Anti-aliasing for Real-time applications in 3D using Deep Convolutional Neural Network

by

F. M. Jamius Siam
16101234
Zahidul Islam Prince
16101172
Ahmed Nafisul Bari
16101237

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
April 2020

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

F. M. Jamius Siam
16101234

Zahidul Islam Prince
16101172

Ahmed Nafisul Bari
16101237

Approval

The thesis titled "Anti-aliasing for Real-time applications in 3D using Deep Convolutional Neural Network" submitted by

1. F. M. Jamius Siam (16101234)
2. Zahidul Islam Prince (16101172)
3. Ahmed Nafisul Bari (16101237)

of Spring 2016 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on April 7, 2020.

Examining Committee:

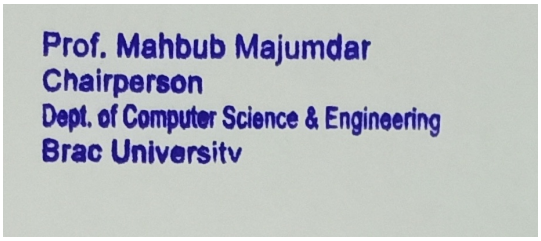
Supervisor:
(Member)

Dr. Jia Uddin
Assistant Professor (Research Track)
Technology Studies, Endicott College, Woosong University
Associate Professor (On leave)
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)



Mahbubul Alam Majumdar, PhD
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

In this paper we present a convolutional neural network model for solving the long-standing aliasing problem in the real time 3D graphics industry. Aliasing refers to the problem of having hard jagged edges in the rendered scene. These jagged edges become a distraction and on a large enough amount, creates an unpleasant viewing experience. There are quite a few techniques out there to counter this problem, namely, FXAA, NFAA, DLAA. Our neural network architecture consists of two-dimensional convolutional layers and max pooling layers for reducing the spatial dimension. We then generate the final output from transposed convolutional layer. Our model is trained on a specialized (trained on a per application basis) and generalized (trained on a variety of dataset to work on all possible conditions) version for anti-aliasing. Based on SSIM and PSNR scores we found out that a specialized version of our model works best, both in terms of visual score and image quality metrics.

Keywords: Anti-aliasing, Fxaa, Msaa, Dlss, Psnr, Ssim, Image processing, Render, Machine Learning, Convolutional Neural Network

Dedication

This research is wholeheartedly dedicated to our beloved parents who have been our source of our inspiration, supported us and gave us strength at each turn of our life. A especial feeling of gratitude towards each of our father who unfortunately are no longer a part of this world but still aspire us to be achieving great things.

We also want to remember our friends with whom we have lots of memories to cherish with.

Acknowledgement

Firstly, all praise to The Great Allah for whom our thesis have been completed without any interruptions.

Secondly, to our advisor Dr. Jia Uddin for his kind support and advice in our work. He provided us with patient guidance, encouragement and advice and helped us whenever we needed help throughout our time as his students. We have been extremely lucky to have a supervisor who cared deeply about our work, and who responded to our questions and queries so promptly.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

API Application Program Interface

BW Black and White

CG Computer Graphics

CNN Convolutional neural network

$DLSS$ Deep Learning Super Sampling

FPS Frames Per Second

$FXAA$ Fast Approximate Anti-Aliasing

GPU Graphics Processing Unit

$MLAA$ Morphological Anti-Aliasing

$MSAA$ Multisample Anti-Aliasing

$NFAA$ Normal Filter Anti-Aliasing

$PSNR$ Peak Signal-to-Noise Ratio

RGB Red Green Blue

$SMAA$ Enhanced Subpixel Morphological Anti-Aliasing

$SSAA$ Supersampling Anti-Aliasing

$SSIM$ Structural Similarity Index

$TXAA$ Temporal Anti-Aliasing

Chapter 1

Introduction

1.1 Background

From a simple image taken by a mobile camera to take an image of a galaxy situated billions of light-years away from earth, people want to experience the best possible quality images. To photographers and video editors in media industries, the aesthetics of an image is significant. Also, image quality plays a serious role in digital devices, especially in 3D rendering. Scientists have invented various kinds of techniques to improve image quality. Processing an existing image with algorithms to tune the images data and creating 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. Anti-aliasing is an image processing technique used in computer graphics to improve the quality of images. There are many types of anti-aliasing and most of them work in a post-processing manner. After rendering a 3D scene there are jagged edges seen in objects of the final image also known as Aliasing effect. This is an unpleasant experience for the human eye. The aliasing occurs because the resolution or the graphics system is not high enough to show straight lines and curves in objects. To the human eye, this aliasing effect is distracting. For example, considering jaggy effects on a tree of a 3D object are annoying to watch because we are used to seeing trees in natural form without the unnatural jaggedness. However, with anti-aliasing the effect of jaggedness can be reduced to show a pleasing image. Anti-aliasing is done in a post-processing manner.

The most popular anti-aliasing techniques are FXAA, SSAA, MSAA, DLAA. These are popular with video games. Using a little or medium processing power these techniques can deliver smoother-looking images in games. Some of these techniques often create slight blur at the edges therefore, the sharp jaggedness is less visible to the human eye. Nvidia Corporation and AMD (Advanced Micro Devices) are the two market-leading companies that mainly develop anti-aliasing technologies for their consumers. They continue to work and develop less processing power consuming anti-aliasing algorithms for a broader market. Machine learning is the process of training a computer with a large number of datasets to achieve certain goals. At the fast-growing technology market, Convolutional Neural Network has created a mechanism to train a Machine Learning model with images that enable us to perform a certain task on it. For example, face detection, object detection, pattern

finder, etc. The image analysis field benefits health, defense, agriculture, research, and many other sectors. Also, the practice and application of machine learning are growing even popular every day.

In the recent year 2019, Nvidia Corporation has developed a machine learning-based anti-aliasing which they call DLSS (Deep Learning Super Sampling). It's the world's first machine-learning based super sampling technique that also resolve the aliasing problem. Also, the first computer game to have machine learning-based anti-aliasing is Final Fantasy XV which has proven to bring brilliant aesthetics in 3D scenes. Later on, they started providing DLSS features to most of the popular games. Also, they continued to push driver update to provide better trained models that bring even greater quality to every scene. By this Nvidia has shown the world that inventing anti-aliasing techniques are not limited by algorithms and the rich capabilities of machine learning. Nvidia has also launched its RTX series graphics cards which have Tensor Cores, capable of handling Machine Learning tasks faster than the regular GPU cores. Therefore, the DLSS is only available through the RTX cards and it's a proprietary technology. Which led to developing Machine Learning based anti-aliasing technologies even harder.

On the other hand, training a machine requires constant feedback. So comparing the output of these vast amounts of pleasing-looking Anti-Aliased images manually is difficult for researchers to train a machine efficiently. Here come the SSIM (Structural Similarity Index) and PSNR (Peak Signal-to-Noise Ratio). These are the most used image comparison and evaluation techniques. These technologies allow us to compare two similar images with different quality and give us a result that allows us to know the attributes where the images differ. With these technologies combined, it is possible to train a machine learning model that will ultimately perform Anti-Aliasing and produce output images which will be more visually appealing to the human eye.

1.2 Research Problem

Different anti-aliasing techniques perform non-identically in a different scenario. Some anti-aliasing methods that perform better on the random scenario, they are computationally expensive. In fact, in computer games, gamers are allowed to choose the anti-aliasing setting for their games because not all GPU can provide playable frames per second with power-hungry anti-aliasing. The performance of anti-aliasing differs in GPUs. FXAA and SMAA are optimized in low range specification. However, the algorithm does not attempt to come up with a suitable solution to aliasing every time [1]. FXAA has trouble correcting aliasing in diagonal edges but performs well on vertical and horizontal lines but this collision creates an unwanted blurry effect which ends up creating a loss in detail in the image and giving unpleasant experience. Besides, mobile devices have low power chip so most anti-aliasing with high GPU impact has no application in them. Also to mention, these high-performance impactful techniques provide the best aesthetics. As a result in mobile phones, applying the best possible anti-aliasing is a big challenge.

Moreover, some anti-aliasing methods are memory intensive. MSAA being one of

them, therefore, GPUs with high computation speed but less memory capacity is not suitable to provide quality results in 3D rendering. SSAA can provide the best quality in every 3D scenario because it upscales the resolution. However, for upscaling, SSAA has to use increase the pixel count which ends up consuming a lot of memory space and throttles the output in FPS. In some cases, it can bring FPS down under 30. To gamers without a minimum FPS of 30, a game is considered not playable. If a game with FXAA runs at a certain FPS, changing the anti-aliasing settings to memory-intensive TXAA, drops the FPS to nearly half. Also, in mobile devices, memory-intensive anti-aliasing techniques are rarely used.

Besides, Nvidia's DLSS creating a revolution in anti-aliasing through machine learning. However, being a proprietary technology it limits the research scope. As technology is rapidly advancing, machine learning is getting more popular and being used in vast sectors. Also, graphics cards are focusing more on providing extra sets of cores that specialize in machine learning-related tasks. The competitor AMD does not have a machine learning based anti-aliasing but to push the limit of performance in artificial intelligent related tasks they introduced Radeon Instinct series GPUs to the market. As a result, it is the best time, image processing and improving image quality-related task to be done via machine learning.

Therefore, Machine Learning based anti-aliasing will benefit not only games but also image and video production industries, medical sector, defense, research, etc. As the world moving toward making tasks easy through machine learning. It is the best time image processing and improving image quality-related task to be done via machine learning too.

1.3 Research Objective

The research aims to improve image quality in 3D rendering, removing the unpleasant jaggedness effect with anti-aliasing using a deep convolutional neural network that will output good quality anti-aliased images in real time. The research objectives are as follows:

- To generate samples of images with high-quality anti-aliasing applied and non-anti-aliased images
- To get rid of the jaggedness caused by sharp edges in 3D rendered scenes
- To create a model using CNN that can apply anti-aliasing
- To be able to compare images and provide feedback to the machine to increase throughput
- To evaluate the outcome and accuracy of the model
- To introduce a machine learning-based anti-aliasing
- To apply anti-aliasing in real time

1.4 Scope and limitations

The detection of patterns is machine learning is previously conducted by many researchers, in our case, the aliased effect can be detected with CNN and with the high-quality samples of anti-aliased images it is possible to train the machine in producing high-quality images as well. The SSIM and PSNR can provide the metrics with which we can tune the learning process. The quality determination matrix will provide feedback on CNN and the model can figure out the desired output. The trained model can process a grayscale image and perform anti-aliasing efficiently for now. The larger number of datasets can help this research attain better-resulting output. Further, the model can be trained with different datasets to improve image quality in other sectors.

This research is used for grayscale images for now as grayscale images require less amount of data to be processed which is still a lot of bits of information compared to training a machine with text related work. Furthermore, with a massive number of datasets of RGB image sample and combining many high-performance graphics cards can result in a better model.

1.5 Thesis Orientation

The rest of the thesis is organized as the following order:

- Chapter 2 - Literature Review where previous and related works of the same problem are discussed and reviewed.
- Chapter 3 - Proposed Approach, here we discuss our proposed model approach to the research.
- Chapter 4 - Includes the Dataset Analysis where we discuss our dataset in details.
- Chapter 5 - Methodology where we discuss the used technologies and workflow of our work in details.
- Chapter 6 - Includes Experimental Result Analysis, where we visualise our data and analyse the result obtained from our model.
- Chapter 7 - Conclusion and Future works where the conclusion consists of our work till now and future works includes the scope of improvement.

Chapter 2

Literature Review

There are two main types of Aliasing, Spatial Aliasing, and Temporal Aliasing; at the cost of increased rendering time, solutions for the improvements against the jaggedness created by either of those types are provided by anti-aliasing [2].

Currently, the top most used anti-aliasing techniques are Supersampling Anti-Aliasing (SSAA), Multi-Sampling Anti-Aliasing (MSAA), Temporal Anti-Aliasing (TXAA) and Fast-Approximate Anti-Aliasing (FXAA).

Which type of anti-aliasing is feasible to apply for real time applications is determined by the rendering time. Also, based on the architecture and behavior of an application, we decide which technique to use. For example, old anti-aliasing solutions do not work with Deferred Shading. In the Figure 2.1 [3], we can clearly see the jaggedness in the first drawn line and how anti-aliasing smoothens it out in the second line.

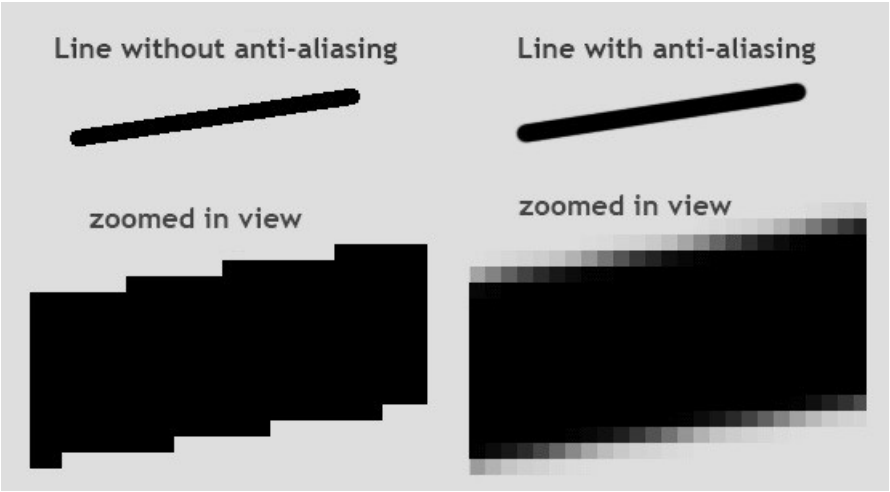


Figure 2.1: Output of Anti-Aliasing

2.1 Previous Anti-Aliasing Techniques

Let's discuss some of the previous work on anti-aliasing that has been done to make the realm of 3D computer graphics better at every turn. There are many of them out there that we have not mentioned yet above.

2.1.1 Supersampling Anti-Aliasing

The most efficient way of anti-aliasing, but also the most demanding one, is Supersampling Anti-Aliasing. In this technique, the image is rendered at a higher resolution than the display and then downsamples it to the resolution of the display. Here, if a user's display resolution is 1920x1080, the SSAA method takes the image and renders it at a higher resolution i.e. 3840x2160 and then downsamples it to meet the user's display resolution which results in a sharper and clearer image without looking pixelated as before. The resulting image's quality is much higher and eye-soothing than other Anti-Aliasing techniques but the only drawback of this technique is it uses huge processing power to work which greatly impacts the performance in real-time rendering situations [4].

There are two varieties of SSAA. One being Rotated Grid Super-Sampling (RGSS) and another being Ordered Grid Super-Sampling (OGSS). A sub-sampling grid is employed by RGSS which is around vertical and horizontal offset axes rotated and typically by 20 to 30 is used in OGSS. RGSS has a basic advantage over OGSS which is more effective anti-aliasing near the horizontal and vertical axes, where the human eye can most easily detect screen aliasing popularly known as jaggies. This advantage permits RGSS to use fewer sub-samples to achieve approximately the same visual effect is OGSS. SSAA is found to be an expensive process that eventually reduces graphics processing performance which is typically by a substantial margin. However, the effect of anti-aliasing on image quality is significant. Despite the performance cost, it is seen to be very important to an improved and worthy gaming experience.

Another type of super-sampling is Jittered Grid Super-Sampling (JGSS). It is similar to OGSS in the sense that extra samples are stored per pixel, but the difference between the two is the position of the sub-samples. The sub-sample grid is parallel and aligned to the horizontal and vertical axis with the OGSS. However, the sub-sample grid is jittered or shifted, off of the axis with the JGSS [1].

2.1.2 Multisample Anti-Aliasing

When quality and performance balance is in question, MSAA is the most common and type of anti-aliasing. MSAA stands for "Multisample Anti-Aliasing". This anti-aliasing technique selectively super samples the edges during the render cycle. It works by manipulating color around the shapes which are geometric and by which it produces an effect of smoothness. MSAA is better at solving aliasing issues than the other techniques, but it is much more intrusive and more expensive. Crucially, MSAA solves spatial aliasing issues [5]. It takes multiple samples i.e. 2, 4, 8 for geometric shapes to process higher quality image. More sample counts result in a better quality, realistic and eye-soothing image [6].

For color and depth, this method uses different multiple samples per pixel, in the time of main rendering process. MSAA also stores these samples in the framebuffer. Once the render is completed, the algorithm then reduces these samples to a single value per pixel. An implementation of MSAA requires a sequence call that writes all the additional samples back to DRAM and then reads them back into the GPU where the GPU resolves them into a single value per pixel [2].

The advantages of MSAA is that the pixel shader usually only needs to be evaluated once per pixel. Among other advantages, another is that the polygons' edges are anti-aliased. Along with some great advantages, there are some disadvantages as well. Aliasing and other artifacts can still be seen inside rendered polygons where fragment shader output contains high frequency components because multi-sampling calculates interior polygon fragments only once per pixel [7].

MSAA can be multiple times more intensive in certain scenarios such as scenes which are heavy in complex fragments, than post processing anti-aliasing techniques such as FXAA, SMAA and MLAA for a given frame while being less performance intensive that SSAA. In this category, early techniques tend towards with a lower performance impact but suffering from accuracy problems [8].

2.1.3 Temporal Anti-Aliasing

Temporal Anti-Aliasing(TXAA) is an anti-aliasing technique developed by Nvidia to provide maximum quality anti-aliased images and games output with a cost of better performance. Temporal aliasing is crawling and flickering seen in motion when playing games. It is induced by a scene sampling rate that is too low compared to the transition rate of artifacts within the scene. The presence of vehicle wheels turning backward, the so-called wagon-wheel effect, is a common example of temporal aliasing in the video. The basic principle of TXAA is to mix the current frame being rendered with frames from the past. By losing a few frames per second (FPS) in games, the TXAA method uses high-quality MSAA multi-samples then post-processes it. Also, per frame Temporal filters are combined to provide the highest quality images [9]. There are some technologies on which this technology is based on which are temporal filter, hardware anti-aliasing and custom CG film-style anti-aliasing resolves. To filter any given pixel on the screen and to offer the highest quality filtering possible, a great number of contribution of samples, on both outside and inside of the given pixel, collectively with samples from prior frames, are used by TXAA. For example, TXAA begins to approach and often exceeds the efficiency of other high-end qualified anti-aliasing algorithms, on fences or foliage and in motion.

TXAA is only available on Nvidia graphics cards. TXAA has increased spatial filtration relative to 2x MSAA and 4x MSAA requirements. The output effect of TXAA may be slightly different depending on the type of shading that is applied in a given game. TXAA has two major limitations, ghosting and blurring by moving objects. Modern TXAA systems produce a very violent blur due to the way the colors of the current frame and background are combined. Some Ghosting is created 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 slow enough to avoid Motion Blur [2].

2.1.4 Fast Approximate Anti-Aliasing

Fast Approximate Anti-Aliasing (FXAA) also known as Fast Sample Anti-Aliasing (FSAA) is developed by Timothy Lottes at Nvidia Corporation. On a single-sample color image, FXAA is run as a single-pass filter. It is a post-processing technique; therefore, it saves a lot of computation power sacrificing quality. It disregards polygons and line edges, and basically breaks down the pixels on the screen. It has two advantages. One being that in all the pixels on the screen, it smooths edges which includes the inside alpha-blended textures and those resulting from pixel shader effects, which were previously sensitive to MSAA effects with no unusual workarounds. Second one is that it is super-fast. On a video card as expensive as U.S. 100 Dollars, the FXAA algorithm of 3rd version, takes about 1.3 milliseconds per frame. If looked closely, we are getting a considerable reduction in aliasing for only a modest 12 or 13 percent cost in framerate which is because earlier versions of FXAA were seen as twofold the speed of that of 4x MSAA [10].

Non-linear RGB color data are given into FXAA as input which it converts internally. FXAA converts the input into a luminance scalar estimate for shader logic. After that, to avoid processing non-edges the algorithm checks for local contrast.

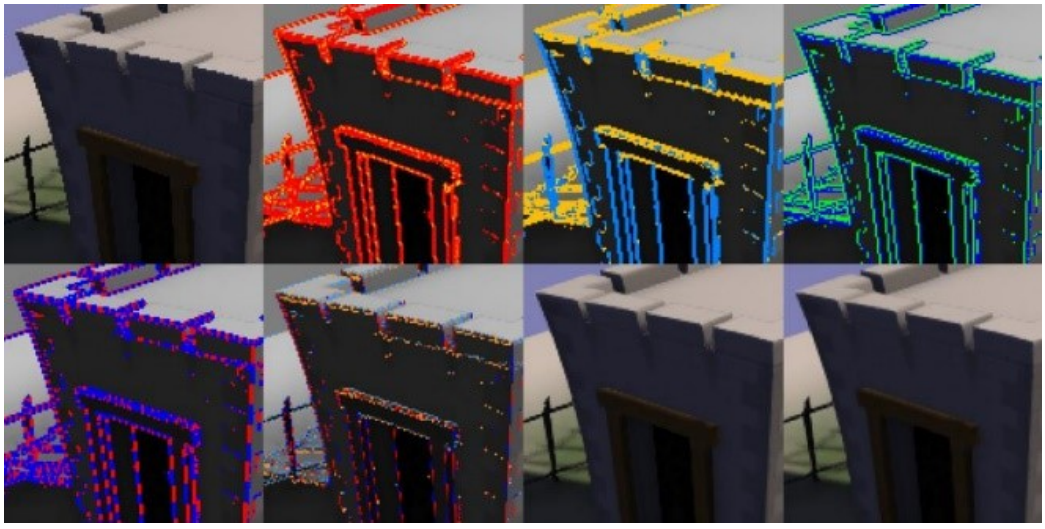


Figure 2.2: Detected Edges in FXAA

Figure 2.2 illustrates perfectly how FXAA works [11]. When FXAA is trying to detect edges, the algorithm is marking them with Red color. Similar to this, for horizontal discontinuation it is marking them as Green color and for vertical discontinuation, it is marking them as Blue color.

To represent the amount of detected sub-pixel aliasing, detected edges are red, with blending towards yellow. Pixels passing the test of local contrast are then marked

as gold or blue in horizontally or vertically respectively. Given the orientation of the edge, the pixel of highest contrast pair being 90 degrees to the edge is selected, in blue/green. After that the algorithm searches for end-of-edge in both the negative and positive, red/blue, directions along the edge and checking for a significant change in average luminance of the high contrast pixel pair along the edge. On the edge, pixel position is transformed into to a sub-pixel shift 90 degrees perpendicular to the edge to reduce the aliasing, red/blue for -/+ horizontal shift and gold/sky-blue for -/+ vertical shift. After that the algorithm resamples the input texture given this sub-pixel offset. Finally, the algorithm blends in a lowpass filter depending on the amount of detected sub-pixel aliasing [12].

2.1.5 Coverage Sampling Anti-Aliasing

The graphics system has modes of operation where real samples and virtual samples are generated with the GPU. In Coverage Sampling Anti-Aliasing(CSAA) the virtual sample's pixel is compared with the real sample's pixel. If the pixel is proximate to the neighboring pixel it detects a polygon. If the pixel is not proximate to the neighboring pixel, it detects as polygon's edge and uses that information to weight the calculation when determining the final color of the pixel [13]. The sampling threshold is selected by the user. CSAA can operate by comparing every 2, 4, 8 or higher amounts of pixel radius of the working pixel. These are known as accordingly 2x CSAA, 4x CSAA, 8x CSAA or higher sampling CSAA. In this anti-aliasing approach, a pixel includes: the estimation of rudimentary distribution over a single real sample position per pixel and at least one simulated sample location per pixel, for each virtual sample position within a pixel, the virtual sample is marked as belonging either to that pixel or to a surrounding pixel and using simulated sample coverage information to change actual sample weights with real neighboring pixel samples.

Coverage samples are easy for the hardware to gather therefor it can increase image quality at small performance overhead compared to MSAA. However, the value of coverage samples depends on the pixel's neighbor so the resulting image's quality can range from some to no change at all. This technique is developed at Nvidia Corporation to enhance picture quality in games at a dynamic performance impact, where low-end to high-end GPUs can use different sampling rates to increase image quality at different rates and users can enjoy the Anti-Aliased output. At the current time, this technique is not much widely used in computer graphics.

2.1.6 Enhanced Quality Anti-Aliasing

A post-processing low-complex anti-aliasing filter is Enhanced Quality Anti-Aliasing (EQAA). It is AMD's updated version of MLAA for their HD 6900 series graphics cards and onwards. For DirectX apps, AMD's EQAA feature is available or can be pushed through the software. EQAA offers greater Anti-Aliasing than FXAA and MSAA. To smooth the aliased edges this technique uses a similar mechanism to CSAA but in here, the color and depth pieces of information are stored in memory [14]. EQAA also resolves the polygon edges and with the stored information it determines the threshold to manipulate the color value of the working pixel. Then the

results are merged together to produce the final image. The resulting image gets a bump in quality but compared to SSAA it is not so eye-soothing but it enhances image quality over the MSAA by adding more coverage samples for each pixel by processing the exact number of color samples.

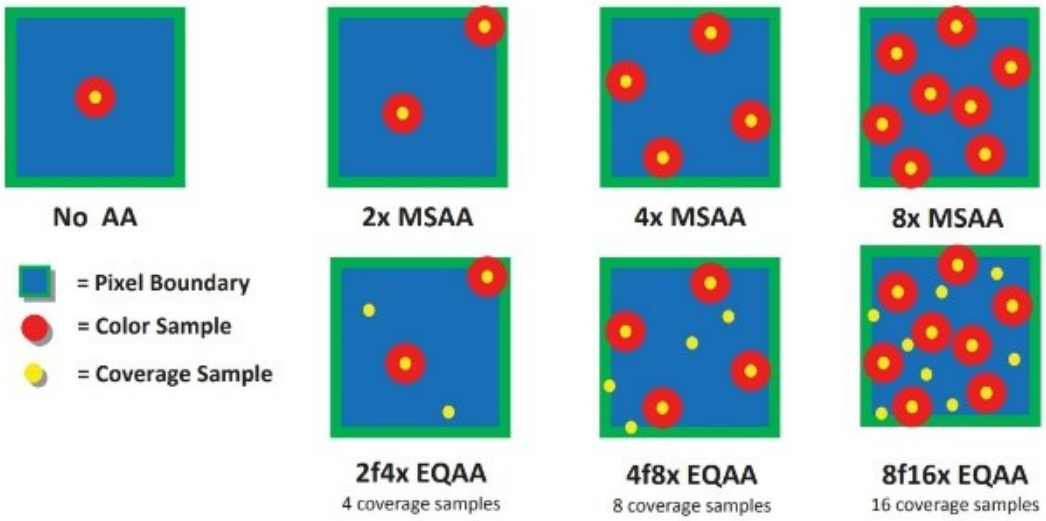


Figure 2.3: MSAA and EQAA color/coverage sample patterns

The anti-aliased image shown in Figure 2.3 [15] is considered one of the sharpest. In this case, EQAA does a good job reducing the jaggedness. In spite of the image quality, it operates at a fraction of the computational cost of true SSAA because coverage samples can be gathered more easily. However, storing extra data in memory is expensive and using higher pixel sampling i.e. 16x EQAA, to blend the colors are also computationally expensive to produce a good quality image. Thus, this technique is not much used with GPUs having lesser memory capacity. As we are using machine learning to anti-alias the images so all these algorithm-based post-processing techniques will be less related to build our model. However, Nvidia's DLSS is a proprietary technology that is mostly related to our work.

2.1.7 Deep Learning Super Sampling

DLSS stands for Deep Learning Super Sampling. Deep learning is a method of machine learning using a virtual neural network. In other words, a multimedia guide to how the brain's neurons know and solve complex problems. This is available only on Nvidia's RTX graphics cards. Where they have Tensor cores. Tensor cores are a special set of cores designed to speed up machine learning tasks. These cores take advantage of DLSS and perform Anti-Aliasing in games. Neural networks require training that gives the network representations of what it should be like. For example, if you want to tell the network how to recognize a person, you are showing it millions of faces, helping it to know the attributes and trends that make up a standard face. If the lesson is well taught, you can show any picture with a face in it, and it will pick it up automatically. To achieve this in games, Nvidia extracts the tremendous number of aliased frames form the target game and match the perfect

frame using either super-sampling or accumulation rendering. Then these paired frames are fed to Nvidia's supercomputers. The supercomputer trains the DLSS model too recognize aliased input and generate its high-quality Anti-Aliased output [16]. Now the model is ready to perform super quality Anti-Aliasing with the Tensor cores and this also reduces the performance impact on CUDA cores.

To take advantage of DLSS on a game, Nvidia has to provide the trained model for the specific game to the consumers. This is done through the Game Ready Driver updates. When an RTX graphics card receives the update for a game, it is able to benefit from DLSS for the specific game. The user only has to enable DLSS settings on in-game settings and enjoy the top of the line Anti-Aliasing and still maintain solid FPS. For every game to have DLSS Nvidia has to train a new model because different games have different scenario and they can be 2D or 3D so it is not possible for a model trained for a specific game to perform Anti-Aliasing in a different game properly. In our implementation, we also train our model to work in a specific environment and use our technology usable to train new models to implement on different scenarios.

Chapter 3

Proposed Approach

From the statistics of past few years, it can be said that no one can stop the gaming industry now. By the year of 2020, the video games industry is expected to be worth over U.S. dollars of 90 billion, from 78.61 billion in 2017. There were 2.5 billion video gamers in the world in 2016 alone and the number is going higher ever since. The Asia Pacific region earned a revenue of 51.2 billion U.S. dollars which made them the largest gaming market in 2017 [17]. In the gaming industry, some real-time gaming platforms which are relatively new in the world are also gaining popularity. Such platforms are AR and VR. AR short for Augmented Reality is a real-world environment's interactive experience where the real-world objects are enhanced by perceptual information generated by computer, sometimes across multiple sensory modalities, including visual, auditory, haptic, somatosensory and olfactory [18][19][20]. An example of Augmented Reality is the popular game Pokémon GO. On the other hand, VR short for Virtual Reality is an experience that is simulated completely by the computer and the experience can be similar to or completely different from the real world [20][21]. There are many popular VR games out there. However, among them some favorites are Batman: Arkham VR, Half-Life: Alyx etc. These gaming platforms along with mobile and pc, they all have to face the wrath of pixelation.

In computer graphics, pixelation is caused by displaying a bitmap or a section of a bitmap at such a large size that individual pixels, small single-colored square display elements that comprise the bitmap, are visible. Such an image is said to be pixelated. Early graphical technologies such as video games ran with a limited number of colors at very low resolutions, resulting in pixels that were easily visible. The resulting sharp edges gave an unnatural appearance to curved objects, and diagonal lines. However, as the amount of colors available increased to 256, it became possible to use anti-aliasing gainfully to smooth the appearance of low-resolution artifacts, not to remove pixelation but to make it less distracting to the eye [7]. In the realm of 3D computer graphics, a huge problem is pixelation. Here, to polygons, bitmaps are applied as textures. As a camera tries to reach a textured polygon, by creating drastic pixelation, nearest neighbor texture filtering would simply zoom in on the bitmap and pixelation leads to aliasing. How pixelation leads to aliasing of images is very primal because when we look at the pixelated image, we can see the pixels of the image with a clear sense of jaggedness. This jaggedness can be removed with anti-aliasing.

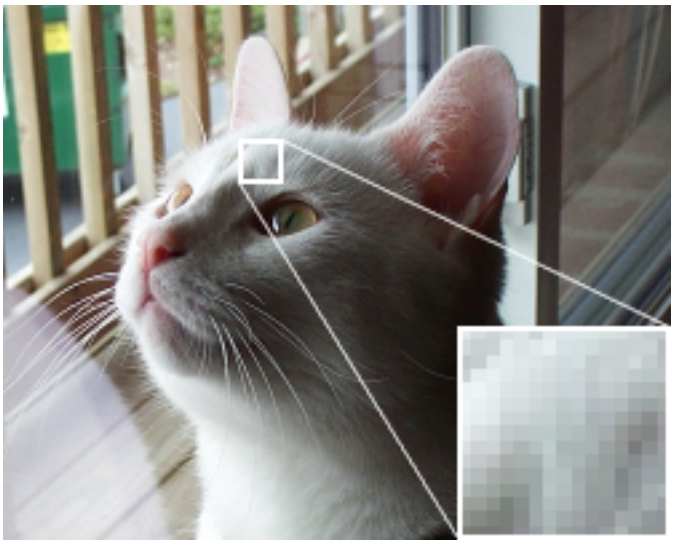


Figure 3.1: An example of pixelation and aliasing.

We can see from Figure 3.1 [22] that the image looks very smooth when zoomed out. However, when a particular portion is viewed closely, we can distinguish individual pixel which describes the concept of pixelation.

3.1 First Machine Learning Approach to Anti-Aliasing

To remove aliasing there were several anti-aliasing techniques were invented that we discussed in the literature review section. However, as video games got better by the day, the anti-aliasing techniques had to be better as well. The anti-aliasing algorithms that are generally used are algorithm based. Meaning that the anti-aliasing techniques are not context aware. Different scenes with different color range and depth does not matter to these algorithms. However, they get the job done, but they are not efficient. They cannot make an intelligent decision based on a particular photo or a scene. To solve this problem, Nvidia first made an approach that is completely different than other conventional anti-aliasing techniques. The anti-aliasing that was made by Nvidia is called DLSS which is short for Deep Learning Super Sampling which we discussed in Chapter 2 [23]. Now, let us dig a little deeper to DLSS. Nvidia Turing and the Nvidia RTX platform launched innovative new ways of integrating rasterization with both ray tracing and deep learning (aka AI). DLSS is the first in a new line of strategies that leverages RTX's deep learning and AI features to provide the game developers and gamers with modern rendering technologies. Artificial Intelligence and Machine Learning is used by DLSS to create an image that looks like a higher resolution image, without the rendering overhead. Nvidia's algorithm relies on hundreds of thousands of rendered image sequences generated using a supercomputer. That teaches the algorithm to be able to generate equally stunning images, but without the graphics card being needed to work as hard to do so.

DLSS is the final result of a series of events of teaching Nvidia's A.I. algorithm to produce better-looking games. After rendering the game at a lower resolution, DLSS gathers information from its prior knowledge based on super resolution image training to generate an image that looks like it was generated at a higher resolution. The idea is to make an illusion. The illusion is to make an image rendered at 1440p look like it was rendered at 4k or an image rendered at 1080p look like it was rendered at 1440p. When a game is being processed by DLSS, it is forced to be rendered at a lower resolution (typically 1440p) and then it uses its taught A.I. algorithm to infer what it would look like if it were rendered at a higher resolution (typically 4k). DLSS achieves this by utilizing Nvidia's own anti-aliasing technique which is TXAA and some automated sharpening. Other things that are brought in are some details which would be present in an image of higher resolution and some artifacts are removed which would not be present in an image of higher resolution. In effect, DLSS is a real-time version of Nvidia's another technology which is screenshot-enhancing Ansel technology. It is because the technique renders an image at a lower resolution for providing a performance boost but then applies various effects and techniques to deliver a relatively better overall effect to raising the resolution [24].

3.2 Choosing ML for Our Research

As good as the DLSS technique from Nvidia is, its propriety is private. They refused to make it public or open source and as it's a method which is game by game basis, we wanted to make a technique which would be game by game basis rather it would be universal for all games or scenes. Along with this, we wanted to do our research using Machine Learning (ML) as it is the future of the world. Almost in every tech field we can see that Machine Learning is dominating and which is why we wanted to make our research ML orientated. One other reason for choosing ML is that our basic approach to our project. As we wanted to make a technique which will be independent of any particular scene, an algorithm approach would not work out for us. It is because an algorithm approach would not be intelligent as well as it will be inefficient for us. We wanted to develop such a model or machine which will work in real-time and be efficient in sense of processing power. We would feed inputs to it and it would give us the output in real-time and without ML this cannot be achieved. This is the only way to make a machine intelligent enough to understand and recognize different scenes and act upon them differently. To that end, we used convolutional neural network.

3.3 Image Quality Comparison Metric

Even if we improve an image with any anti-aliasing technique whether its our own or the one that already exists in the industry, we cannot simply look closely to the input image and the output image and say that we have improved the image quality. There should be enough mathematical data and proof to say with confidence that the image has been improved by the model. This is why there exists some algorithms which compare two different i.e. input and output and give us the result which states

whether the image has been improved or not. Such algorithms are SSIM, PSNR and MSE.

3.3.1 Structural Similarity

Image quality assessment plays an important role in computer graphics. Image quality evaluation methods are of two types: Subjective evaluation and objective evaluation. The most correct method of image quality evaluation is the subjective evaluation method. HVS is a subjective evaluation method. To mark the distorted image the subjective evaluation methods, need to organize the observers. Researches have tried to develop a mathematical model to simulate HVS characteristics and Wang proposed a model called Structural Similarity (SSIM) [25]. 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 equation (3.1):

$$SSIM(x, y) = [l(x, y)]^{\alpha}.[c(x, y)]^{\beta}.[s(x, y)]^{\gamma} \tag{3.1}$$

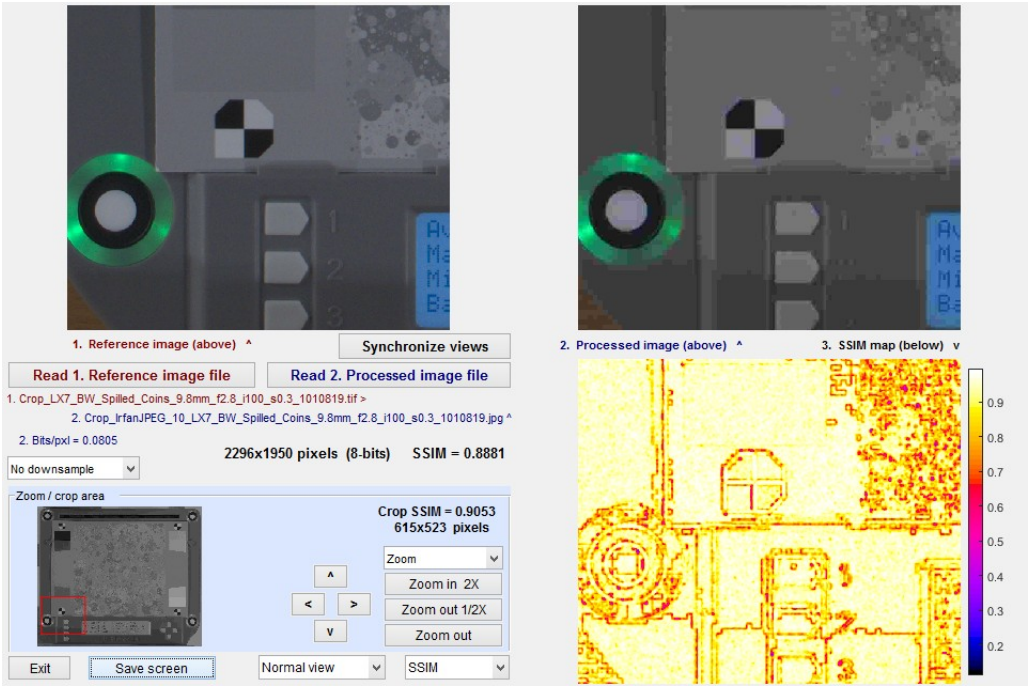


Figure 3.2: An example of SSIM Comparison

From the Figure 3.2 [26], the comparison of two images by the SSIM algorithm. The processed image file is shown on the upper-right. The file name is shown in dark blue below the images, towards the center. It was saved with an Irfanview quality level of 20 (of 100) relatively low quality; highly compressed. The reference image file is shown on the upper-right. The file name is shown in brown below the images on the left. Note that SSIM is a symmetrical measurement: results are identical if the processed and reference file images are interchanged. The SSIM index map (SSIM as a function of location) is shown on the lower-right. Colorbar (Options II) has been selected in the View dropdown menu. The other options are No colorbar (grayscale image) and Grayscale colorbar.

The overall image quality is evaluated as mean SSIM(MSSIM), which is defined as the equation (3.2):

$$MSSIM(x, y) = \frac{1}{M} \sum_{j=1}^{M} SSIM(x_j, y_j) \tag{3.2}$$

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

SSIM is a paradigm based on perception, it views the loss of the picture as a perceived shift in structural details. Also, it considers perceptions as contrast masking and luminance masking. Structural information tells that pixels are relatively close by having strong interdependencies. In a visual scene, these dependencies carry vital information about the objects. Luminance masking is a process in which light irregularities in bright areas appear to be less apparent. Contrast masking is a process where distortions are less noticeable in textures of the image. Comparison of SSIM results on a scale of 1 to -1. While comparing two images, SSIM score 1 means the images are purely similar and less than 1 or down to -1 means there is a significant difference between the images. However, even in comparing a different image the result of the SSIM score is often a positive number but not 1.

3.3.2 Peak Signal-to-Noise Ratio and Mean Square Error

In analog systems, Peak Signal-to-Noise Ratio (PSNR) has been used traditionally for a while as a consistent quality metric. It is used to measure the ratio between the highest possible signal power and the power of the distorting noise affecting the quality of its representation. It is also used to calculate the quality of reconstruction of lossy image compression codecs. The ratio is calculated in decibel form between two images. As the signals have a very wide dynamic range, the PSNR is calculated usually as the logarithm term of decibel. This dynamic range ranges from the biggest to the smallest possible values and can be adjusted by their consistency [28]. The PSNR value, for 8-bit data representation, lies in between 30 to 50 dB and for 16-bit data representation the value differs from 60 to 80 dB. Approximately 20-25 dB is the accepted range of quality loss in wireless transmission [29].

The most popular and widely used image quality measurement metric estimator is Mean Square Error (MSE). The values of MSE closer to zero are the better and it is a full reference metric. The variance of the estimator and the bias are both incorporated with mean squared error. In case of unbiased error, the MSE is the variance of the estimator. As the square of the quantity being measured like a variance, it has the same units of measurement. For a reference image f and a test image g, both of size MN, the Mean Squared Error (MSE) between the two images would be as in equation (3.3):

$$MSE(f, g) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (f_{ij} - g_{ij})^2 \tag{3.3}$$

The Peak Signal-to-Noise Ratio (PSNR) would be:

$$PSNR(f,g) = 10\log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \tag{3.4}$$

In the equation (3.4), the MSE value tends to zero and along with that the PSNR value approaches infinity which eventually proves that a higher PSNR value provides a higher image quality. At the end of the scale, a small value of the PSNR implies high numerical differences between images [30].

(a) Original uncompressed image

(b) Q=90, PSNR 45.53dB

(c) Q=30, PSNR 36.81dB

(d) Q=10, PSNR 31.45dB

Figure 3.3: Example of Luma PSNR values for a cjpeg compressed image at various quality levels.

From Figure 3.3 [31], we can see the calculation PSNR algorithm. The first image is the raw image. The quality index of the raw image in this calculation is counted as 100. However, there are other three images which are generated from the raw image. The quality of the generated image are 90, 30 and 10 respectively. We can the difference very rarely with our open eyes. As so, we need to compare the generated images with the raw image. We can see from the PSNR value that as the

quality drops of the images so as the PSNR values. The image which has a quality index of 90 has a PSNR value of 45.53 dB. From other two generated images of quality index 30 and 10 have PSNR values of 36.81 dB and 31.45 dB respectively.

Chapter 4

Dataset Analysis

As we had planned to work with Machine Learning, we needed a very large dataset. To be specific, we are going to work with Convolutional Neural Network (CNN or ConvNet) which is a class of Deep Learning. What our primary end goal is to train the CNN model with enough data meaning images, so that when we put an unknown image with aliasing to the model as an input, it can work its magic and turn the aliased photo to an anti-aliased photo.

4.1 Generating Images

To take better control over our data and our CNN model, we fabricated our own dataset using Unity video game engine. We produced 10,000 images for our project. We followed the most used and conventional method for dataset when using with Machine Learning which is to take 70% of the data for training of the model and to keep 30% of the data for future testing. Among those 10,000 images, we took 7,000 of the images to train our model and we reserved 3,000 images as test data.

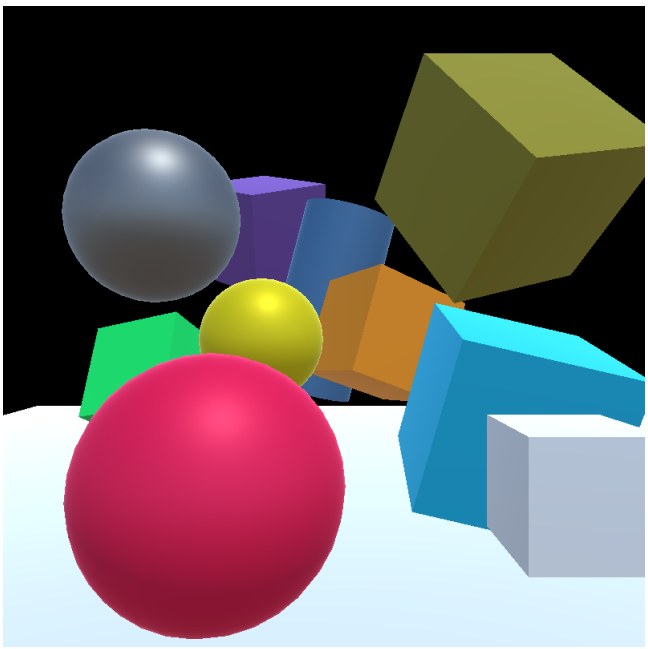


Figure 4.1: Created scene in Unity Game Engine

Using Unity scene builder, we built a scene with different shapes of objects such as triangle, square, cylinder and circle. Figure 4.1 demonstrates the scene that we created in the Unity scene builder. The output images from the scene are going to be the primary source of our data.

We took the output frames in two resolutions. One being 800x800 pixels and another being 3000x3000 pixels. There is a very good underlying core reason for doing this. The images with the resolution 800x800 pixels are our raw images and aliased. We will use these images as input to our convolutional neural network. Figure 4.1 is an example of those 800x800 pixels images. We would want our neural network to convert these input images which are aliased to make them anti-aliased in the final output. To make our neural network understand what we want to achieve we converted the images of 3000x3000 pixels resolution to 800x800 pixels and fed them to our neural network so it can understand what we want to achieve as the output it would provide. Figure 4.2 is such an example.

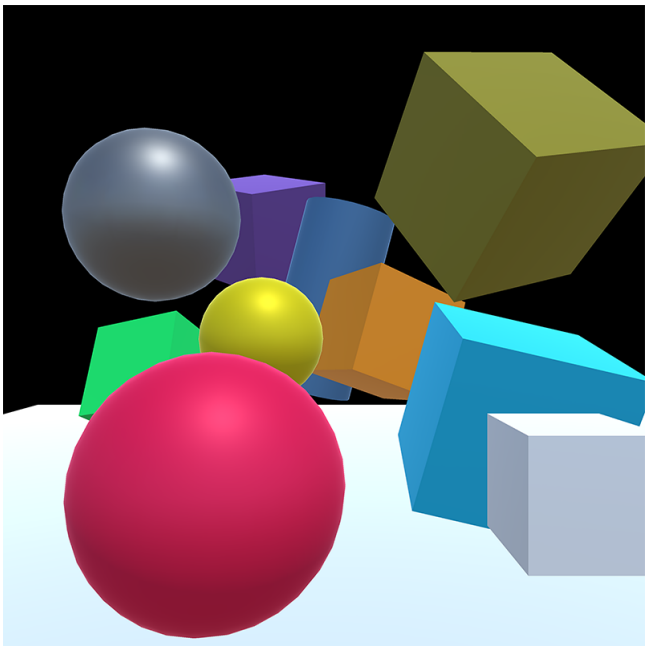
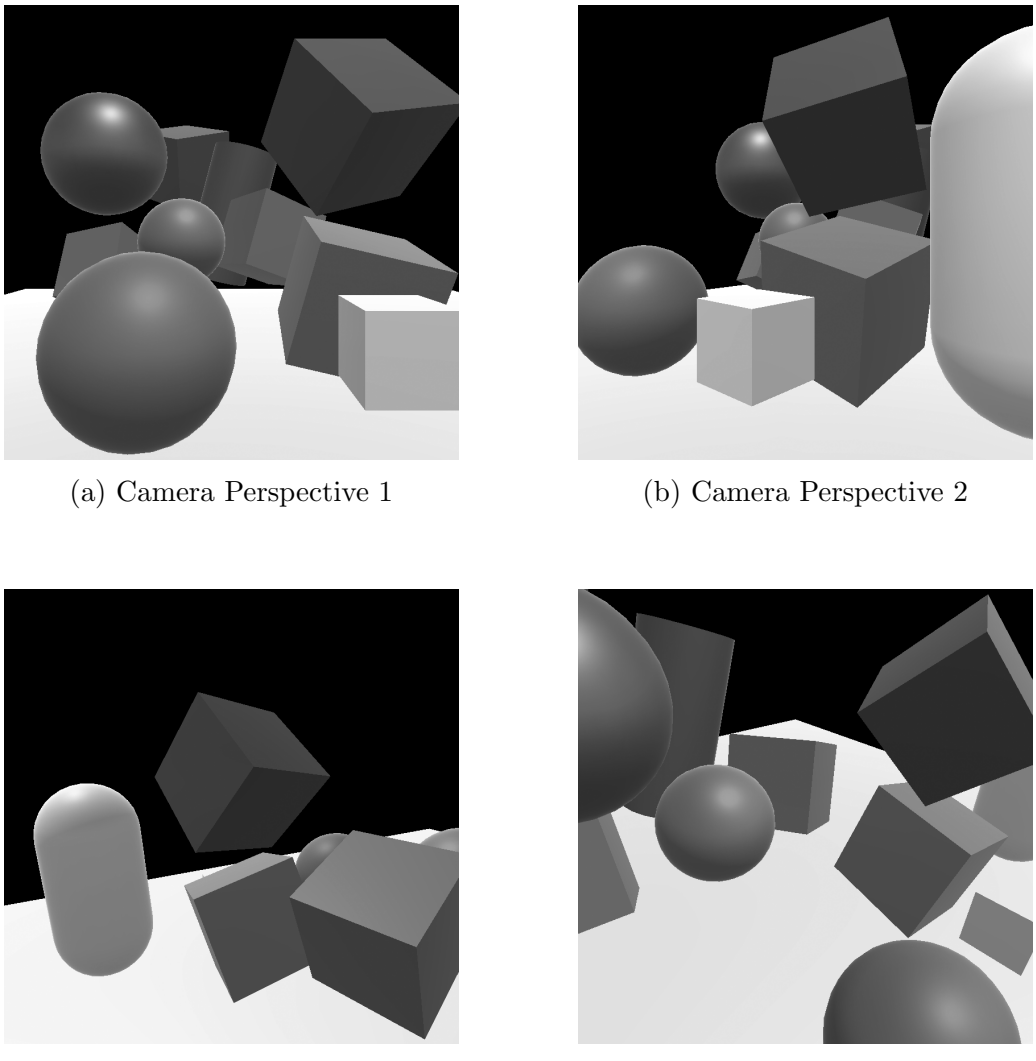


Figure 4.2: An image of 800x800 pixels converted from 3000x3000 pixels

We can see from Figure 4.2 that it is a less jagged photo than the photo in Figure 4.1. This what we want to achieve by our model. We will feed our model the anti-aliased images like the one in Figure 4.2 in different camera perspective. By that, it will understand that we want to make our images anti-aliased and not match them pixel by pixel and when we give an aliased image as an input, it would turn it to anti-aliased like the one in Figure 4.2. In short, the converted 800x800 pixels images are our ground truth and this is the level of anti-aliasing that we want to achieve given any aliased input image.

4.2 Working with the Images

After creating the scene, using a built-in tool of Unity which is called Unity Recorder, we made an animation. To generate both our input and output images, we took frame by frame shot of that animation from different camera perspective. We had to make sure very carefully that each and every shot differ from each other by a substantial number of pixels. It is because we want our neural network to understand each and every frame differently. If the difference between two frames are two or three pixels, then our neural network would be confused. It will not understand whether to anti-alias the images or to match them pixel by pixel. We took the output frames in uncompressed format which is because in compressed format the underlying data are lost from the images and our neural network model will not be able to work with those images.



(a) Camera Perspective 1

(b) Camera Perspective 2

(c) Camera Perspective 3

(d) Camera Perspective 4

Figure 4.3: Different Camera Angles of a Scene

Before feeding our model the input images of 800x800 pixels, we converted the RGB images to Black and White (BW) as shown in Figure 4.3. It is because using RGB is a very complex and highly expensive process with an extra matrix to work with.

However, working with Grayscale gives us an extra edge of advantage to come up with the model. The processing power drops drastically when shifted to BW from RGB. For example, let us think about an 800x800 pixels image. While working with this image in RGB model, we have to process with 800x800x3. This third dimension is there because of RGB model. The processing cost also boosts up exponentially because of this third dimension. However, when to try to work with BW images, the processing cost is inexpensive and it goes linearly rather than going upwards exponentially.

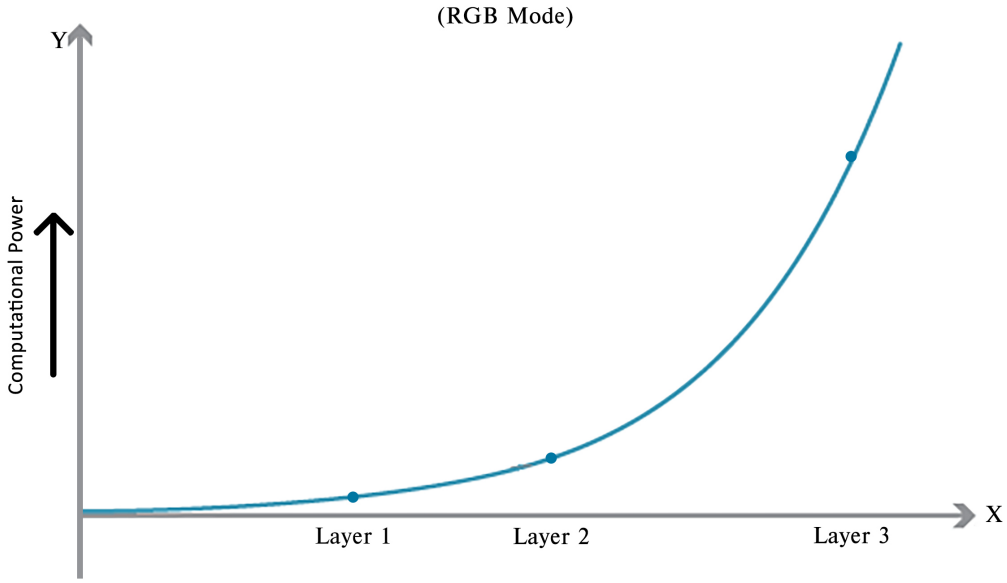


Figure 4.4: Increase of Computational Power for RGB

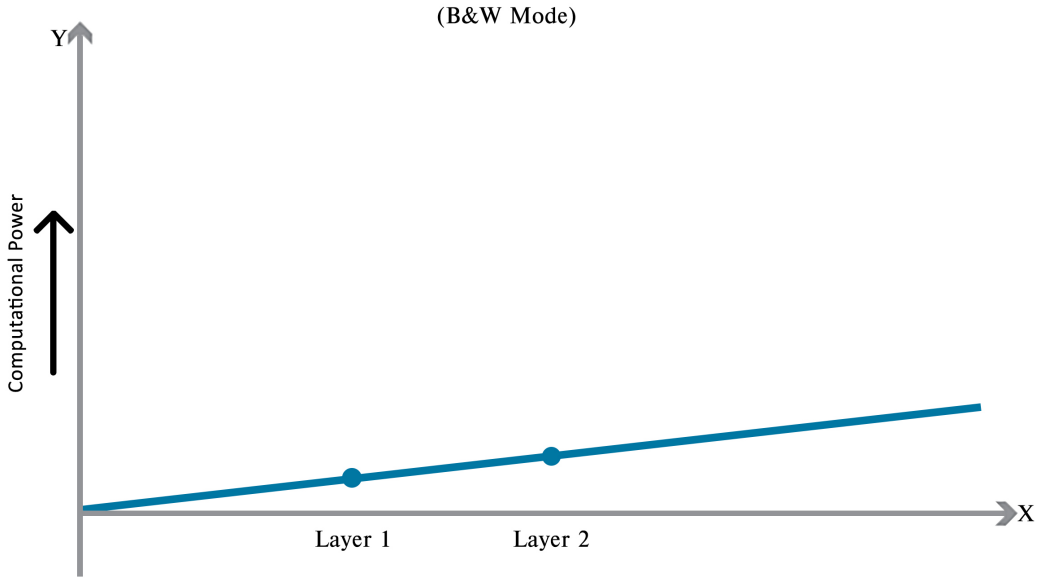


Figure 4.5: Increase of Computational Power for BW

The graphs in Figure 4.4 and Figure 4.5 expresses perfectly what we faced while working with RGB and BW. The computational cost or the processing cost grows

exponentially while working with RGB color scale. However, it grows linearly while we worked with BW or the grayscale color mode.

We converted the RGB images to Grayscale mode using the Luminosity method. It averages the values of RGB. However, it forms a weighted average to account for human perception because we're more sensitive to green than other colors, so green is weighted most heavily. The formula for Luminosity is as defined in equation (4.1):

$$0.21R + 0.72G + 0.07B \tag{4.1}$$

One more thing that we had to do before feeding the input images to our neural network is that we had to normalize the input images. For grayscale images, the brightness of the pixel value is a single number. The byte image is the most common pixel format. In the byte image that single number is stored as an 8-bit integer which gives us a range of possible values from 0 to 255 [32]. 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 ranges from 0 to 1. For this reason, we normalized our images to values of 0 to 1.

4.3 Tabular Description of the Dataset

SL.	**COLUMN_NAME**	**DESCRIPTION**
1	RGB Images	These are images that we had generated using Unity scene builder and Unity Recorder.
2	BW Images	To lower the processing cost and make the process inexpensive, we converted the RGB images to BW images and to feed them to out model. To train our neural network, we had to divide the BW images to two parts.
3	Train Images	This is one of the two parts of BW images. This dataset holds about 7000 BW images to be trained to neural network.
4	Test Images	This dataset is another part of the two parts of BW images. This one holds about 3000 images. This test dataset is set to test our model.

Table 4.1: Description of Dataset

Chapter 5

Methodology

This chapter describes how we have used our dataset of images, an open-source library called TensorFlow, an open-source neural network library Keras and Convolutional Neural Network to train our model that can apply anti-aliasing in real-time.

5.1 Technologies

The core technologies used to train and test our model are TensorFlow, Keras and, CNN. TensorFlow is the popular choice to train any machine-learning model. Also, Keras is hugely popular because it's a high language level API. Also, to work with images in the machine learning field CNN is commonly used.

5.1.1 TensorFlow

TensorFlow is an artificial intelligence library that is written in Python, C++ and CUDA programming language which is also to mention the most popular open-source library. TensorFlow can work in large scale settings and uses data flow diagrams to train models. It maps the data-flow graph nodes across several computer devices, using multicore CPUs and generally-designed ASICs known as Tensor Processing Units (TPUs) across several machines in one cluster [33]. For Google internal usage, TensorFlow was developed by the Google Brain team. On November 9, 2015, it was released under the Apache License 2.0. TensorFlow is vastly used on recognition of voice and sound, recognition of image and its pattern, text-based applications such as sentiment analysis, generating statistics with big data, self-driving cars. Deep learning is part of a broader family of machine learning approaches that focus on artificial neural networks and representation learning. Training can be supervised, semi-supervised, or unsupervised.
In this paper, to train the model in every situation, a base image and the desired output image is being provided therefore it is a supervised learning method. The model TensorFlow represents individual mathematical operators as data flow map nodes (such as matrix multiplication, convolution, etc.) [33]. This way, we can write new layers using an interface. With adding and removing new layers, we find the efficient approach that will train the model with lesser processing power overhead. With the python package manager pip, TensorFlow was installed. From TensorFlow's GitHub repository TensorFlow/models section, we can find a large number

of models trained by open-source contributors that helped us find the right values to tune the settings which guided us to train the model efficiently.

5.1.2 Keras

Keras is a high-level Python-based neural network API that can run on top of TensorFlow. François Chollet, a Google engineer, developed and maintained support for Keras. In this thesis, we used Keras Python library that provides a clean and convenient way to create a range of deep learning models on top of TensorFlow. Keras was used because it follows the best programming practices that allowed us to understand the use of Keras and implements it. It was designed keeping in mind of user-friendliness to work with Deep Neural Network. Also, for our TensorFlow workflow, Keras API combines smoothly. From 2007 Google's TensorFlow team started support of Keras in TensorFlow's core library. One of the great advantages of working with Keras is it is not limited to use CPU power only but also uses Nvidia GPUs together with the CPU to provide maximum throughput. To train our model we have used Nvidia's GTX 1080 graphics card. Which is a flagship model, with the GPU's 2560 CUDA cores, it gave us a great boost in the computation speed which ultimately saved time in training with the huge dataset. With the package manager of python called pip, Keras was installed in the system. Keras is a high-level API that is the popular choice to build Artificial Neural Networks.

5.1.3 CNN

CNN short for Convolutional Neural Network is a type of artificial neural network that is specifically designed for the processing of pixel data in image recognition and processing. As discussed in the previous section we have applied deep CNN because some CNN models significantly outperformed the image classification accuracy of previous machine learning approaches [34]. Therefore, Convolutional Neural Network is a class of deep neural network that is used for machine vision or visual image processing. The CNN image classifications take an image, process it and identify it under selected categories. For example, Face, Human, Animal, and Objects. The machine takes the input image as a pixel array which relies on the resolution of the image.
The layer of convolutional networks is shown in Figure 5.1 [35]. These layers are transferred through some images of specific bounding boxes. The first point is the image kernel in a matrix of 5 x 5 x 1 [35]. Then, True Padding increases the image dimensionality for the second sheet. The pooling layer then reduces the measured element size. The last production of the picture you want is the completely connected layer [35]. So, depending on the resolution of the image, the machine will see the $h * w * d$, as Height, Width and Dimension. For a deep-learning CNN model to train and check each input image would go through a series of filter convolution layers, pooling, and Softmax FC to identify objects with probability values from 0 to 1. Now depending on the numbers and trained model's accuracy, we can set parameters and finally the model will output the type of category the image falls into. CNN is not only used to classify objects but also it can find the difference between two images and we can use that to train the model. However, the model will not know what to do with that data. Therefore, we specify the model to act upon a scenario

in this paper, the model is specified to find the difference and apply anti-aliasing on the image.

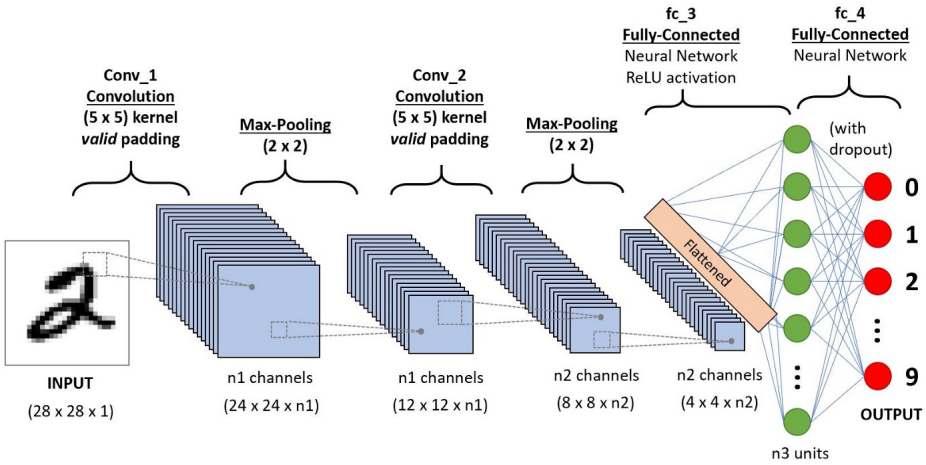


Figure 5.1: Convolutional Layers

The image data is a matrix of pixel values and in an array of pixels set to 1 for the shade of black and 0 for the shade of white, it looks like the image below in Figure 5.2. With this CNN is used here to find the aliasing effects by comparing an image with aliasing effect to an high super-sampled image [35].

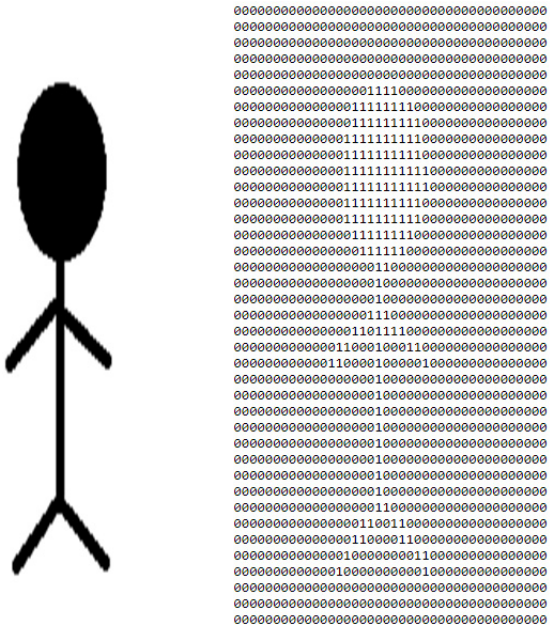


Figure 5.2: Visualization of an image to an array of pixel data

5.2 Workflow

To work, CNN requires data in the form of numbers. Such numbers are pixel values for images. These values reflect a spectrum of grayness from 0 (black) to 255 (white) when we have a grayscale picture. Also, we have mentioned in our data collection

section that these 0 to 255 values are represented as 0 or 1 to feed the image data to our model. To train our model we have used 8 layers as shown in Figure 5.3.

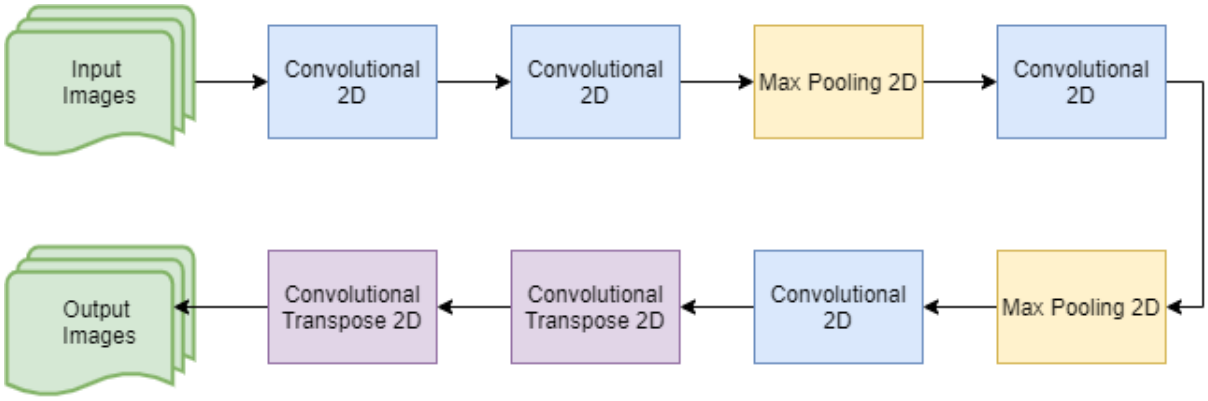


Figure 5.3: Model Architecture the CNN layers used to train the model

5.2.1 Layer 1

The first layer we feed our 800x800 pixels image is Keras Conv2D, which is a 2D Convolution Layer. This layer generates a kernel of convolution which will be translated to a tensor of outputs that use the layer output. Also, it generates a convolution kernel that is a gale of input layers that helps to generate an output tensor. The convolution kernel is a matrix which takes 0 or 1 as values in each cell depending on the white or black shade of the image. There are several options while using the convolutional layers. As we are working with 2D images we had to use the conv2D method.

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1),
  padding='valid', data_format=None, dilation_rate=(1, 1),
  activation=None, use_bias=True, kernel_initializer='glorot_uniform',
  bias_initializer='zeros', kernel_regularizer=None,
  bias_regularizer=None, activity_regularizer=None,
  kernel_constraint=None, bias_constraint=None)
```

Figure 5.4: Keras's Conv2D interface

The parameters we tuned are, the filters parameter takes an integer value. This identifies the output filters that will be generated. 32 filter value is passed in the parameter in our model. The kernel size parameter also takes an integer or it can take a tuple value which determines the height and width of the convolution 2D border. As our dataset's image size is 800x800 pixels, therefore, we passed 800 in this parameter. Padding takes a string value and we kept it as same. In the activation parameter, the seed was changed to 1919. Random seed takes and analyzes random images but we wanted to train the model with a specific seed so that it can help in future diagnostics. The rest of the parameters are not tuned during the training of the model so it was kept default just as shown in Figure 5.4. 800x800 pixels image generated 32 feature maps. Feature maps are the outputs of the Conv2D

layer. In this context, 32 different 800800 pixels image data are created tuning 832 configurable values we label them as "Param". Converting image data to the matrix of pixels to 32 different feature maps costs quite a processing power. First, we set the settings to generate 128 and 64 feature maps to have a better accuracy but our training machine of core i5 6600K processor overclocked at 4.1 GHz and 16 GB DDR4 RAM at 3200 MHz bus speed, crashed several times due to more memory demand. However, changing the parameter to get 32 feature maps it had less impact on our machine and able to train the model at a decent speed.

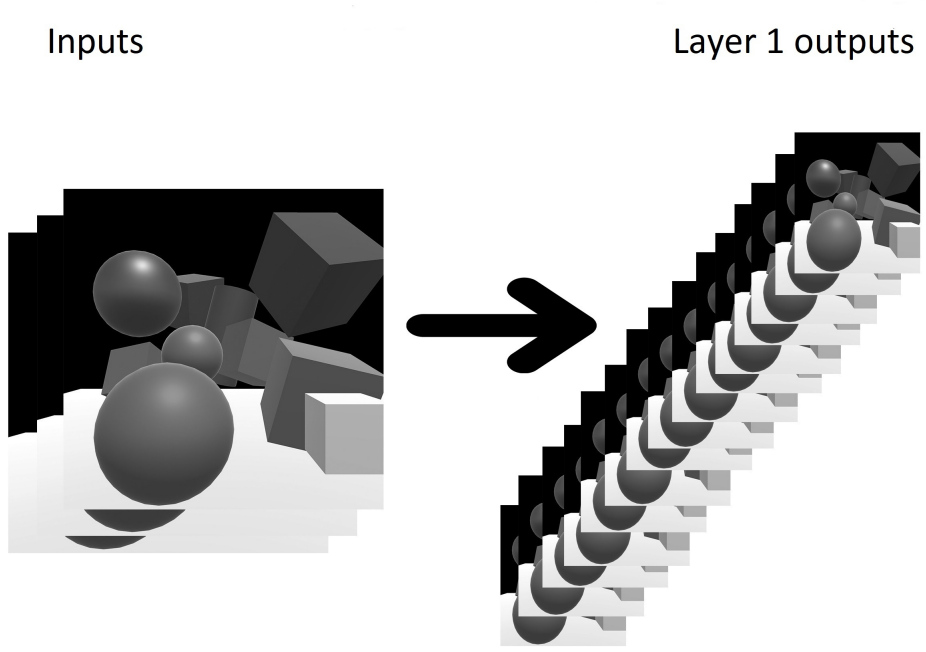


Figure 5.5: Visual representation of the outputs of Layer 1

Visual representation of the outputs of our first layer is shown in Figure 5.5. 3 images of 800x800 pixels are given as input to the Conv2D layer and it generates 32 feature maps of 800x800 pixels.

5.2.2 Layer 2

Again, we feed layer 1's data to another Conv2D layer. Where again 32 different 800800 pixels image data are being created from Layer 1's every feature maps. These feature maps are being generated by tuning 9248 configurable values. Also, 1024 samples are generated in layer 2. We use the Conv2D layer twice to let the machine have a great amount of differently tuned samples because this model will be able to figure out different techniques to create the desired output and ultimately teach itself to be efficient. In various image processing with machine learning techniques, multiple application of conv2D layer is seen.
Figure 5.6 demonstrates visually how the output of layer 2 samples are generated in the training process of our model.

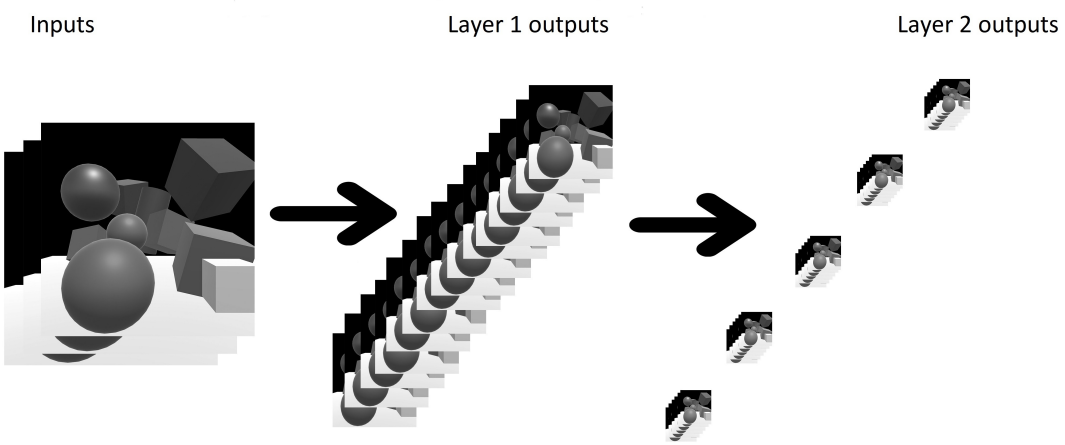


Figure 5.6: Visual representation of the outputs of Layer 2

5.2.3 Layer 3

A problem with the maps of the output function is that they are sensitive to the position of the input features. To overcome this problem, we have used the Max Pooling 2D layer. With convolutional layers, downsampling can be accomplished by changing the stage of the convolution in the image data. The use of a pooling layer is a more stable and growing solution. Pooling layers provide a sampling feature map solution by summarizing the location of features in the feature map patches. A 4x4 matrix of data is sent to a 2x2 Max Pool filter will generate a 2x2 matrix. In a 4x4 matrix has 4 segments of 2x2 values. Form each segment the maximum number is taken and set to the corresponding 2x2 matrix. This is how every 4 segments get processed.

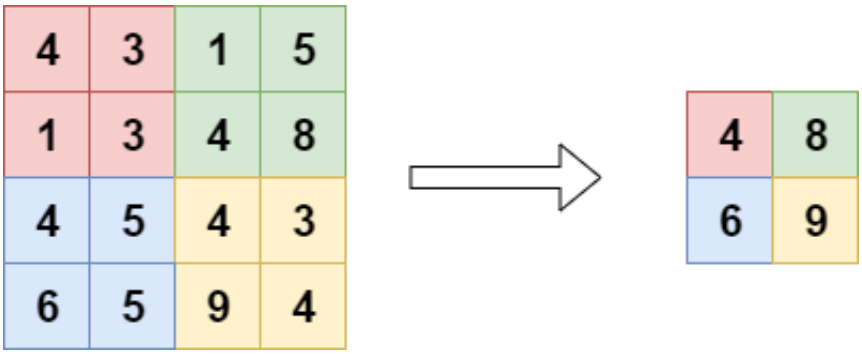


Figure 5.7: Max Pooling over a 4x4 matrix

Figure 5.7 [36] shows the visual overview of Max Pooling and how it generates a smaller matrix. Moreover, from the output of layer 2 we are getting 32 images and in the layer 3 max-pooling is done over each image of layer 2 and we are getting 32 image outputs. As we are providing 800x800 pixels images to the Max Pooling 2D layer therefore, it is converted to 400400 pixels values.
Visual representation of layer 3 outputs is shown in Figure 5.8 as shrunk due to the conversion of 800x800 pixels images to 400x400 pixels images. Also, the layer 3 output count are shown same as layer 3.

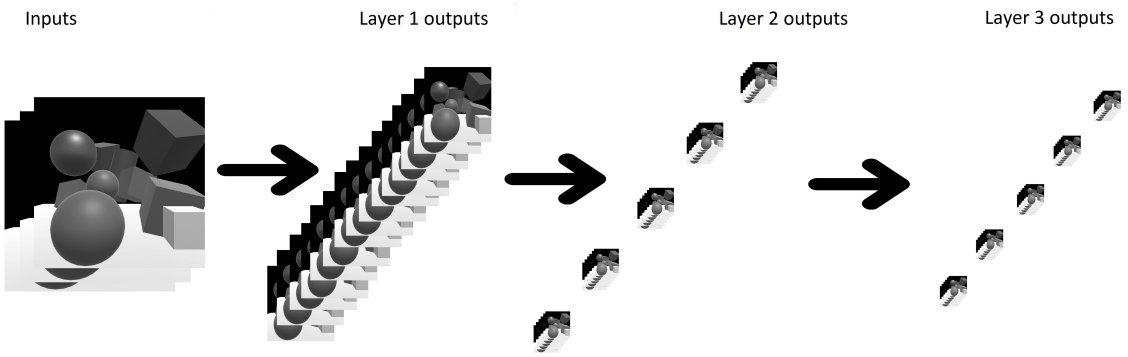


Figure 5.8: Visual representation of Layer 3 outputs

5.2.4 Layer 4

Again, we feed this layer 3's data to another Conv2D layer. This time we are generating 64 feature maps from 400x400 pixels data. Possible tweaking parameters are 2112 now which can be tuned to have more control over the training process.

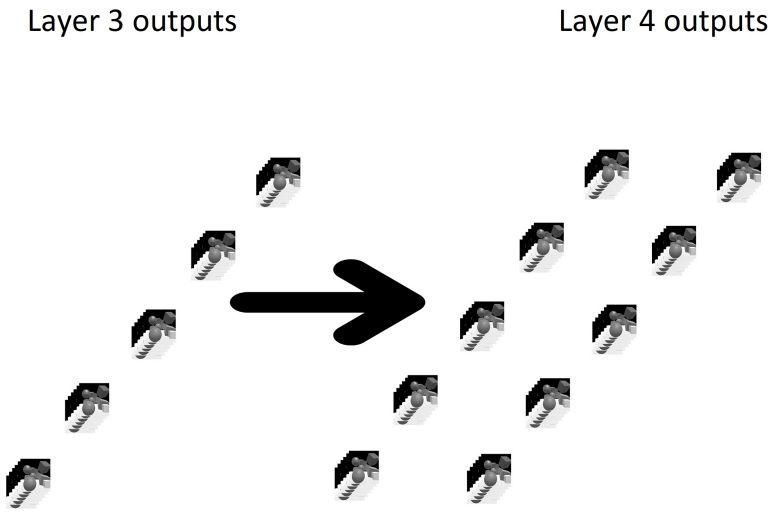


Figure 5.9: Comparison of Layer 4 outputs

Figure 5.9 shows how the dataset gets double in arrays of pixel data. Moreover, in the training process, deep CNN uses different algorithms to figure out the pattern on how to apply anti-aliasing to the base image.

5.2.5 Layer 5

Layer 4's operations increasing the data to double causing the training process to affect the throughput therefore, another Max Pooling 2D layer is added here which we are calling layer 5. This layer's main job is to tackle layer 4's overhead, reducing

the image data from 400x400 pixels data to 200200 pixels values. By doing this we have reduced memory consumption massively.

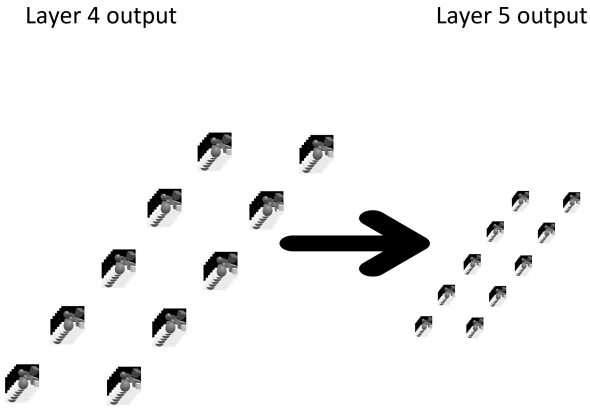


Figure 5.10: Comparison of Layer 4 and Layer 5 outputs

Visual representation of layer 5's output in Figure 5.10 as reduced resolution images.

5.2.6 Layer 6

We repeat the process by feeding layer 5's data to another Conv2D layer. By doing this the model can create many different feature maps, compare each output with the loss function which is 1-SSIM value. After feeding this loss function to the machine, our model is able learn to do the anti-Aliasing efficiently. This layer generates 200200 pixels image data into 128 feature maps.

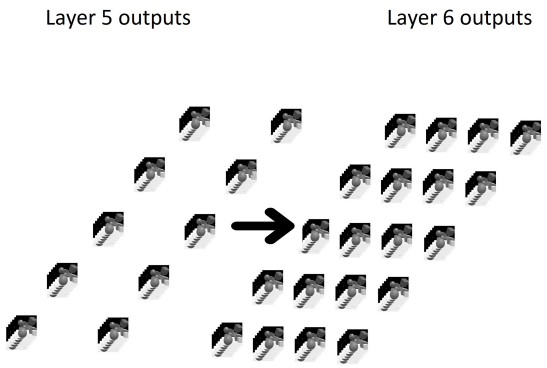


Figure 5.11: Comparison of Layer 5 and Layer 6 outputs

In figure 5.11 we can see the dataset gets doubled again so that CNN can apply its algorithms to apply anti-aliasing on images. Also, giving 73856 possibilities of parameter tuning.

5.2.7 Layer 7

As the Max Pooling 2D layers reduce the image's size but we want to get an anti-aliased image with the same resolution as the given input image so we need to upsample the image now. Upsampling can be done in two ways, using the UpSampling2D layer or the Conv2Dtranspose layer. However the UpSampling2D only upsamples the data therefore, quality loss is expected. We have to upsample without sacrificing the image quality. During the training process, the Conv2Dtranspose can learn how to maintain quality and upsample the data. Therefore, used the Conv2Dtranspose as layer 7 to upsample the image's data. Also, we have set this layer to merge the 128 feature maps of layer 6 into 64 feature maps.

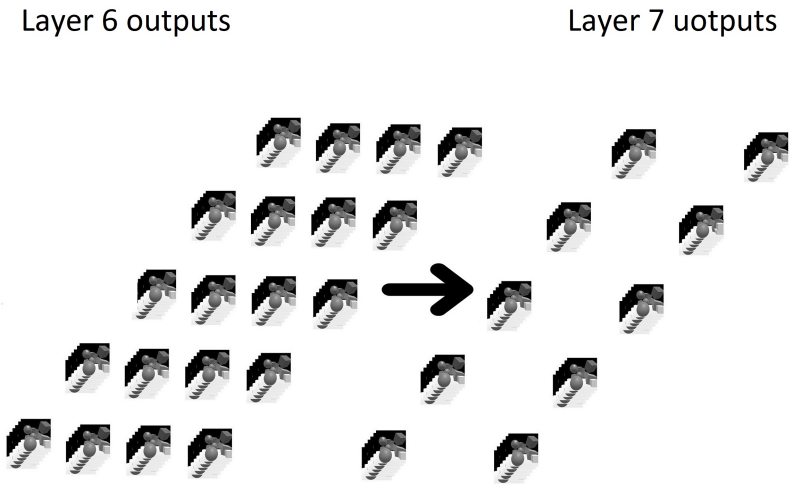


Figure 5.12: Comparison of Layer 6 and Layer 7 outputs

Figure 5.12 shows the comparison of data reduced using the Conv2Dtranspose layer. Also, this layer increases the matrix array of 200x200 to 400x400, acting as an upsampling of resolution.

5.2.8 Layer 8

Lastly, as layer 8, we use another Conv2Dtranspose layer to get upsample the 400x400 pixels data to 800800 pixels image data as given in the input image. Also, the Conv2Dtranspose layer is used to merge 64 feature maps into one image. Moreover, tuning 577 parameters different results can be achieved. By applying the Conv2Dtranspose layer an anti-aliased image was produced. This final image is our expected output. In the training process of our model tuning various parameters, we got different quality anti-aliasing. However, with the mentioned layers as described we have achieved outstanding anti-aliasing on grayscale images.
The application of anti-aliasing was done through all these layers. Figure 5.13 shows the conversion of 64 featured maps to one 800x800 resolution anti-aliased image as the final output.

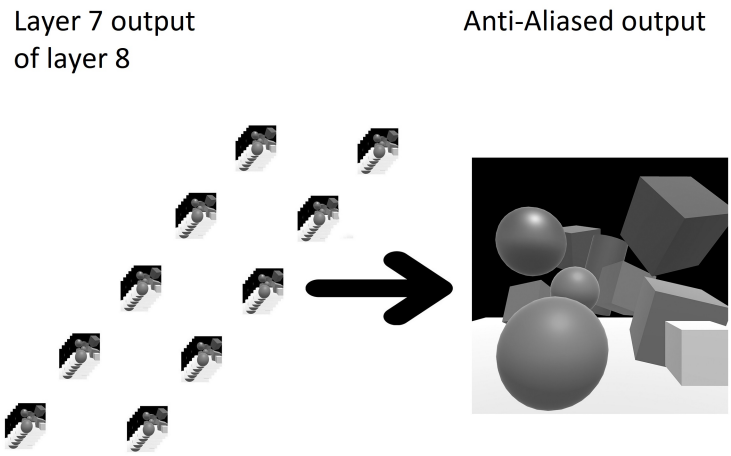


Figure 5.13: Conversion of Layer 7 data to Layer 8 image output

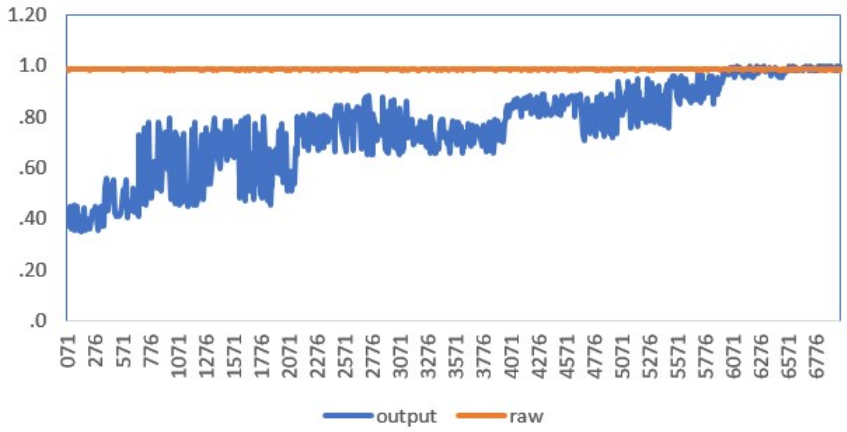


Figure 5.14: SSIM scores during training

Figure 5.14 shows the model's accuracy in terms of SSIM value during training. It took us 2 hours to train the model with 7000 images.

Chapter 6

Experimental Result Analysis

This chapter shows and compares the model's accuracy in anti-aliasing grayscale images. The image quality varies from different people's perspectives. How a computer understands image quality is a different scenario. As we have mentioned earlier that with SSIM and PSNR we are comparing the image quality. Also, the SSIM shows its output between 0 to 1 and PSNR shows it from 0 to infinity. However, in this paper, the higher the number, the better the anti-aliasing is applied. Here, we are showing the comparison of the aliased image(raw) and anti-aliased image applied by our model side by side and also with SSIM and PSNR scores with popular anti-aliasing techniques.

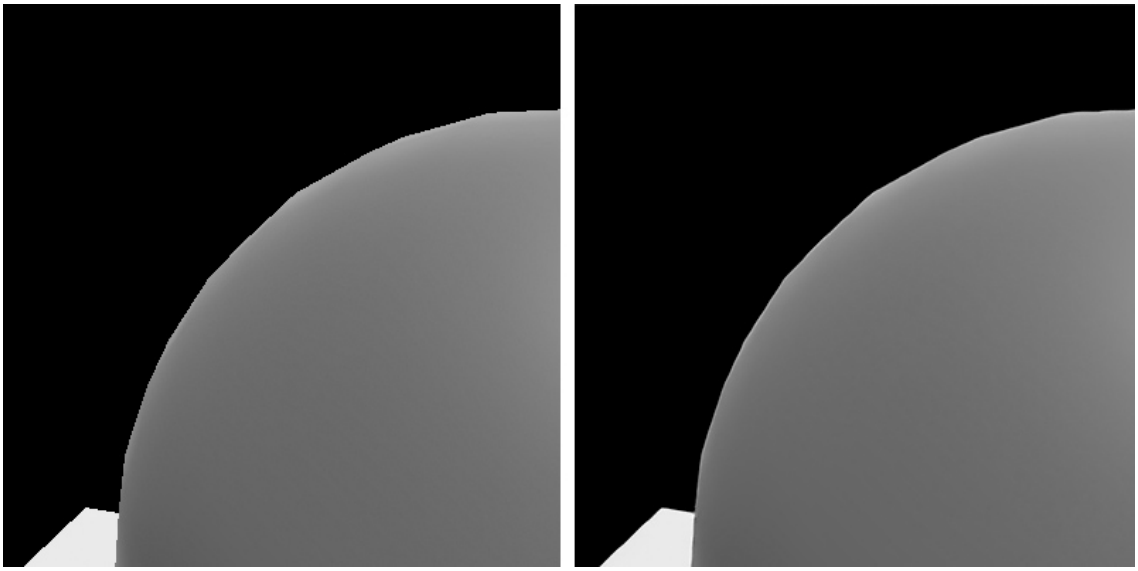


Figure 6.1: Zoomed view of raw image vs our model's anti-aliasing output

Figure 6.1's left image shows a zoomed view of a sphere from our test dataset. We can see that on the left image, the edges of the sphere have a lot of aliasing effects. Applying anti-aliasing over the edge of a sphere is tricky for most algorithms. However, our model has done a great job applying anti-aliasing and in Figure 6.1's right image we can see the output were the aliasing effects are gone. Also, how the sphere's edges are smoothed out is visible. Here, unlike other computation expensive anti-aliasing, the object is not blurred, maintaining good quality without tuning other parameters of the image.

	Raw	FXAA	DLAA	NFAA	Our model
SSIM	0.98993	0.99044	0.98814	0.98815	0.9951
PSNR	38.16266	39.46958	37.01372	37.23657	43.75568

Table 6.1: Quality comparison of anti-aliasing applied over Figure 6.1

From Table 6.1 we can say, our model can not only apply anti-aliasing that is visually improving image quality but also can achieve higher scores with SSIM and PSNR comparison.
Another example from our test dataset contains many spheres and square objects which is also a complicated scene.

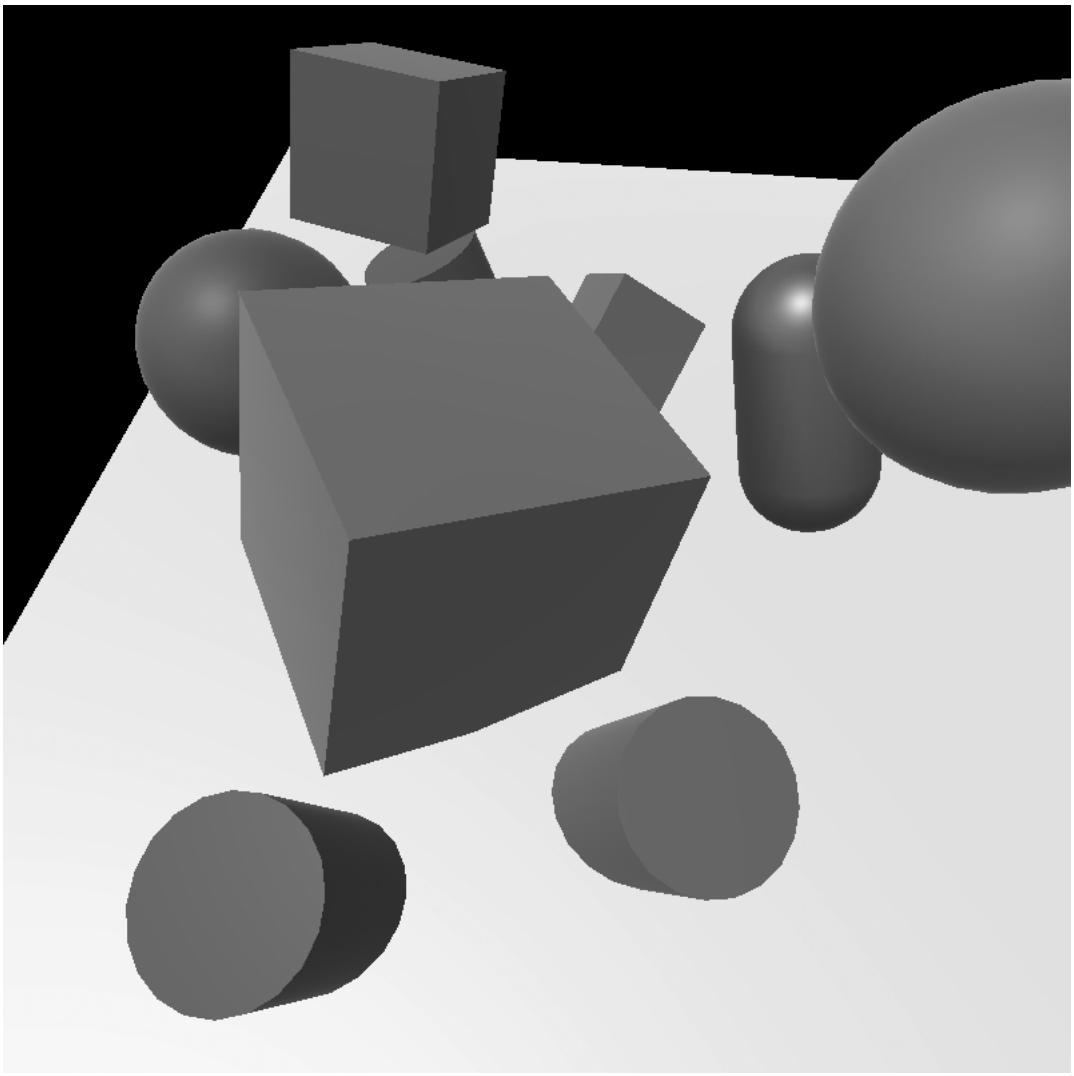


Figure 6.2: Raw image of a scene with multiple objects

Figure 6.2 is the raw image that consists of multiple objects and a complex scene. Upon providing the model the image of Figure 6.2 the models output the anti-aliased image which is shown in Figure 6.3. In Figure 6.3 we can see how the edges are taken care of with smooth anti-aliasing. Also, the edges over another object are treated as a different object and the jaggedness is reduced greatly without any visible errors, producing a quality looking image.

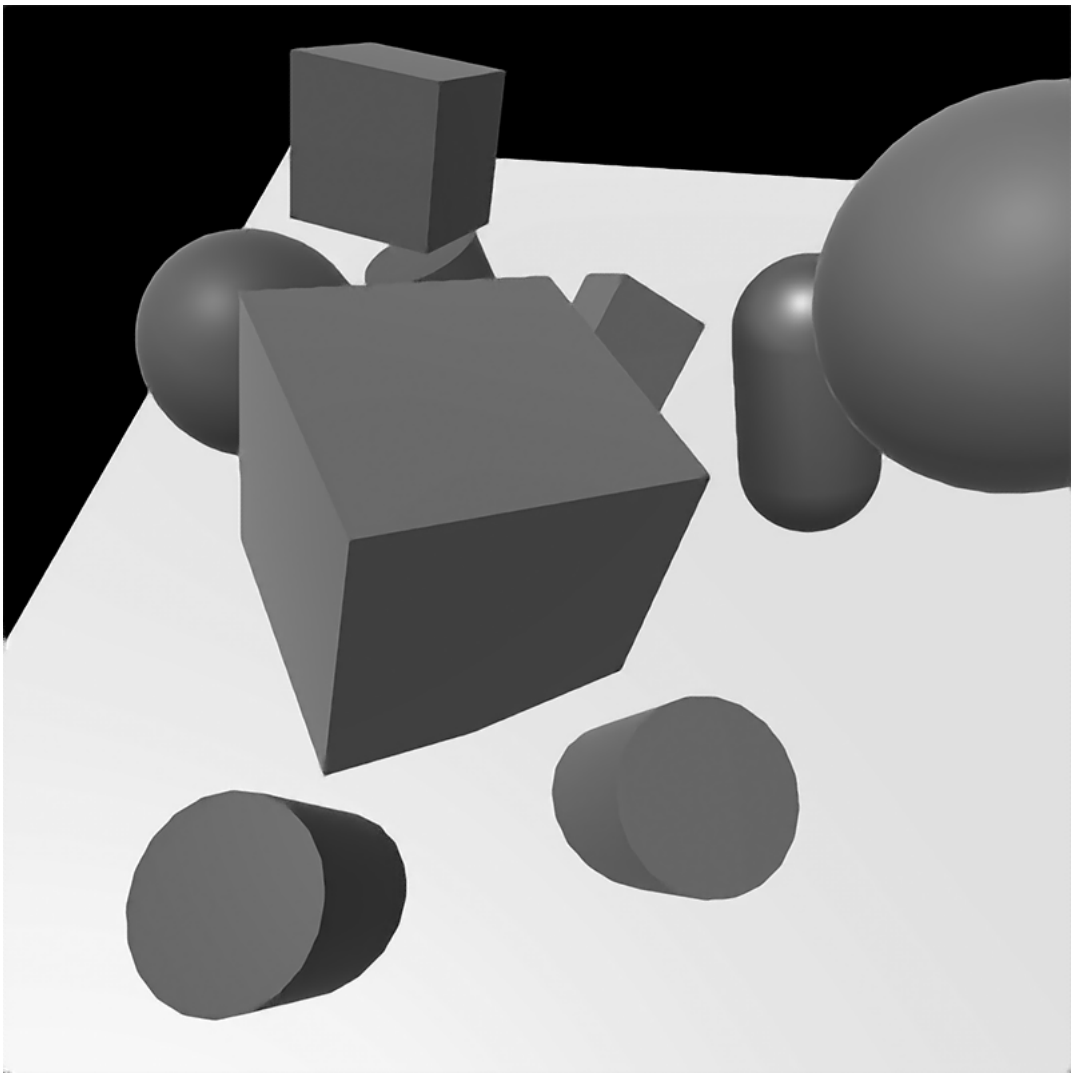


Figure 6.3: Our models anti-aliasing applied over Figure 6.2's image

	Raw	FXAA	DLAA	NFAA	Our model
SSIM	0.98448	0.98558	0.98328	0.98301	0.99350
PSNR	35.45589	36.72744	35.31345	35.04870	40.95645

Table 6.2: Quality comparison of anti-aliasing applied over Figure 6.2

Also, we can see in Table 6.2 both SSIM and PSNR scores our model scored the highest number. The SSIM score of our model is slightly over FXAA. However, in terms of PSNR score, our model scored much higher. Compared to the raw image where other algorithms score approx 1.0 point higher but our model scored over 4.0 points.

The two images are not enough to prove the model's performance. Therefore we have to test the model with larger number of dataset. As we mentioned earlier we had divided our dataset into 70-30 ratio. From the 3000 test samples we took 10 images and to test the models accuracy with multiple samples, we provided the model with those 10 images.

Table 6.3 shows the SSIM and PSNR scores of 10 test images. In all of test cases the model has achieved higher scores than FXAA, DLAA, NFAA. Also, the scores

Image	Type	Raw	FXAA	DLAA	NFAA	Our model
0000.png	SSIM	0.98782	0.98872	0.98725	0.98691	0.99477
	PSNR	37.21314	38.75829	37.18481	36.99072	43.17276
0001.png	SSIM	0.98839	0.98925	0.98759	0.98785	0.99513
	PSNR	37.79884	39.07804	37.50954	37.62629	42.78457
0002.png	SSIM	0.98971	0.99001	0.98765	0.98835	0.99579
	PSNR	38.81713	39.82603	37.81833	38.22745	44.75467
0003.png	SSIM	0.98848	0.98908	0.98741	0.98767	0.99431
	PSNR	37.54802	38.92850	37.35816	37.38727	41.90409
0004.png	SSIM	0.99078	0.99142	0.98966	0.98975	0.99481
	PSNR	40.69677	42.07886	39.15692	39.58075	44.53222
0005.png	SSIM	0.98448	0.98558	0.98328	0.98301	0.99350
	PSNR	35.45589	36.72744	35.31345	35.04870	40.95645
0006.png	SSIM	0.98423	0.98517	0.98210	0.98223	0.99019
	PSNR	35.40156	36.80594	34.76427	34.53790	37.02740
0007.png	SSIM	0.98872	0.98953	0.98755	0.98718	0.99396
	PSNR	38.25745	40.02589	37.86480	36.72762	43.31078
0008.png	SSIM	0.98993	0.99044	0.98814	0.98815	0.99510
	PSNR	38.16266	39.46958	37.01372	37.23657	43.75568
0009.png	SSIM	0.98656	0.98822	0.98576	0.98558	0.99392
	PSNR	35.84948	37.73382	35.44221	35.28798	40.34686
0010.png	SSIM	0.98898	0.99011	0.98773	0.98720	0.99434
	PSNR	36.83258	38.81912	36.28711	35.82578	43.54259

Table 6.3: Quality comparison of anti-aliasing applied over 10 test images

of the model is much higher.
Figure 6.4 and Figure 6.5 visualizes the scores of Table 6.3. It is visible that our model has outperformed FXAA, DLAA, NFAA in terms of boh SSIM and PSNR scores.

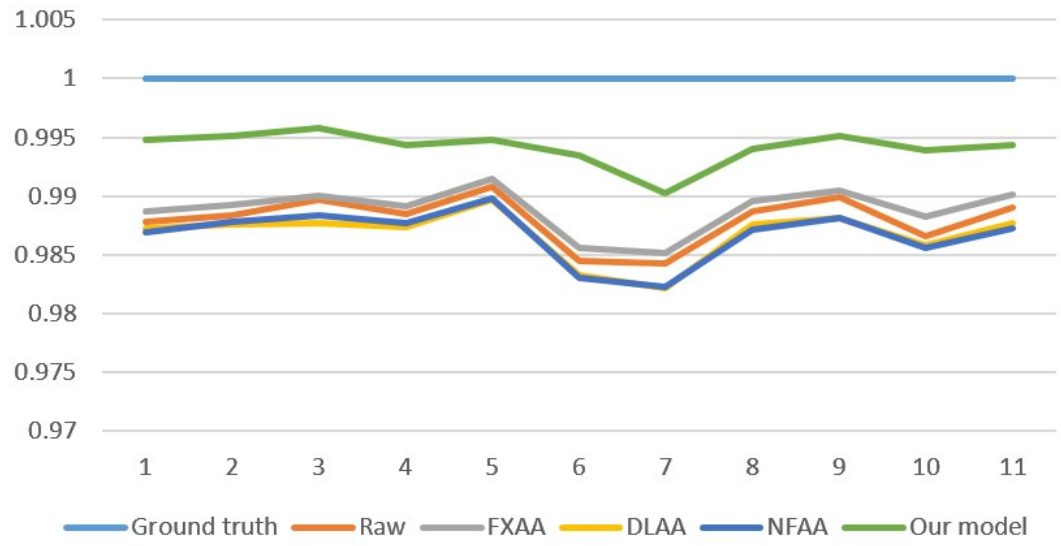


Figure 6.4: SSIM score trend of Table 6.3

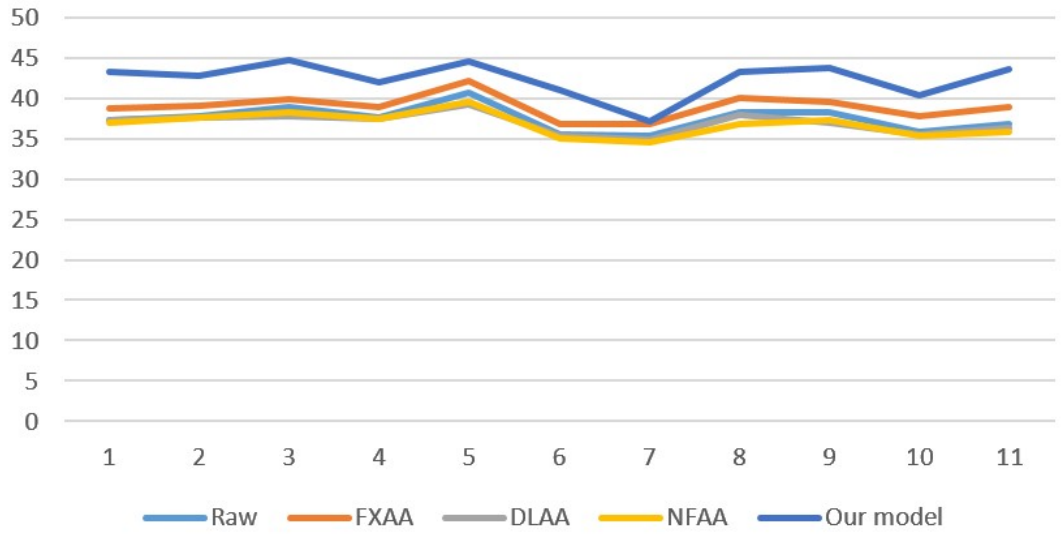


Figure 6.5: PSNR score trend of Table 6.3

After that, we wanted to evaluate our model even further, from our test dataset we provided the model with an extremely complex scene with lots of small and big edges.
Here, the model has also outperformed other algorithms. In Figure 6.4 we can see the FXAA's SSIM score is lower than our model's output.

	Raw	FXAA3	Our model	4x Super-sampled
SSIM	0.9903	0.9911	0.9916	1
PSNR	37.21314	38.75829	43.17276	∞

Table 6.4: Quality comparison of anti-aliasing applied over Figure 6.6

Also, Table 6.4 shows the score of different anti-aliasing techniques and their SSIM and PSNR scores of anti-aliasing applications over the raw image of Figure 6.6. De-

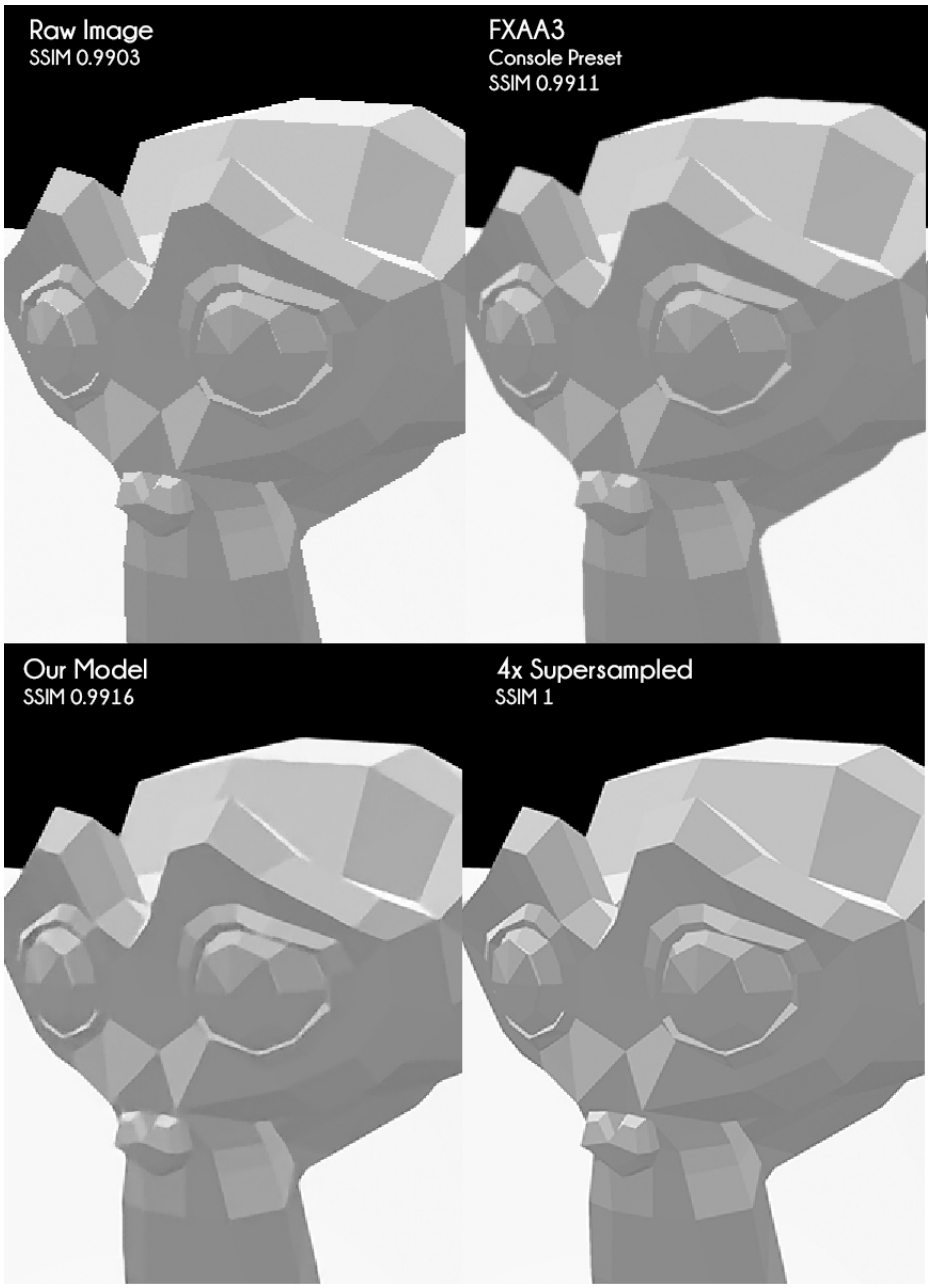


Figure 6.6: Comparison of anti-aliasing on a complex scene

spite the test image being exceptionally complex with over vast number of edges, anti-aliasing is applied very efficiently reaching high scores beyond expectations. This represents the powerful capability of anti-aliasing done through machine learning.

Image	Raw	Our model
10227.png	0.99319	0.99426
10451.png	0.99341	0.99390
10508.png	0.99366	0.99389
10724.png	0.99440	0.99417
3006.png	0.99122	0.99404
3131.png	0.99132	0.99456
3394.png	0.99317	0.99446
4122.png	0.99160	0.99344
5972.png	0.98739	0.99434
6237.png	0.98774	0.99336
7345.png	0.98857	0.99429
8244.png	0.98786	0.99392
8289.png	0.98769	0.99379
9193.png	0.98747	0.99422
9794.png	0.98840	0.99447

Table 6.5: Raw vs our model's SSIM scores of 15 test images

Table 6.5 represents the SSIM scores of raw vs our model's output over 15 test images.

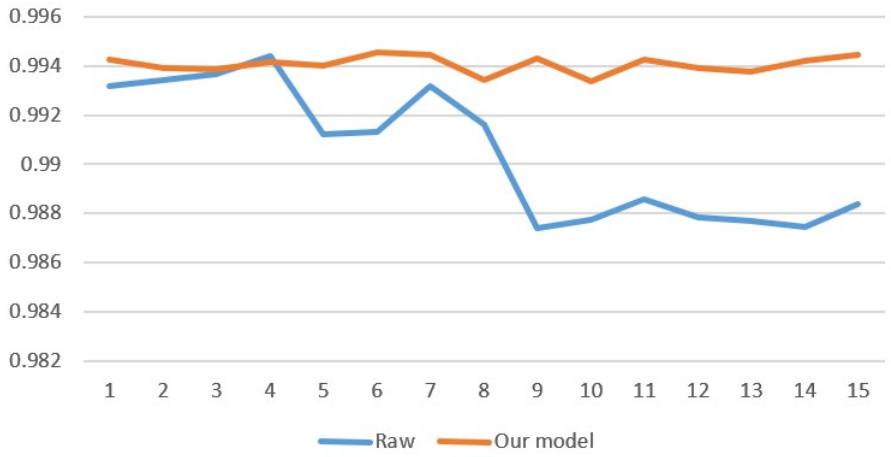


Figure 6.7: Table 6.5's data represented as chart

Our model can apply anti-aliasing accurately and SSIM and PSNR scores are consistent. Chart of Figure 6.7 proves that our model can perform anti-aliasing and provide quality enhancement over other anti-aliasing algorithms. Moreover, in 98 out of 100 images anti-aliasing is applied perfectly to enhance image quality.

Chapter 7

Conclusion and Future works

In our attempts, to make an intelligent anti-aliasing system using Deep Learning which is content aware, efficient and inexpensive enough for such a process, we have come a long way. From a dataset consisting of tens of thousands of RGB images and later on converting them to BW for an efficient processing expense, we have generated an output which is far better than some of the industry standard anti-aliasing techniques. To be specific, for a ground truth image of SSIM score 1, our model produced in image whose SSIM score is 0.99477 whereas for the same image the values of DLAA, FXAA and NFAA are 0.98725, 0.98872 and 0.98691 respectively. The difference of values is far greater when we consider the PSNR values. To state an example of such occurrences, for a ground image of PSNR valuing at infinity, the PSNR values of the same image are 36.72744, 35.31345 and 35.04870 which images are generated by FXAA, DLAA and NFAA respectively. However, for our model, PSNR gives us a value of 40.95645 which is far greater than other algorithms.

All these milestones set by our model, are only on BW images. However, we are thinking, in near future we will work on RGB images too and find out how our model can be improved in various ways.

The trained model's accuracy of anti-aliasing is groundbreaking. However, it has some limitations in real-time application scenarios. The first limitation of this research is the model is trained only using grayscale images therefore, it can apply anti-aliasing in grayscale images only. In the future, this model could be trained using RGB images and it will be able to apply anti-aliasing in RGB images too. Moreover, the dataset to train and test were generated by us. Using a larger amount of dataset of different 3D scenes or images, in the future, it can enhance accuracy. Also to mention the model can apply anti-aliasing in 29 FPS in real-time. Gamers like to play their game over 30 FPS. Some even prefer to play at 60, 144 or 240 FPS. Therefore, parameters of the layers can be tuned depending on test cases and even layer structures can be rearranged to get the model to apply anti-aliasing even faster than 30 FPS. Moreover, our model cannot apply anti-aliasing over an image with textures. It blurs out the textures and unable to maintain good quality. Therefore, it can be said that various effective research scopes are present as future work.

Bibliography

[1] K. Beets and D. Barron, "Super-sampling anti-aliasing analyzed", 2000.

[2] C. A. Oliveros Labrador, "Improved sampling for temporal anti-aliasing (a sobel improved temporal anti-aliasing)", eng, LU-CS-EX 2018-02, 2018, Student Paper, ISSN: 1650-2884.

[3] U. Catak, *Anti-aliasing*, Dec. 2017. [Online]. Available: https://www.definition.net/define/anti-aliasing.

[4] J. Jimenez, J. I. Echevarria, T. Sousa, and D. Gutierrez, "Smaa: Enhanced subpixel morphological antialiasing", *Computer Graphics Forum*, vol. 31, no. 2pt1, pp. 355–364, 2012. DOI: 10.1111/j.1467-8659.2012.03014.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03014.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03014.x.

[5] *Multisampling anti-aliasing in hdrp: High definition rp: 6.7.1-preview*. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@6.7/manual/MSAA.html.

[6] J. Jimenez, D. Gutiérrez, J. Yang, A. Reshetov, P. Demoreuille, T. Berghoff, C. Perthuis, H. Yu, M. Mcguire, T. Lottes, H. Malan, and E. Persson, "Filtering approaches for real-time anti-aliasing", *ACM SIGGRAPH 2011 Courses, SIGGRAPH'11*, Aug. 2011. DOI: 10.1145/2037636.2037642.

[7] *Multisample anti-aliasing*, Dec. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Multisample_anti-aliasing.

[8] Mjp, *A quick overview of msaa*, Aug. 2017. [Online]. Available: https://mynameismjp.wordpress.com/2012/10/24/msaa-overview/.

[9] A. Reshetov, "Reducing aliasing artifacts through resampling", in *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, ser. EGGH-HPG'12, Paris, France: Eurographics Association, 2012, pp. 77–86, ISBN: 9783905674415.

[10] J. Atwood, *Fast approximate anti-aliasing (fxaa)*, Dec. 2011. [Online]. Available: https://blog.codinghorror.com/fast-approximate-anti-aliasing-fxaa/.

[11] V. Bondarev, *Fxaa – fast approximate anti-aliasing*, Oct. 2013. [Online]. Available: http://bondarev.nl/?p=96.

[12] T. Lottes, *Fxaa*, Feb. 2009. [Online]. Available: https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf.

[13] E. A. Hutchins, *System and method for single-sample virtual coverage anti-aliasing*, US Patent 7,372,471, May 2008.

[14] C. J. Brennan, J. Nguyen, E. T. Yu, and N. Lu, "Interface adhesion between 2d materials and elastomers measured by buckle delaminations", *Advanced Materials Interfaces*, vol. 2, no. 16, p. 1 500 176, 2015.

[15] JEGX, *Amd eqaa modes (enhanced quality anti-aliasing) for radeon hd 6900 series*, Sep. 2011. [Online]. Available: https://www.geeks3d.com/20110915/amd-eqaa-modes-enhanced-quality-anti-aliasing-for-radeon-hd-6900-series/.

[16] A. Edelsten, *Nvidia dlss: Your questions, answered*, Feb. 2019. [Online]. Available: https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-your-questions-answered/.

[17] *2020 video game industry statistics, trends data - the ultimate list*, Mar. 2018. [Online]. Available: https://www.wepc.com/news/video-game-statistics/.

[18] D. W. II, *The history of augmented reality (infographic)*, Dec. 2017. [Online]. Available: https://www.huffpost.com/entry/the-history-of-augmented-_b_9955048.

[19] P. Schueffel, *The concise fintech compendium*, Oct. 2017. [Online]. Available: https://web.archive.org/web/20171024205446/http://www.heg-fr.ch/EN/School-of-Management/Communication-and-Events/events/Pages/EventViewer.aspx?Event=patrick-schuffel.aspx.

[20] M. Noghabaei, A. Heydarian, V. Balali, and K. Han, "Trend analysis on adoption of virtual and augmented reality in the architecture, engineering, and construction industry", *Data*, vol. 5, p. 26, Mar. 2020. DOI: 10.3390/data5010026.

[21] M. Noghabaei, K. Asadi, and K. Han, "Virtual manipulation in an immersive virtual environment: Simulation of virtual assembly", in *Computing in Civil Engineering 2019*, pp. 95–102. DOI: 10.1061/9780784482421.013. eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784482421.013. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/9780784482421.013.

[22] A. Charlon, *Dithering on the gpu*, Jun. 2016. [Online]. Available: http://alex-charlton.com/posts/Dithering_on_the_GPU/.

[23] *News center*, Sep. 2018. [Online]. Available: https://news.developer.nvidia.com/dlss-what-does-it-mean-for-game-developers/.

[24] J. Martindale, *Nvidia rtx dlss: What it is and why it matters*, Feb. 2020. [Online]. Available: https://www.digitaltrends.com/computing/everything-you-need-to-know-about-nvidias-rtx-dlss-technology/.

[25] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: From error visibility to structural similarity*, Apr. 2004. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/15376593/.

[26] *Ssim: Structural similarity index*. [Online]. Available: https://www.imatest.com/docs/ssim/.

[27] Chen, G. H., Yang, C. L., Xie, and S. L., *Gradient-based structural similarity for image quality assessment*, Oct. 2006.

[28] U. Sara, M. Akter, and M. S. Uddin, *Image quality assessment through fsim, ssim, mse and psnr-a comparative study - journal of computer and communications*, Mar. 2019. [Online]. Available: https://www.scirp.org/journal/paperinforcitation.aspx?paperid=90911.

[29] R. G. Deshpande, L. L. Ragha, and S. K. Sharma, "Video quality assessment through psnr estimation for different compression standards", 2018.

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

[31] *Peak signal-to-noise ratio*, Mar. 2020. [Online]. Available: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio.

[32] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, *Pixel values*, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm.

[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge", *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[35] Saha, *A comprehensive guide to convolutional neural networks*, Dec. 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[36] Chris, *What are max pooling, average pooling, global max pooling and global average pooling?*, Jan. 2020. [Online]. Available: https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/.