CAN DEPTHWISE SEPARABLE CONVOLUTION MAKE NEURAL STYLE TRANSFER MORE LIGHTWEIGHT? A COMPARATIVE STUDY

A PREPRINT

Shichao Guo
Department of Computer Science
Aarhus University
au779770@uni.au.dk

December 2, 2025

ABSTRACT

This project explores how well Depthwise Separable Convolutions can lower the computing cost of Neural Style Transfer. We implemented a baseline Fast Neural Style Transfer network and three progressive lightweight variants. Our experiments on the COCO 2017 dataset demonstrate that replacing standard convolutions with DSCs can reduce the model size by up to 87.62% and achieve a 1.44x speedup on CPU inference, while maintaining comparable visual quality.

Keywords Neural Style Transfer · Depthwise Separable Convolutions

1 Introduction

Neural Style Transfer (NST), first introduced by Gatys et al. [Gatys et al., 2016], has become a popular application of deep learning, allowing users to blend the content of one image with the artistic style of another. While the original optimization-based approach was slow, Johnson et al. [Johnson et al., 2016] proposed a feed-forward network (Transformer Net) to generate stylized images in real-time. However, these networks often rely on heavy standard convolution layers, making them computationally expensive for real-time applications on mobile or edge devices.

The objective of this study is to determine if Depthwise Separable Convolutions—a technique popularized by MobileNet [Howard et al., 2017] for efficient computing—can be applied to the style transfer domain to create a "lightweight" generator without significantly compromising artistic quality. We conduct a comparative study across three different levels of architectural modification to answer the question: Can we make style transfer lighter without losing the "style"?

2 Related Work

Neural Style Transfer: Gatys et al. [Gatys et al., 2016] demonstrated that deep features from Convolutional Neural Networks (CNNs) can separate and recombine image content and style. Johnson et al. [Johnson et al., 2016] improved upon this by training a feed-forward network to approximate the optimization process, enabling real-time style transfer.

Lightweight Convolutional Neural Networks: To make convolutional neural networks more efficient and lightweight, reducing computational cost is crucial. Howard et al. [Howard et al., 2017] introduced MobileNets, which utilize Depthwise Separable Convolutions (DSC) to factorize a standard convolution into a depthwise spatial convolution and a pointwise (1×1) channel convolution. This factorization significantly reduces both parameters and computation (FLOPs).

Our work bridges these two fields by integrating DSCs into the architecture of Johnson et al., evaluating the trade-offs between efficiency and visual fidelity.

3 Methods

3.1 Dataset and Preprocessing

We utilized the **COCO 2017 Validation Set** as our content image source. Due to computational constraints, we used only the Validation Set of COCO 2017, splitting it into training, validation, and test subsets as follows:

- **Training Set:** 4,455 images
- **Validation Set:** 495 images (used for monitoring training progress)
- **Test Set:** 50 images (reserved for final evaluation)

All images were resized to 256×256 pixels, center-cropped, and added with Gaussian noise (See Table 1).

Table 1: Samples of the MS COCO 2017 Set

The style reference image used was Vincent van Gogh's *The Starry Night* (See Figure 1).

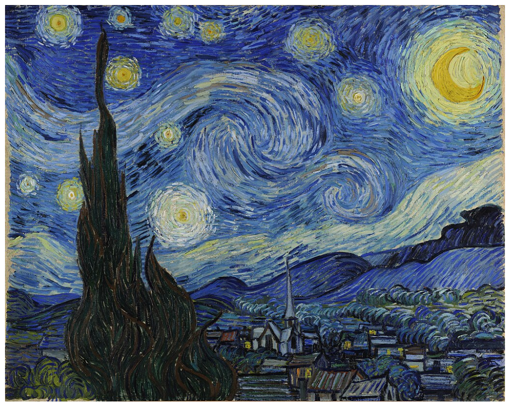


Figure 1: Style Image (Starry Night) used for Neural Style Transfer

3.2 Model Architectures

We created four versions of the model so we could clearly see how the lightweight layers affect performance.

Baseline: A standard Transformer Net consisting of 3 convolution layers (Encoder), 5 Residual Blocks, and 3 Transposed Convolution layers (Decoder), according to Johnson et al. [Johnson et al., 2016].

Lightweight Convolution Layer Design: We built a *LightweightConvLayer* that replaces the usual spatial convolution in the Baseline model. It uses a depthwise convolution with groups equal to C_{in}, then a pointwise convolution with a size of 1×1. We added Instance Normalization and a ReLU function between the depthwise layer and the pointwise layer to ensure non-linearity and stable training.

The Variants:

- **Lightweight-v1:** Replaces only the 5 **Residual Blocks** with lightweight blocks.
- **Lightweight-v2:** Replaces the **Encoder** (downsampling layers) and **Residual Blocks**.
- **Lightweight-v3 (Fully Lightweight):** Replaces the **Encoder**, **Residual Blocks**, and the **Decoder** (upsampling layers).

3.3 Training

The networks were trained to minimize a Perceptual Loss function ($L_{total} = \lambda_c L_{content} + \lambda_s L_{style} + \lambda_{tv} L_{tv}$), computed using a pre-trained VGG-16 network. The content loss ($L_{content}$) was calculated using the feature maps, while the style loss (L_{style}) was computed using Gram matrices. The total variation loss (L_{tv}) was included to encourage spatial smoothness in the output images.

- **Optimization:** Adam optimizer ($lr = 1e - 3$) with `ReduceLROnPlateau` scheduler.
- **Training Duration:** 20 Epochs.
- **Hardware:** Training was performed on a GPU (A100), while inference benchmarking was conducted on a CPU to simulate edge constraints.

4 Results

4.1 Quantitative Analysis: Efficiency

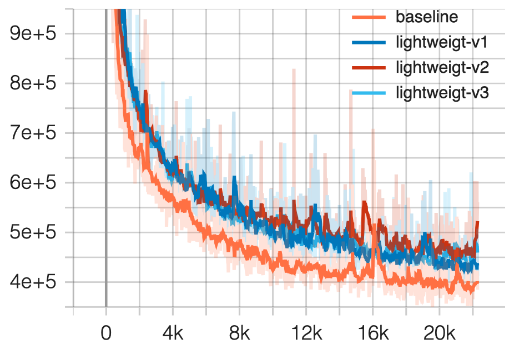
We evaluated the models based on Parameter Count (Space complexity) and Inference Latency (Time complexity). As shown in Table 2, the **Lightweight-v3** model achieved a massive **87.6% reduction in size**, shrinking the model from ∼1.68 million parameters to just ∼0.2 million. This translated to a consistent speedup, reducing inference time per image from ∼71ms to ∼49ms.

Table 2: Model Efficiency Comparison

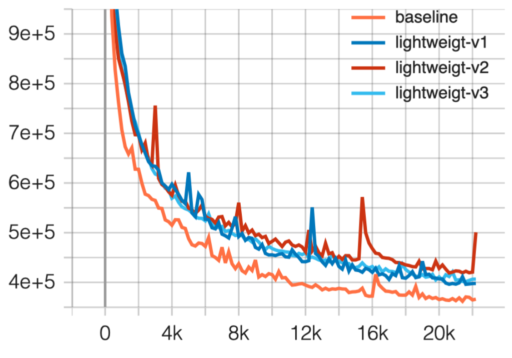
Model Variant	Parameters	Reduction (%)	Avg Latency (CPU)	Speedup
Baseline	1,679,235	-	71.31 ms	-
Lightweight-v1	381,315	77.29%	60.46 ms	1.18x
Lightweight-v2	292,796	82.56%	57.65 ms	1.24x
Lightweight-v3	207,865	87.62%	49.49 ms	1.44x

4.2 Training Dynamics

We monitored the training process by tracking the total loss (content + style + variation) on both training and validation sets. As shown in Figure 2, all models demonstrated a consistent decrease in loss, indicating successful convergence without overfitting. The lightweight models v1, v2, and v3 learned a little more slowly than the baseline, which matches their lower capacity.



(a) Training Loss



(b) Validation Loss

Figure 2: Loss curves showing consistent convergence for all models.

We also tracked the learning rate (Figure 3). The `ReduceLROnPlateau` scheduler reduced the learning rate when the validation loss plateaued, allowing for fine-grained optimization in later epochs.

4.3 Qualitative Analysis: Visual Quality

We compared the output images generated by the models on the validation set and test set.

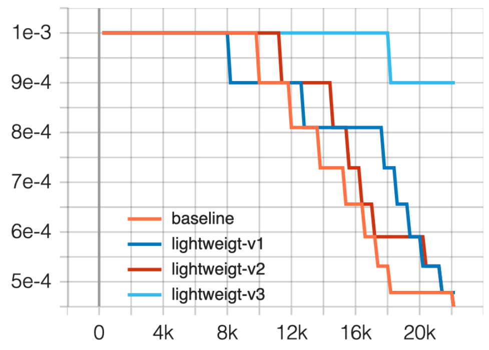


Figure 3: Learning Rate Schedule

Early Training (Step 800): As observed in Table 3, the Baseline model learned the style features much faster than the lightweight variants. At step 800, the Baseline output is already easy to recognize, while the outputs of v1, v2, and v3 still look somewhat blurry. This confirms that the lightweight models have a "Capacity Gap" and converge slower.

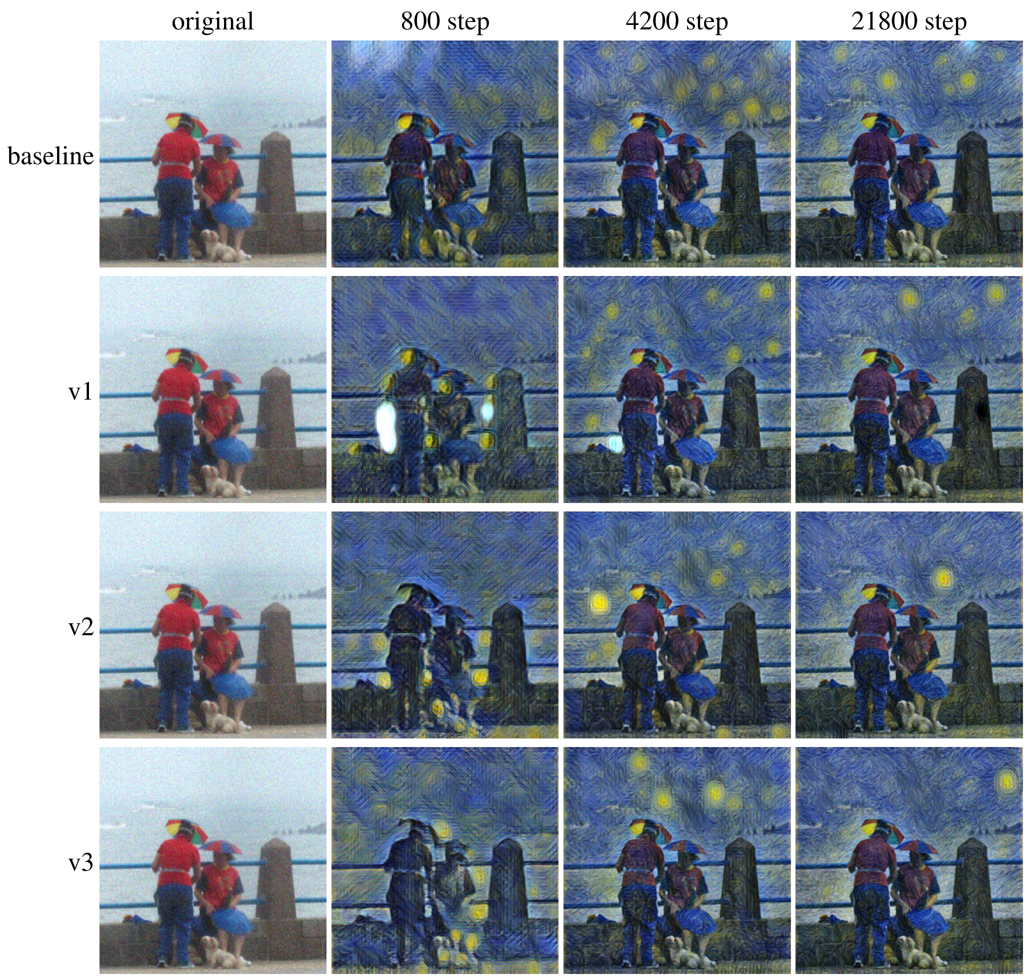


Table 3: Comparison of a validation image at different training steps.

Final Result: Despite the slower start, all models converged successfully by the end of training. Last columns (Step 4200 and Step 21800) of Table 3 shows that althougth the loss curves continued to decrease, the visual quality improvements became less noticeable. Table 4 shows the final outputs on several test images. Visually, the lightweight models (v1, v2, v3) are almost indistinguishable from the baseline in terms of global structure and color. The residual bottleneck appears to be highly redundant, as removing it caused little to no visual degradation.

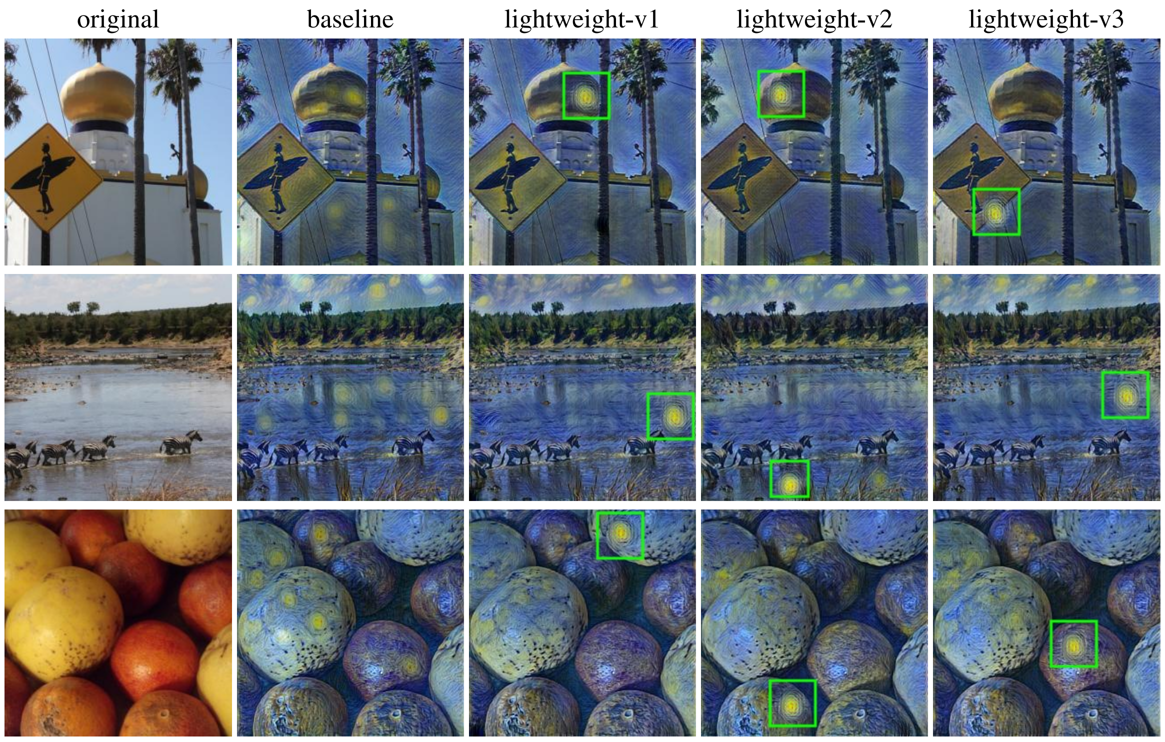


Table 4: Comparison of final stylized outputs on test images.

4.4 Specific Artifacts

A specific artifact was observed in the lightweight models: checkerboard-distributed red spots appeared within the yellow celestial orbs (See the parts in Table 4 that are marked with green squares.), which were less frequent in the baseline. This reduces the subjective quality slightly in high-contrast regions.

5 Discussion

Capacity Gap and Convergence: The lightweight models shows a slower convergence rate compared to the baseline. This is attributed to their reduced parameter capacity. However, the consistent decrease in both training and validation losses indicates that the lightweight architectures were successfully learning the style transfer task without suffering from overfitting.

Artifacts from Decoupling: We hypothesize that the "red spot" artifacts are a side effect of Depthwise Separable Convolutions. By decoupling spatial and channel-wise correlations, the network's ability to smooth out high-frequency color transitions locally might be reduced, leading to checkerboard-like patterns in specific style features.

The Efficiency Trade-off: While the lightweight variants introduced minor visual artifacts, they offered a compelling trade-off for deployment. The visual quality stopped improving after a few epochs of training, so twenty epochs were enough and training longer did not help much.

6 Conclusion

We successfully demonstrated that replacing standard convolutions with depthwise separable convolutions is a highly effective strategy for Neural Style Transfer. Our **Lightweight-v3** model offers a practical solution for deployment, achieving an **87% size reduction** and **1.44x faster inference** with minimal impact on visual fidelity. Future work can include making the model smaller with methods such as quantization, and it can also include trying to remove red spots artifacts.

References

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.