) Model Selection View, (2) Image Selection View, (3) Style Transfer Application View.

module, which dynamically adjusts content features based on style features using the formula:

$$\text{AdaIN}(c, s) = \sigma_s \cdot \left(\frac{c - \mu_c}{\sigma_c} \right) + \mu_s$$

where c and s are content and style features, and μ and σ represent their channel-wise mean and standard deviation, respectively. A simple decoder network then reconstructs the stylized image from the AdaIN output. The overall architecture illustrating the interaction between the encoder, AdaIN module, and decoder is depicted in Figure 3.

- **Training and Loss:** For training, we utilize a separate, pre-trained VGG19 network solely to compute perceptual losses. Content loss is calculated as the Mean Squared Error (MSE) between features (e.g., `relu4_1`) of the generated image and the original content image. Style loss is computed by comparing the Gram matrices of features across multiple layers (`relu1_1` to `relu3_1`) between the generated image and the style image. This leverages the proven effectiveness of VGG for perceptual similarity while keeping the inference network lightweight. The model was trained for 20 epochs using subsets of the MS COCO dataset for the content images and the WikiArt dataset for the style counterparts. The decoder uses standard upsampling, convolutional, and ReLU layers, followed by a Sigmoid activation to ensure outputs are in the valid [0, 1] range.
- **Rationale:** This custom model aims to strike a balance between the arbitrary style flexibility of AdaIN and the efficiency required for mobile deployment by using MobileNetV2.

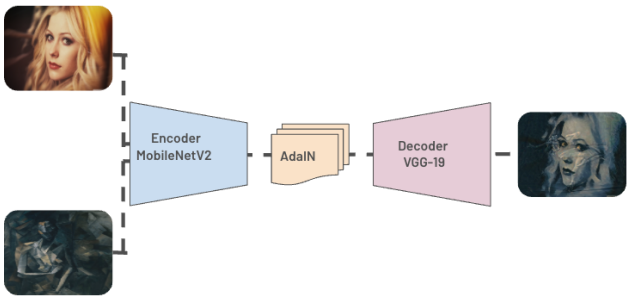

Fig. 3. Architectural design of the custom MobileNet AdaIN model implemented in *ArtEdge*, showing the flow from input images through the MobileNetV2 encoder, AdaIN stylization, and decoder.

2) Fast Style Transfer: To include a baseline representing efficient single-style transfer, we integrated models based on the Fast Style Transfer approach [6].

- **Architecture:** These are feed-forward CNNs trained using perceptual loss functions, enabling very fast inference but requiring a separate network for each artistic style.
- **Adaptations:** We utilized publicly available pre-trained models for various styles. A minor but crucial modification was made to the final layer of the transformation network: the output is explicitly clamped to the range [0, 1]. This prevents potential numerical instability or invalid pixel values that could arise during inference, ensuring compatibility with image processing pipelines.

3) StyTr^2: Transformer-Based Style Transfer: To explore the capabilities of attention mechanisms for NST, we incorporated the StyTr^2 model [1].

- **Architecture:** StyTr^2 uses separate Transformer encoders for content and style features and a Transformer decoder to merge them, leveraging self-attention to capture global dependencies. It introduces Content-Aware Positional Encoding (CAPE) for better spatial understanding in images.
- **Adaptations:** We used the official pre-trained StyTr^2 model. However, the original codebase relied on functions and modules that have been deprecated in recent versions of PyTorch and Torchvision. To ensure compatibility with our environment without compromising functionality, we replaced these deprecated components with equivalent custom functions or updated library calls. These changes were necessary to run the model within our development and deployment pipeline.

4) AesFA: Mobile-Optimized NST: Representing the state-of-the-art in mobile-specific NST, we included AesFA [7].

- **Architecture:** AesFA is designed for efficiency, using frequency domain disentanglement and avoiding heavy pre-trained backbones like VGG during inference. It employs an aesthetic contrast loss to improve visual quality.
- **Adaptations:** We performed mobile-specific optimizations on the AesFA architecture, which involved reworking certain network layers, adjusting the loss calculation mechanisms to enhance compatibility with the Core ML framework, thereby improving inference stability and performance for the target mobile hardware. Specific implementation details regarding these adaptations can be found in our project repository.

By implementing and adapting these four diverse models, *ArtEdge* provides a platform to compare different NST approaches directly within a mobile context, evaluating their trade-offs in terms of visual fidelity, inference speed, and resource consumption on edge devices.

IV. Simulation Results

This section presents the evaluation of the neural style transfer models integrated into the *ArtEdge* framework. We assess the performance of MobileNet AdaIN, Fast Style Transfer, StyTr^2, and AesFA on both a standard development platform (laptop) and the target mobile platform (iOS device). The evaluation focuses on inference speed and qualitative visual output to determine the suitability of each model for real-time mobile deployment.

A. Experimental Setup

1) Hardware Platforms: Experiments were conducted on two distinct platforms to compare performance under different computational constraints:

- **Development Platform (Laptop):** MacBook Air M1, equipped with an 8-core CPU, 8-core integrated GPU, and 16 GB RAM, running macOS Sequoia. Model execution utilized PyTorch with CPU.

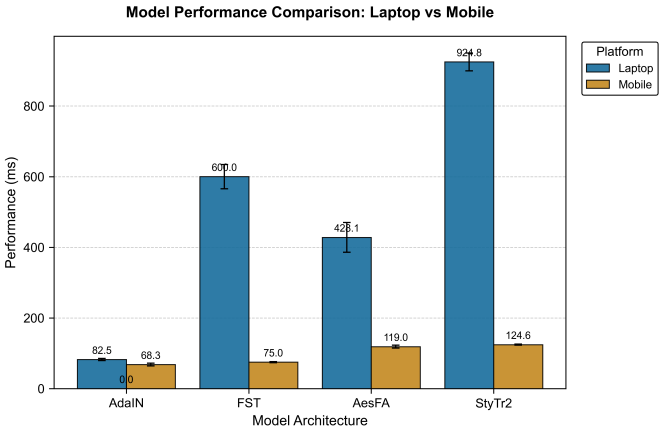

Fig. 4. Comparison of average inference times for the four NST models on the development laptop and the target mobile device across six runs for the images shown in Figure 5.

- **Target Mobile Platform (Edge Device):** iPhone 15 Pro running iOS 18, featuring the A17 Pro chip with its integrated Neural Engine. Model inference on the mobile device leverages Apple's Core ML framework for hardware acceleration.

2) Software and Implementation: The models were initially developed and trained (where applicable, like MobileNet AdaIN) using Python 3.9 and PyTorch 2.5. Pre-trained weights were used for Fast Style Transfer, StyTr^2, and AesFA, with adaptations made as described in Section 3. For mobile deployment, PyTorch models were converted to the Core ML format (`.mlpackage`) using Core ML Tools 8.2. The conversion process utilized float32 precision, preserving the original model weights without quantization at this stage. The *ArtEdge* iOS application was developed using Swift and Xcode 16.3, integrating the Core ML models for on-device inference.

B. Datasets for Evaluation

To ensure a comprehensive evaluation across diverse visual inputs, we utilized images from the following sources:

- **MS COCO (Microsoft Common Objects in Context) [8]:** A subset of the validation set was used, providing a wide range of common scenes, objects, and environments (typical resolution 640x480 px). This tests content preservation across varied inputs.
- **WikiArt [10]:** A selection of images representing various artistic styles (e.g., Impressionism, Cubism, Abstract Art) was used as style references (variable resolutions, typically 500-3000 px). This tests the models' ability to capture and transfer diverse aesthetics.
- **User-Generated Images:** A collection of custom photographs taken with mobile devices under real-world conditions (variable lighting, subjects, typical resolution 1080x1920 px) was used to simulate typical user input for the *ArtEdge* application.

For quantitative timing tests, input images were resized to a standard resolution (256x256 pixels) before being fed into the

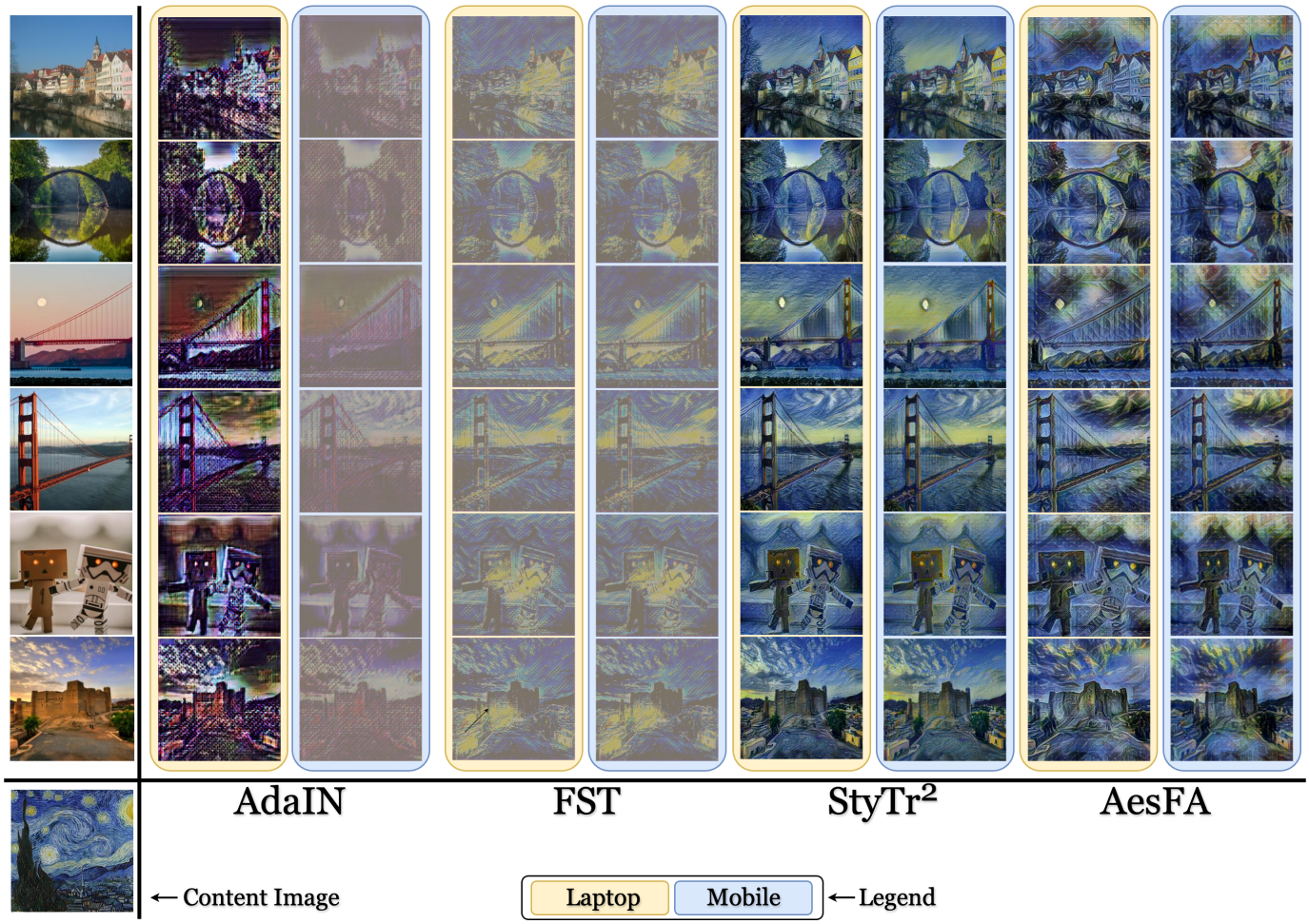

Fig. 5. Qualitative comparison of Neural Style Transfer results. Column 1: Original Content Image, Style Reference Image. Columns 2-5: Stylized outputs from MobileNet AdaIN, Fast Style Transfer, AesFA, and StyTr^2, respectively. For each model, the left column (yellow background) shows results generated on the laptop and the right column (blue background) shows results generated on the mobile device via *ArtEdge*.

models to ensure fair comparison. Qualitative examples use various resolutions to showcase real-world applicability.

C. Evaluation Metrics

We evaluated the models based on two primary criteria:

- **Inference Time (Latency):** Measured as the average time in milliseconds (ms) required to process a single image (content + style input to stylized output). Measurements on the mobile platform specifically target the Core ML model execution time, leveraging the Neural Engine. Averages were taken over 6 runs after an initial warm-up phase to exclude initialization overhead.
- **Qualitative Visual Assessment:** Judged subjectively based on several factors: fidelity to the target style (texture, color palette, brushstroke patterns), preservation of the original content structure, absence of visual artifacts (e.g., checkerboard patterns, content leakage, unnatural distortions), and overall aesthetic appeal.

While model size and memory footprint are crucial for mobile deployment, detailed profiling was beyond the scope of this initial evaluation, but remains important future work.

D. Quantitative Results: Inference Speed

The inference speeds for each model on both the laptop and the mobile platform are visualized in Figure 4.

As observed in the figure, inference on the mobile platform using the Neural Engine via Core ML generally achieves significantly lower latency compared to the laptop platform for most models, highlighting the effectiveness of specialized mobile hardware acceleration. MobileNet AdaIN demonstrates the lowest latency, consistent with its feed-forward, single-style architecture. Fast Style Transfer offers a competitive speed suitable for real-time interaction. AesFA, designed for mobile, also shows efficient performance. StyTr^2, being the most complex transformer-based model, exhibits the highest latency, although the mobile optimization still provides a substantial speedup over the laptop execution. These results confirm the feasibility of running sophisticated NST models in near real-time on modern mobile devices.

E. Qualitative Results: Visual Fidelity

Figure 5 compares stylizations produced by the four NST models on both the development laptop (yellow background)

and the mobile device (blue background). The MobileNet and AdaIN combination (column 2) yields recognizable style transfer but with relatively muted color saturation and visibly smoothed textures. This is especially apparent in the fine structural details (e.g., the bridge cables and castle battlements), which remain underrepresented. The authors attribute this limitation to our abbreviated training schedule, which consisted of only 20 epochs, and the minimal capacity of the MobileNetV2 encoder in capturing high-frequency style cues.

Fast Style Transfer (column 3) produces crisper boundaries and better preservation of content structure compared to AdaIN. The overall style strength is still lighter than in the reference; brushstroke patterns are present but lack the depth and contrast of the source style image. Notably, on the mobile device, the output exhibits only a slight drop in texture prominence, indicating that the Core ML conversion (using float32 precision) and resizing pipeline impose minimal degradation on feed-forward CNNs.

Columns 4 (AesFA) and 5 (StyTr^2) deliver the richest and most faithful renderings of the style reference. AesFA's frequency-domain disentanglement manifests in well-defined, high-contrast strokes that adhere closely to both the texture and color palette of the style image, with only minor halo artifacts around high-contrast edges. StyTr^2 further enhances global style coherence via its self-attention mechanism, yielding smooth, evenly distributed stylization across the image. On mobile, both of these methods retain the bulk of their desktop-level fidelity, although we observe a consistent slight softening of the finest textural elements. The authors credit such a drawback to likely a byproduct of repeated resizing when converting to and from the Core ML image buffer.

Across all methods, resizing steps required for Core ML input/output introduce subtle artifacts (e.g., minor boundary ringing and interpolation-induced blurring), but these effects are most apparent in the highest-capacity models (AesFA, StyTr^2) where the richness of detail makes any degradation more salient. Overall, our qualitative evaluation confirms that while lightweight methods like AdaIN and Fast Style Transfer can operate at very high speeds with acceptable visual quality, AesFA and StyTr^2 offer superior style fidelity on mobile, validating the utility of both frequency-domain and transformer-based approaches for edge-deployed NST.

V. Conclusion & Future Work

This paper introduced *ArtEdge*, a mobile-first system designed to perform neural style transfer directly on edge devices, prioritizing real-time performance and user privacy. We successfully implemented, adapted, and evaluated four distinct NST models: a custom MobileNet AdaIN, Fast Style Transfer, the transformer-based StyTr^2, and the mobile-optimized AesFA. All models were integrated within an iOS application leveraging Core ML for hardware-accelerated inference. Our experiments demonstrated the feasibility of achieving near real-time stylization on modern mobile hardware, with inference times significantly reduced compared to CPU-based execution. Key findings highlight a trade-off between speed and visual fidelity: lightweight models like MobileNet AdaIN and Fast Style Transfer offered the lowest latency, suitable for interactive use, while AesFA and StyTr^2 provided superior stylistic accuracy and detail preservation at a higher computational cost. These results validate the potential of advanced architectures on mobile platforms despite minor quality differences compared to their desktop counterparts, likely influenced by factors such as the float32 Core ML conversion and image resizing pipelines.

Building on this work, several avenues for future research emerge. Firstly, refining the custom MobileNet AdaIN model through extended training (beyond the initial 20 epochs) or architectural adjustments could improve its visual output while retaining its speed advantage. Secondly, a thorough investigation into minimizing the subtle quality degradation observed during mobile deployment, perhaps through optimized Core ML conversion techniques or exploring different quantization methods (e.g., float16 or int8) beyond the current float32 implementation, is warranted. Furthermore, conducting detailed profiling of memory footprint and power consumption for each model on target devices would provide critical insights for practical mobile applications. Exploring the adaptation of even more recent NST paradigms, such as highly optimized diffusion models tailored for edge constraints, could push the boundaries of mobile creative tools. Finally, incorporating user-adjustable parameters (e.g., style strength) and performing formal user studies would yield valuable feedback on the perceived aesthetic quality and overall user experience of the *ArtEdge* system.

VI. Summary of Contributions

Nayeem Mohammad specifically contributed by designing and implementing three of these frameworks (MobileNet AdaIN, Fast Style Transfer, and StyTr²), conducting the experiments to benchmark performance on a laptop, and generating the final Core ML model outputs necessary for iOS deployment.

Nitin Gupta contributed by implementing the AesFA model, developing and deploying the complete iOS mobile application using Xcode, and benchmarking the models specifically on the mobile platform; the laptop and mobile benchmarks were then combined for final analysis.

REFERENCES

[1] DENG, Y., TANG, F., DONG, W., MA, C., PAN, X., WANG, L., AND XU, C. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 11326–11336.

[2] GATYS, L. A., ECKER, A. S., AND BETHGE, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[3] HUANG, X., AND BELONGIE, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 1501–1510.

[4] JING, Y., LIU, X., DING, Y., WANG, X., DING, E., SONG, M., AND WEN, S. Dynamic instance normalization for arbitrary style transfer. In *Proceedings of the AAAI conference on artificial intelligence* (2020), vol. 34, pp. 4369–4376.

[5] JING, Y., YANG, Y., FENG, Z., YE, J., YU, Y., AND SONG, M. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics 26*, 11 (2019), 3365–3385.

[6] JOHNSON, J., ALAHI, A., AND FEI-FEI, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision* (2016).

[7] KWON, J., KIM, S., LIN, Y., YOO, S., AND CHA, J. Aesfa: an aesthetic feature-aware arbitrary neural style transfer. In *Proceedings of the AAAI conference on artificial intelligence* (2024), vol. 38, pp. 13310–13319.

[8] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P., AND ZITNICK, C. L. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13* (2014), Springer, pp. 740–755.

[9] SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A., AND CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 4510–4520.

[10] STEUBK. Wikiart, Sep 2022.

[11] SUSLADKAR, O., DESHMUKH, G., MITTAL, S., AND SHASTRI, P. D^2styler: Advancing arbitrary style transfer with discrete diffusion methods. In *International Conference on Pattern Recognition* (2025), Springer, pp. 63–82.

[12] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[13] WANG, Z., ZHAO, L., ZUO, Z., LI, A., CHEN, H., XING, W., AND LU, D. Microast: towards super-fast ultra-resolution arbitrary style transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2023), vol. 37, pp. 2742–2750.

[14] ZHOU, X., ZHENG, Y., AND YANG, J. Bridging the metrics gap in image style transfer: A comprehensive survey of models and criteria. *Neurocomputing* (2025), 129430.