A Real-Time Anti-Aliasing Approach for 3D Applications Using Deep Convolutional Neural Network

F. M. Jamius Siam, Zahidul Islam Prince, Ahmed Nafisul Bari

Department of Computer Science and Engineering
Brac University, Dhaka, Bangladesh

Jia Uddin*

AI and Big Data Department, Endicott College
Woosong University, Daejeon, South Korea

Abstract

In real-time 3D applications, delivering smooth edges in the output images is essential, mainly due to limitations in resolution, memory, and processing power. This paper proposes a deep convolutional neural network-based model designed to address this aliasing issue. Aliasing in an image is characterized by hard, jagged edges that are present especially when the edges do not line up with the pixel grid of the output device. Our approach leverages a deep convolutional neural network to learn these jagged patterns in images from a training dataset and generates anti-aliased output images. The model's architecture includes several layers of convolutional neural networks, max-pooling layers, and convolutional transpose layers. During the experimental analysis, we used a dataset comprising demo 3D scenes created with both the Unity and Unreal game engines. This dataset contains raw and super-sampled images along with images processed with various other anti-aliasing techniques. To assess performance, we used both SSIM and PSNR scores as metrics to analyze the model's accuracy. The experimental results show that our proposed model not only competes with but often surpasses other state-of-the-art methods like MSAA, FXAA, TAA, and SMAA, by achieving higher SSIM and PSNR scores.

Contribution of the Paper: A deep learning-based anti-aliasing model trained on a per-application basis with visual fidelity approaching or surpassing traditional methods.

Keywords: Anti-aliasing, FXAA, MSAA, DLAA, NFAA, jaggedness, CNN, image-processing

ISSN: 2186-1390
http://cennser.org/IJCVSP

Article History:
Received: 30 July 2023
Revised: 16 September 2023
Accepted: 20 February 2024
Published Online: 25 February 2024

1. INTRODUCTION

People always want to experience the best possible quality images. Hence image quality plays an important role in 3D rendering industry. Researchers have developed various kinds of techniques to improve the image quality. In this context, processing an existing image with algorithms to tune the image data to create the final image is called a post-processing technique. In 3D graphics, multiple post-processing methodologies are applied to tune different attributes, for example, contrast, color, exposure, depth, etc. After rendering a 3D scene, jagged edges can be seen in objects of the final image, especially if the output resolution is not sufficiently high; these jagged edges are also known as the aliasing effect. This is an unpleasant experience for the human eyes [1], as depicted in Fig. 1. The aliasing occurs because the resolution of the rendered image is not high enough to render cleaner and sharper-looking straight lines and curves in the objects, and these lines/curves don't

*Corresponding author
Email addresses: jamiussiam@gmail.com (F. M. Jamius Siam), zahidulisprince@gmail.com (Zahidul Islam Prince), nafisulbari@gmail.com (Ahmed Nafisul Bari), jia.uddin@wsu.ac.kr (Jia Uddin)

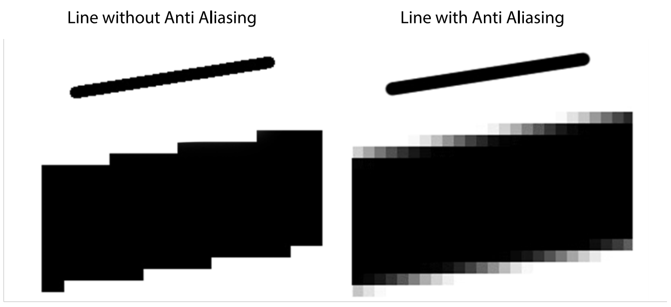

Figure 1: Render of a straight line with and without anti-aliasing

fit perfectly in a pixel grid. To the human eye, this aliasing effect is distracting because in the real world, we do not see such artifacts. As such, it is a hindrance to the experience of the medium as a whole. However, with the help of anti-aliasing methods, the effects of jaggedness can be reduced to show a pleasing natural looking image. Anti-aliasing is an image processing technique used in computer graphics to improve the quality of images and address the issue of aliasing. Anti-aliasing refers to the process of removing the unwanted aliasing artifact commonly occurring in sharp edges to achieve better quality images [2]. There are many types of anti-aliasing methods, and most of them work in a post-processing manner. However, this paper presents a sequential CNN-based deep learning anti-aliasing model for real-time 3D applications.

2. BACKGROUND STUDY

This section presents a brief introduction to the methods that are currently being used in the industry for anti-aliasing with their advantages and disadvantages compared to each other. These methods include both algorithmic and deep learning-based methods.

2.1. Super Sampling Anti-Aliasing (SSAA)

One of the most effective ways of anti-aliasing is Super sampling Anti-Aliasing (SSAA). In this technique, the image is rendered at a higher resolution than the display of the device in which the anti-aliased image will be shown, and then it is downsampled it to the native resolution of the display. Here, if a user's display resolution is 1920×1080 pixels, the SSAA method takes the image and renders it at a higher resolution, for example, 3840×2160, and then downsamples it to meet the user's display resolution which results in a sharper and clearer image without the jaggedness that would occur had the image was rendered at the native 1920x1080 resolution. The resulting image's quality is much higher and eye-soothing than the other Anti-Aliasing techniques. The major drawback of this technique is that it uses huge processing power to work, which greatly impacts the performance, making it infeasible in most real-time rendering situations [3].

2.2. Multi Sample Anti-Aliasing (MSAA)

A spatial anti-aliasing technique called Multi Sample Anti-Aliasing (MSAA) selectively super samples the edges during the render cycle [4]. It takes multiple samples, e.g., 2, 4, 8 for geometric shapes to process the higher quality images. More sample counts result in a better quality, realistic and eye-soothing image [5]. MSAA is better at solving aliasing issues at the edge of objects while preserving the texture and sharpness of the scene. However, it is still expensive computationally [6], and the effectiveness of this method is reduced when aliasing occurs within a textured surface. As a result, aliasing and other artifacts can still be seen inside rendered polygons where fragment shader output contains high-frequency components due to the calculation of multi-sampling interior polygon fragments per pixel [7].

2.3. Fast Approximate Anti-Aliasing (FXAA)

The Fast-Approximate Anti-Aliasing (FXAA), also known as Fast Sample Anti-Aliasing (FSAA) is run on a single-sample color image as a single-pass filter. It is a postprocessing technique; therefore, it saves a lot of computation power sacrificing quality, which is its disadvantage [8]. Although FXAA is effective at reducing aliasing in textured surfaces, it often leads to blurriness of the textures, hence sacrificing the sharpness and fine details of the overall image [9].

2.4. Temporal Anti-Aliasing (TXAA/TAA)

Temporal Anti-Aliasing (TXAA/TAA) is an anti-aliasing technique developed by Nvidia to provide film-like antialiased quality images with marginally increased compute power [10]. Temporal aliasing is the crawling and flickering seen in motion when playing games. The presence of vehicle wheels turning backward, the so-called wagon-wheel effect, is a common example of temporal aliasing. The basic principle of TXAA is to mix the current frame that is being rendered with frames from the past. The TXAA method uses high-quality MSAA multi-samples with frame temporal filter post-processing to achieve high-quality anti-aliasing. TXAA has two major limitations image quality-wise, ghosting and blurring caused by moving objects [11]. Modern TXAA systems produce a violent blur due to the way the colors of the current frame and background are combined. Along with that, some ghosting artifacts can be observed when objects move, especially under particular light and background conditions that make the foreground and background look alike. This is partially corrected with motion blur. Nevertheless, some of it remains near objects that move fast enough to create some ghosting but not noticeable enough to be a detriment to the image quality. Along with these, TXAA is more expensive than either MSAA or FXAA.

2.5. Subpixel Morphological Anti-Aliasing (SMAA)

Subpixel Morphological Anti-Aliasing (SMAA) works by taking into account the various patterns of lines, and curves and applying anti-aliasing in the same direction [1]. It also analyzes the subpixel layout for each pixel when doing so, whereas other post processing based anti-aliasing techniques often work on per pixel basis. As a result, it often preserves the finer details of the image. It is a little bit more demanding on resources when compared with FXAA, but generally provides a better result visually. However, being a post processing method, it performs a lot faster than sampling-based techniques like MSAA.

2.6. Deep Learning Super Sampling (DLSS)

Deep learning is a method of machine learning using a virtual neural network. Neural networks require training that gives the network representations of what it should be like. To achieve this in games, Nvidia extracts a tremendous number of aliased frames from the target game and creates the anti-aliased frame using either super-sampling or accumulation rendering. Then these paired frames are fed to Nvidia's supercomputers. DLSS reconstructs its high-quality anti-aliased output by training on a large dataset with the help of their supercomputers [12].

To take advantage of DLSS on a game, Nvidia has to provide the trained model for the specific game to the consumers via their Game Ready Driver updates. When an RTX graphics card receives the update, it is able to run DLSS for those specific supported games. The user only has to enable DLSS settings to get state-of-the-art anti-aliasing while maintaining good in-game performance. For every game to have DLSS, Nvidia has to train a new model because different games have different scenarios in 2D or 3D. Therefore, it is not possible for a model trained for a specific game to perform anti-aliasing in a different game properly, which is its noticeable disadvantage.

2.7. Normal Filter Anti-Aliasing (NFAA)

Normal Filter Anti-Aliasing (NFAA) is an anti-aliasing method where the algorithm achieves anti-aliasing by filtering out the normals, which represent the direction of the surface of a polygon [13]. NFAA improves the visual fidelity of the image by taking advantage of the normal discontinuities that occur at the edge of a polygon where aliasing is most prominent. It preserves the inner detail and texture of objects as in those parts the normal direction is continuous. Therefore, it is adept at handling complex geometric patterns and shapes of 3D objects commonly used in games. Its focus on the polygon's edge means it can perform anti-aliasing while being computationally efficient, relatively speaking.

Although this technique is not without its limitations, while it can handle complex scenarios, residual artifacts and blurring may often be noticeable when working with particularly small/fine details, narrow structures, alpha blending, and moving objects. In the case of the latter, it will often introduce temporal artifacts as described in the Temporal Anti-Aliasing (TXAA) section. Also, NFAA is generally computationally slower than both FXAA and SMAA; the latter being an improvement upon FXAA [14]. The performance characteristics of this method are also highly dependent on the capabilities of the hardware and implementation detail.

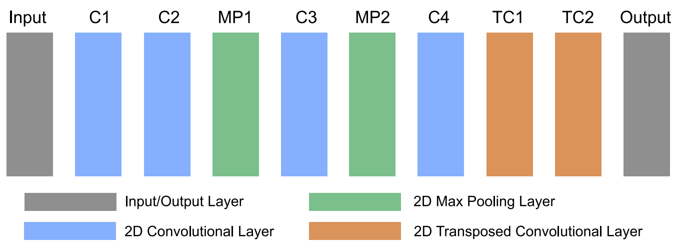

Figure 2: Architecture of the proposed model

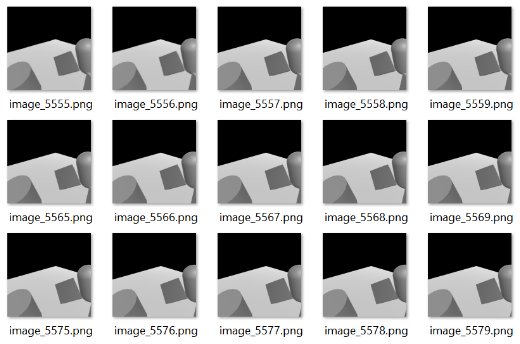

Figure 3: Some random images from the dataset

3. PROPOSED MODEL

The overview of the proposed model is presented in Fig. 2. The figure describes the different steps, such as Keras Convolution 2D, Max Pooling 2D, and Convolutional Transpose 2D, that are used in the proposed model. To train the model, we have used TensorFlow, Keras, and a sequential Convolutional Neural Network. To work, CNN requires data in the form of numbers. Such numbers are pixel values of the images. These values reflect a spectrum of grayness from 0 (black) to 255 (white) for a grayscale image. However, the 0 to 255 values are converted to 0 or 1 to feed the image data into the model. In the experimental setup, we have used 7 layers to train the model, which currently works with grayscale images only.

By using convolutional layers, the model can learn about features of the image e.g., corners, and edges. Then as we go deeper, the inner convolutional layers can learn about high frequency components that cause the aliasing effects by storing the complex relations between pixels. After that, the convolutional transpose layers can up-sample the image to the intended resolution while making sure the high-frequency jaggedness is not propagated back, effectively achieving the goal of anti-aliasing the input image.

Table 1: Layer information of the proposed model

Layer Type	Input Shape	Output Shape	Parameters
Convolution 2D	800x800x1	800x800x32	832
Convolution 2D	800x800x32	800x800x32	9248
Max Pooling 2D	800x800x32	400x400x32	-
Convolution 2D	400x400x32	400x400x64	2112
Max Pooling 2D	400x400x64	200x200x64	-
Convolution 2D	200x200x64	200x200x128	73856
Convolution Transpose 2D	200x200x128	400x400x64	73792
Convolution Transpose 2D	400x400x64	800x800x1	577

3.1. Model Architecture

In this section, we provide a detailed overview of the proposed model. In Table 1, a detailed architectural overview of the proposed model is presented. The table details all the necessary parameters for each layer of the model, along with their expected shapes and parameters.

3.2. Loss Function and Optimizer

The choice of the Loss function is vitally important in the implementation of deep CNN to get the optimal result. Several loss functions are available such as cross-entropy loss, mean-squared error, Huber, etc. [15]. However, in the implementation of our model, we utilize Eq. (1), where x is the predicted image, and y is the target image (ground truth).

$$Loss(x, y) = 1 - SSIM(x, y) \tag{1}$$

The lesser the loss function is, the more accurately the anti-aliasing is applied. For SSIM, we calculate the SSIM score of the predicted image with respect to the target image (ground truth). We will discuss about this in more detail in the Experimental Setup and Result Analysis section. To update the model with the loss function feedback and train the neural network, the Adam (Adaptive Moment Estimation) optimizer is used in the implementation [15]. Adam is an adaptive learning rate optimization algorithm and it often converges faster than stochastic gradient descent with a fixed learning rate and uses an adaptive learning rate approach to automatically adjust the learning rate for each parameter based on the historical gradients [16].

4. EXPERIMENTAL SETUP AND RESULT ANALYSIS

4.1. The Dataset

To evaluate the performance of the proposed model, two custom datasets has been generated using both the Unity [17, 18, 19] and Unreal game engine with different objects such as spheres, cylinders, and squares, etc. After that, an animation was created to move the camera around the scene. Then, to create both the input and output images, frame-by-frame images of that animation from various different camera perspectives were taken while making sure that each frame differs from each other by a substantial number of pixels because the neural network needs to understand each frame separately. If the difference between the two frames is two or three pixels, then the neural network will not be able to understand whether to anti-alias the images or transform them. The output frames were stored in a lossless format because in a lossy image, vital data about the frame is often lost, and as such, the neural network will not be able to work properly. Additionally, we can expect to receive lossless frames when working inside the game engine itself as a post-processing effect.

Fig. 3 shows a snapshot of sample images of the dataset. The output frames are in two different resolutions- 800×800 pixels and 3000×3000 pixels. The images of resolution 800×800 pixels are raw aliased input images. To give the neural network a target output to achieve from the raw aliased input images, the high-resolution images of 3000×3000 pixels are converted to 800×800 pixels using the Bicubic Sharper algorithm [20, 21]. These converted images are considered the ground truth.

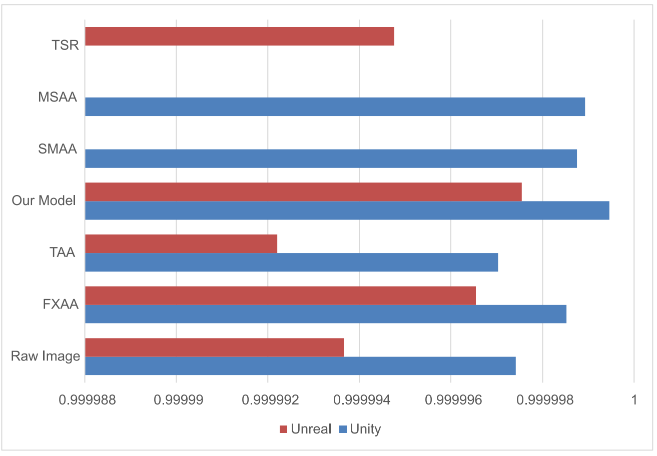

Figure 4: Average SSIM scores from the test dataset after applying different anti-aliasing methods

Table 2: Layer information of the proposed model

Game Engine	Training Set	Testing Set
Unity	5000	1000
Unreal	3700	1000

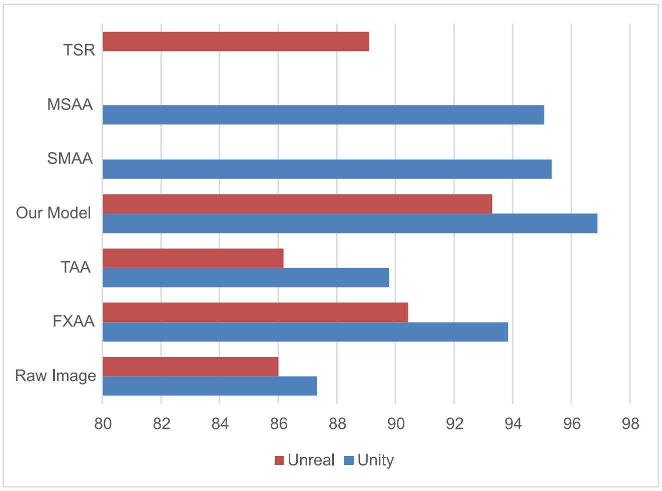

Figure 5: Average PSNR scores from the test dataset after applying different anti-aliasing methods

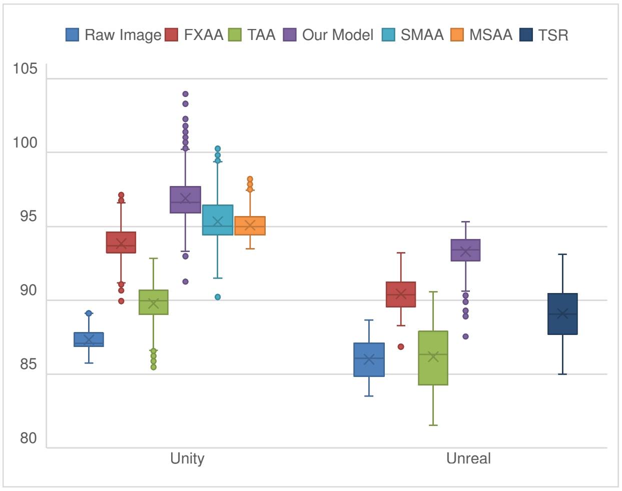

Figure 6: Average PSNR score distribution from the test dataset after applying different anti-aliasing methods

RGB images exhibit three different values for each pixel, making the required matrix calculations expensive. Therefore, in this initial iteration of our model, the input images to the neural network are converted from RGB to Grayscale color space using the Luminosity method [22]. This method forms a weighted average to account for human perception because we are more sensitive to green than other colors, so green is weighted more heavily. The formula for Luminosity is given in Eq. (2) [23].

$$Lum(R, G, B) = 0.299R + 0.587G + 0.114B \tag{2}$$

Normalization is also done before feeding the input images to the neural network. For the grayscale images, the brightness of the pixel value is a single number. The images are stored in byte array format; as a result, a single pixel is stored as an 8-bit integer which gives us a range of possible values from 0 to 255 [24]. However, machine learning works with only two values which are 0 and 1. It is because machine learning only gives a probability, the values of which range from 0 to 1. For this reason, we normalized the images to values of 0 to 1. In addition to these, the images were shuffled and flipped both horizontally and vertically at random. In Table 2, the size of the datasets used to train and test the model is detailed.

4.2. Quality Comparison

Different HVS-based (Human Visual System) methods are often used to evaluate image quality [25]. To simulate the HVS characteristics, Wang et al. proposed a model called Structural Similarity Index Measure (SSIM) [26]. SSIM has three parts: Luminance comparison $l(x, y)$, Contrast comparison $c(x, y)$, and Structure comparison $s(x, y)$. SSIM is defined as in Eq. (3) [16].

$$SSIM(x, y) = [l(x, y)]^{\alpha}.\ [c(x, y)]^{\beta}.\ [s(x, y)]^{\gamma} \tag{3}$$

The overall image quality is evaluated as mean SSIM (MSSIM), which is defined as Eq. (4), where M is the matrix containing the image information.

$$MSSIM(x, y) = \frac{1}{M}\sum_{j=1}^{M} SSIM(x_j, y_j) \tag{4}$$

From the definition of SSIM, the higher the value of $SSIM(x, y)$ is, the more similar the images x and y themselves are [26, 27].

Along with SSIM, Peak Signal-to-Noise Ratio (PSNR) is commonly used to assess the quality of image representation by measuring the ratio between the highest possible signal power and the power of distorting noise affecting the image. PSNR is typically expressed in logarithmic decibels, providing a value that ranges from the largest to the smallest possible values, with adjustments based on their consistency Eq. (6) [28, 29].

To calculate PSNR (Peak Signal-to-Noise Ratio), we employ one of the most widely used image quality measurement metrics, Mean Square Error (MSE) [28]. A lower MSE value indicates better image quality, making it a full reference metric that accounts for both variance and bias.

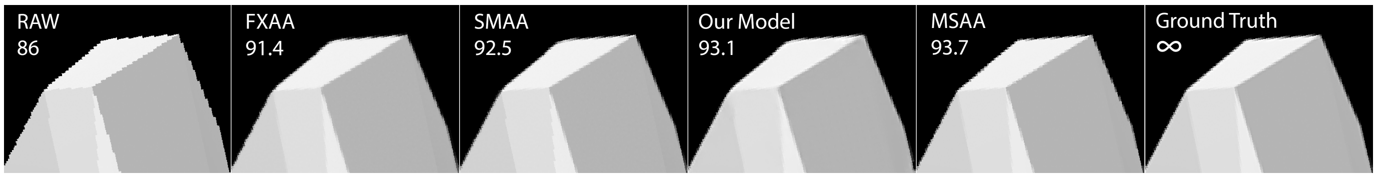

Figure 7: Comparison of different anti-aliasing methods on an unseen image depicting a monkey's face (zoomed in)

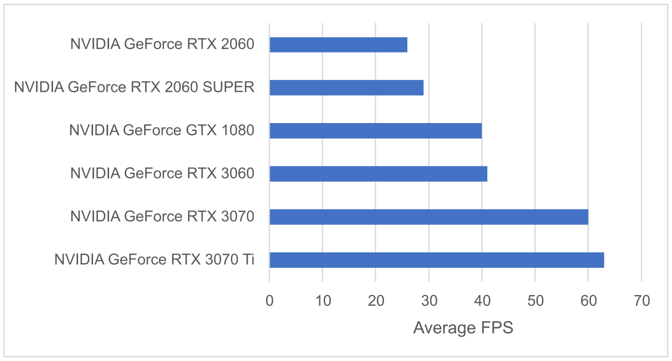

Figure 8: Average FPS of the proposed model running on different GPUs in the Unity game engine

In the case of an unbiased error, MSE represents the variance of the estimator. Given a reference image f and a test image g, both of size MN (M rows and N columns), the MSE between the two images can be expressed as Eq. (5).

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \tag{5}$$

$$PSNR(f,g) = 10\ log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \tag{6}$$

In Eq. (5), the MSE value tends to zero, and the PSNR value from Eq. (6) approaches infinity when the two images are similar, which is in line with the idea behind PSNR that a higher value provides a higher image quality. On the contrary, a small value of the PSNR implies a high numerical difference between images [30].

4.3. Image Quality Analysis

The following sub-section briefly discusses the result analysis of the proposed model. In this experimental analysis, we have used both SSIM and PSNR as the performance metrics for evaluating the image quality. Generally, SSIM output ranges between 0 to 1, and PSNR output value ranges from 0 to infinity. The higher the values of SSIM and PSNR, the better the anti-aliasing is applied. In the experiment, we calculated the SSIM and PSNR values using the Tensorflow library's image package.

In Fig. 4, the average SSIM scores from the dataset of 1000 test images are shown. MSSA is not applicable in Unreal Engine's Deferred Rendering Pipeline, and SMAA is not available by default. On the flip side, TSR (Temporal Super Resolution) anti-aliasing is an Unreal Engine specific method, hence not available in Unity.

In this test image dataset using SSIM metric, our model outperforms all the other anti-aliasing methods. This trend is also visible in the average PSNR scores as it can be seen in Fig. 5. In terms of both SSIM, and PSNR, our model outperforms other anti-aliasing methods. Along with that, from Fig. 6, we can see the distribution of PSNR scores across various anti-aliasing methods where it is evident that the scores of the metric is very stable in our proposed model indicating a better overall image quality across a number of frames. Furthermore, this trend is similar in SSIM as well.

In both SSIM, and PSNR metric, the scores across all the anti-aliasing methods are quite high. However, it should be noted that for the most part, a scene is essentially the same in both the resulting image and the ground truth image; it is only in the sharp edges that the difference starts to occur. Therefore, these small differences are significant for our purpose.

To evaluate the anti-aliased image quality of the proposed model in different scenarios apart from the testing dataset, we also experimented with a 3D model with lots of small and large details. In the Unity game engine, we created a scene with blender's monkey's head and evaluated our model with other anti-aliasing methods as depicted in Fig. 7. In this scenario, the proposed model outperforms all the other algorithms except MSAA as can be seen from the PSNR scores in the figure. However, it should be noted that our model was not trained on this level of geometrically complex scenarios.

With all of these experimental scenarios, it should be noted that all of the images used here are completely unique, and no single image was repeated across the described scenarios above.

4.4. Realtime Processing Analysis

Our proposed model can be integrated into virtually any game engine provided it supports running Open Neural Network Exchange (ONNX) models. In Unity, this is supported via the Barracuda neural network interfacing library developed by Unity Technologies [31]. Firstly, we converted our model to an `onnx` file using the `tf2onnx` library in Python. Next, as our model expects images in grayscale, we used a compute shader in Unity to convert the currently rendered frame into grayscale using Eq. (2).

Then the frame is fed to the model and the output is displayed on screen. By leveraging the GPU for all this processing, we can run our model in real-time scenarios. In Fig. 8, we can see the almost all last-gen GPUs from Nvidia 3060 and up can run our model in realtime at above 30 frames per second (FPS). Even flagship GPUs from two or three generations prior can hit 30 FPS, albeit in simpler scenes. Following this trend, all of the current generation GPUs should be able to run the proposed model at 30 FPS at the minimum, enabling realtime application of our proposed model.

5. CONCLUSION AND FUTURE WORK

In this study, we have presented a deep convolutional neural network based anti-aliasing method that consistently outperforms other industry standard anti-aliasing methods in scenarios it was trained for. In addition to that, the variance in image quality is also lower than other methods. In unseen scenarios, the model generally performs well, but is surpassed by other methods, indicating a per application basis training like DLSS [12] is the best way to implement it. Furthermore, the frame per second performance metric is well above the acceptable minimum for realtime application on a modern GPU. However, there are some constraints. The model is currently limited to grayscale image only, making it unsuitable for many real-world scenarios. Additionally, the computation cost for realtime processing is on the higher side. Despite these limitations, the experimental results are promising and further work to support color images, and model optimization for performance are the logical next steps.

References

[1] Anti-aliasing: What is it and why do we need it?, Available from: `https://vr.arvilab.com/blog/anti-aliasing`.

[2] K. Beets, D. Barron, Super-sampling anti-aliasing analyzed, Available from: `http://www.x86-secret.com/articles/divers/v5-6000/datasheets/FSAA.pdf`.

[3] J. Jimenez, J. I. Echevarria, T. Sousa, D. Gutierrez, SMAA: Enhanced subpixel morphological antialiasing, Computer Graphics Forum 31 (2012) 355–364.

[4] Multisampling anti-aliasing in hdrp, Available from: `https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@6.7/manual/MSAA.html`.

[5] J. Jimenez, et al., Filtering approaches for real-time anti-aliasing, in: SIGGRAPH'11, 2011, pp. 1–329.

[6] A quick overview of msaa, Available from: `https://mynameismjp.wordpress.com/2012/10/24/msaa-overview/` (August 2017).

[7] Multisample anti-aliasing, Available from: `https://en.wikipedia.org/wiki/Multisample_anti-aliasing` (December 2019).

[8] Fxaa, Available from: `https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf`.

[9] Nvidia fxaa anti-aliasing performance, Available from: `https://www.phoronix.com/scan.php?page=article&item=nvidia_fxaa&num=1`.

[10] A. Reshetov, Reducing aliasing artifacts through resampling, in: EGGH-HPG'12: Proc. of 4th ACM Conference on High Performance Graphics, Paris, France, 2012, pp. 77–86.

[11] C. A. O. Labrador, Improved sampling for temporal anti-aliasing (a sobel improved temporal anti-aliasing), Ph.D. thesis (2018).

[12] Nvidia dlss: Your questions, answered, Available from: `https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-your-questions-answered/` (February 2019).

[13] X. Chermain, S. Lucas, B. Sauvage, J. M. Dischler, C. Dachsbacher, Real-time geometric glint anti-aliasing with normal map filtering, Proceedings of the ACM on Computer Graphics and Interactive Techniques 4 (1) (2021) 1–16.

[14] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, S. Lacoste-Julien, Painless stochastic gradient: Interpolation, line-search, and convergence rates, in: Advances in neural information processing systems, Vol. 32, 2019.

[15] P. Isola, J. Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in: Proc. IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[16] V. Bruni, D. Vitulano, An entropy based approach for ssim speed up, Signal Processing 135 (2017) 198–207.

[17] Scenes, Available from: `https://docs.unity3d.com/Manual/CreatingScenes.html`.

[18] The scene view, Available from: `https://docs.unity3d.com/Manual/UsingTheSceneView.html`.

[19] Scene view navigation, Available from: `https://docs.unity3d.com/Manual/SceneViewNavigation.html`.

[20] Z. Dengwen, An edge-directed bicubic interpolation algorithm, in: 3rd International Congress on Image and Signal Processing, Yantai, 2010, pp. 1186–1189.

[21] R. Keys, Cubic convolution interpolation for digital image processing, IEEE Transactions on Acoustics, Speech, and Signal Processing 29 (6) (1981) 1153–1160.

[22] G. Jyothi, C. H. Sushma, D. S. Veeresh, Luminance based conversion of gray scale image to rgb image, Int. J. of Computer Science, Inf. Tech. Res. 3 (3) (2015) 279–283.

[23] Convert rgb image or colormap to grayscale, Available from: `https://www.mathworks.com/help/matlab/ref/rgb2gray.html`.

[24] Pixel values, Available from: `https://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm` (updated 2003).

[25] Y. Al-Najjar, S. D. Chen, Comparison of image quality assessment: Psnr, hvs, ssim, uiqi, International Journal of Sci., Engineering Res. 3 (2012) 1–5.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612.

[27] G. Chen, C. Yang, S. Xie, Gradient-based structural similarity for image quality assessment, in: Int. Conf. on Image Processing, Atlanta, GA, 2006, pp. 2929–2932.

[28] U. Sara, M. Akter, M. Uddin, Image quality assessment through fsim, ssim, mse and psnr- a comparative study, Journal of Computer and Communications 7 (2019) 8–18.

[29] R. G. Deshpande, L. L. Ragha, S. K. Sharma, Video quality assessment through psnr estimation for different compression standards, Indonesian Journal of Electrical Engineering and Computer Science (2018) 918–924.

[30] A. Horé, D. Ziou, Image quality metrics: Psnr vs. ssim, in: 20th International Conference on Pattern Recognition, Istanbul, 2010, pp. 2366–2369.

[31] Introduction to barracuda, Available from: `https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/index.html`.