Detecting Aliasing Artifacts in Image Sequences Using Deep Neural Networks

Anjul Patney
NVIDIA

Aaron Lefohn
NVIDIA

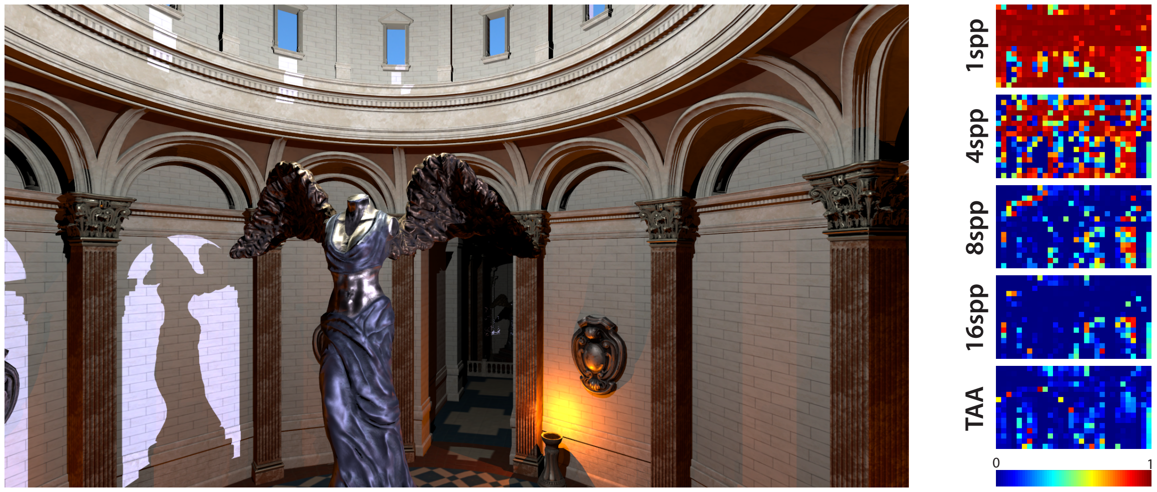


Figure 1: We use a deep classification network to detect aliasing artifacts in image sequences. In this example we show the output for one sequence of 4 frames. The network generates a scalar value as a measure of aliasing for each 64×64 tile of this sequence, resulting in the images shown on the right. Increasing sample counts result in lower overall output. Further, since we only measure aliasing, use of high-quality temporal antialiasing (TAA) correctly results in output similar to that of 16spp images. Metrics like SSIM and PSNR would classify the former as being much different from the latter.

ABSTRACT

In this short paper we present a machine learning approach to detect visual artifacts in rendered image sequences. Specifically, we train a deep neural network using example aliased and antialiased image sequences exported from a real-time renderer. The trained network learns to identify and locate aliasing artifacts in an input sequence, without comparing it against a ground truth. Thus, it is useful as a fully automated tool for evaluating image quality.

We demonstrate the effectiveness of our approach in detecting aliasing in several rendered sequences. The trained network correctly predicts aliasing in $64 \times 64 \times 4$ animated sequences with more than 90% accuracy for images it hasn't seen before. The output of our network is a single scalar between 0 and 1, which is usable as a quality metric for aliasing. It follows the same trend as (1-SSIM) for images with increasing sample counts.

CCS CONCEPTS

• **Computing methodologies** → *Antialiasing*; *Machine learning*;

KEYWORDS

aliasing, visual quality assessment, machine learning

ACM Reference Format:
Anjul Patney and Aaron Lefohn. 2018. Detecting Aliasing Artifacts in Image Sequences Using Deep Neural Networks. In *HPG '18: High-Performance Graphics, August 10–12, 2018, Vancouver, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3231578.3231580

1 INTRODUCTION

Ongoing evaluation of image quality is an important aspect of designing rendering software. While manual assessment can require a lot of time and effort, automated assessment often requires the availability of high-quality ground truth images to compare against a renderer's output. In this work, we explore the ability of deep neural networks to detect rendering artifacts in images and sequences without comparing them against a reference. Inspired by the ability of humans to do the same, we believe that an appropriately designed neural network will be able to identify patterns in image sequences that indicate undesirable visual artifacts.

Specifically, we train a deep neural network to identify aliasing artifacts in rendered image sequences, and utilize it to build an automated tool for detecting aliasing in real-time renderers. Since aliasing is a common problem in real-time rendering, our work is useful in evaluating the quality of output from an interactive graphics application, and is potentially useful in development of antialiasing techniques [Jimenez et al. 2011].

Our results show that a machine learning network is capable of distinguishing images with different degrees of aliasing. The output of our trained network is usable as a metric for image quality, producing results that correlate with traditional image metrics like Structural Similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR) when aliasing is the only artifact.

Our key contributions include:

- An approach to trains a neural network to locate artifacts in rendered image sequences without ground truth images
- An automated method for measuring and locating aliasing in rendered images, which generates a scalar metric representing the magnitude of aliasing in an image sequence

2 PREVIOUS WORK

There are several metrics usable in image quality analysis. Straightforward options include $\mathcal{L}_1$ and $\mathcal{L}_2$ norms, and Peak Signal-to-Noise Ratio (PSNR), which establish numerical differences between an image and some ground truth. Perceptually-based metrics can better identify visibility differences between images, and can thus be more useful in identifying viewer-perceivable visual artifacts. Some popular perceptual metrics include Structural Similarity (SSIM) [Wang et al. 2004] and HDR-VDP-2 [Mantiuk et al. 2011]. More recently, researchers have found that activations within trained image-classification networks (VGG19 [Simonyan and Zisserman 2014]) are also capable of computing perceptual differences between images.

However, all of these approaches compute difference between a pair of images, requiring access to a ground truth image whenever measuring image quality. Generating ground truth images can often be impractical (e.g. path tracing) and sometimes impossible (e.g. MRI scans). In contrast, this short paper proposes a practical algorithm and metric which is able to measure aliasing in image sequences without looking at antialiased ground truth.

Further, there are no standardized metrics to measure spatiotemporal aliasing in a image sequences. While temporal image-quality metrics have been studied [Aydin et al. 2010; Li et al. 2016], unlike our method these methods don't directly address aliasing artifacts, and additionally they still require ground truth reference images for comparison.

Deep neural networks are increasingly useful in computer graphics [Nalbach et al. 2017], and our work explores their ability to measure quality of rendered image sequences. We specifically study aliasing, but our approach will likely apply to other artifacts as well.

3 METHODOLOGY

3.1 Network Architecture

Figure 2 provides and overview of our network architecture. Broadly, it resembles deep image classification networks, and is inspired by VGG19 [Simonyan and Zisserman 2014]. Specifically, it has two phases: a first phase with multiple groups of convolutions followed by pooling, and a second phase with several fully-connected layers. The key modifications that we made to the VGG19 architecture are:

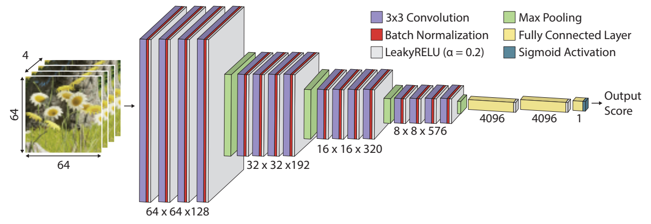


Figure 2: Overview of our aliasing detector network. The architecture is inspired by the VGG19 network, with modifications to prioritize detecting small-scale artifacts like aliasing. We implemented this network using Tensorflow and Keras.

- We use $64 \times 64 \times 4$ RGB image sequences as network input; trading off spatial pixels with temporal ones helps localize detected artifacts to smaller regions, and also enables the network to use spatiotemporal information in detecting aliasing. We tried different spatial and temporal tiles, and found that larger tiles and longer sequences worked better. However, we picked $64 \times 64 \times 4$ because it resulted in high accuracy for a practically-sized network.
- We include additional convolutions in the first few high-resolution layers, designed to capture the local nature of aliasing artifacts, and consequently fewer convolutions as the latter low-resolution layers.
- We increase the overall feature counts based on experimental observations.

Additionally, we also use batch normalization before activation of each convolutional layer, and leaky ReLU instead of ReLU for layer activations. We found both these changes to help the network achieve lower loss during training as well as validation. In total, our network architecture has 70,160,129 trainable parameters.

The final fully connected layer has a single output with sigmoid activation, and constrains network outputs to be scalars between 0 and 1 to indicate the magnitude of aliasing detected.

The following subsections provide details of our training and inference procedures. We used Tensorflow [Abadi et al. 2015] and Keras [Chollet et al. 2015] as the machine learning frameworks for our work, and a NVIDIA Titan V GPU for acceleration.

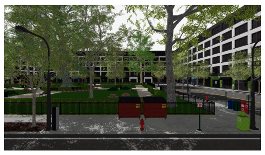
3.2 Training

We train our network to detect aliasing in $64 \times 64 \times 4$ RGB image sequences by showing it aliased as well as antialiased inputs in a supervised fashion, setting that target output as 1 for the former and 0 for the latter.

Dataset. Figure 3 shows the scenes we used for training and validation, all part of ORCA [NVIDIA 2017] and rendered using Falcor [Benty et al. 2017]. We consider several animated sequences in our dataset, and extract the first 4 of each sequence of 25 frames in each animation. This helps provide a diverse database of short animated sequences.

During training, we randomly crop each sequence into a 64×64 tile, which our network's input dimension. To add variety to our training, we also augment our data with random permutations.

(a) Classroom (600 frames)



(b) Emerald Square (600 frames)

(c) Sun Temple (2000 frames)

(d) Lumberyard Bistro (600 frames)

Figure 3: We used the above scenes as the dataset for our studies. Of these, we used 3(a), 3(b) and 3(c) for training, and 3(d) for validation. To generate short sequences from these animations, we only consider the first 4 of each set of consecutive 25 frames. This also ensures we fully cover each animation.

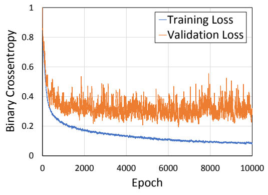
Specifically, we randomly flip our sequences along the time, height, and width axes with a probability of 0.1 each, and scale the brightness and contrast of the sequence to between 0.6 and 1.4.

For each of our sequences, we use a 1 sample-per-pixel (1spp) rendering as the *aliased* input, and 16–64 sample-per-pixel superssampled rendering as the *antialiased* reference. For scenes with simple content (e.g. Classroom), 16spp inputs were sufficient to provide stable anti-aliased images, but for scenes with more complicated content (e.g. Emerald Square), we had to use 64spp renderings to fully eliminate aliasing.

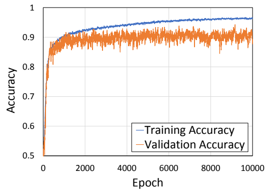
Training Procedure. For training, we use batches containing four $64 \times 64 \times 4$ sequences. We choose each sequence in a batch randomly from either the aliased or the antialiased subsets of the training dataset, perform a randomized crop to 64×64 pixels and augmentations as described above.

A key requirement of our aliased inputs (target output 1) is that those sequences must contain visible aliasing artifacts. This is not a guarantee in rendered images, since there may be image regions that are entirely covered by low frequency effects (e.g. sky) or texture maps (e.g. flat surfaces). To ensure that we don't mark those inputs as aliased, we compare the cropped, augmented sequence against the corresponding antialiased sequence to ensure that the two are sufficiently different. We tried various metrics to compute this difference (e.g. $\mathcal{L}_1$ and $\mathcal{L}_2$ norms) but got the best results with SSIM. If the SSIM between our aliased input sequence and the corresponding antialiased sequence was less than 0.97, we set the target output to 1 (aliased). Otherwise, we set the target output to 0 (antialiased) even though the sequence comes from the aliased (1spp) subset.

The above modification unbalances our dataset, resulting in far more antialiased inputs than aliased ones. To ensure that the aliased inputs with target 1 are considered equally, we weigh their loss higher than antialiased ones. For this purpose use a running count of our target outputs to compute the weight.



(a) Loss



(b) Accuracy

Figure 4: Running loss and accuracy during training. In these plots, we show the exponential moving average ($\alpha = 0.1$) of the loss as well as accuracy as it evolves over the training procedue. The validation loss and accuracy follow the trend of training loss, but slightly diverge in later epochs.

Since our target outputs are either 0 or 1, we use binary crossentropy as our loss function. For our optimizer, we use Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and no decay. To help improve generalization, we perform batch normalization between each convolution layer and its activation, and include dropout with a probability of 0.2 after each of the first two fully connected layers.

Using Keras for training on a single NVIDIA Titan V GPU takes approximately 19 seconds per epoch (256 batches), including the time taken for validation. Figure 4 shows how the training and validation loss evolves over the training process. It also shows how the accuracy of aliasing classification evolves. We get good inference results after approximately 1000 epochs, but the loss continues to improve after that. The validation losses do not improve much beyond 3000 epochs.

3.3 Inference

During inference, we simply perform forward evaluation of the network for all 64×64 tiles across four consecutive frames of a test sequence. This results in a coarse output image as shown in Figure 1, which roughly indicates the magnitude of detected aliasing across the image. This output is then useful to locate aliased regions in the image. We can also use the average output across the image as an overall measure of aliasing artifacts in the image. Inference for each $64 \times 64 \times 4$ sequence takes 7 ms as reported by Keras. See Section 4 for details.

4 RESULTS

Figure 1 shows an example output after training the detector for 5000 epochs. Please see the supplementary video for results on other scenes, a still sequence, and a downsampled real-life video. For increasing sampling densities, the detector reports consistently fewer aliasing artifacts. At 1 sample per pixel, it detects most tiles as being aliased, while at 16 samples per pixel, it detects very few tiles as containing aliasing artifacts. It also detects very few aliased tiles for image sequences antialiased using temporal antialiasing (TAA), which is the current state-of-art algorithm for high-quality antialiasing in contemporary modern video games. Since TAA is known to cause blur, conventional image metrics will tend to report a high difference against ground truth. Note that we did not include any TAA images in our training dataset.

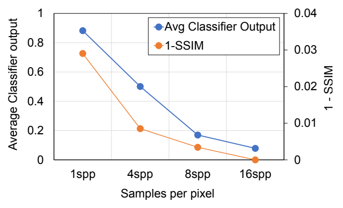


Figure 5: Comparing our output to 1-SSIM for image sequences rendered with increasing sample counts. Our output is consistent with 1-SSIM, since aliasing is the dominant artifact in these images. As shown in Table 1, our approach is able to look at aliasing in isolation, even when other artifacts are present.

Table 1: Comparing our output to SSIM and PSNR, as measured against a 16spp ground truth, for the image (and 3 preceding frames) shown in Figure 1. Our network does not require a ground truth for comparison, and unlike SSIM and PSNR, we correctly report TAA output as low aliasing.

Image Source	Ours	SSIM	PSNR
1spp	0.8822	0.9709	30.87
4spp	0.5012	0.9915	36.68
8spp	0.1697	0.9966	41.67
16spp	0.0787	1.000	-
TAA	0.0572	0.9269	29.32

Figure 5 and Table 1 shows the output of our network averaged over the image, as it varies with increasing samples per pixel. As expected, the output is close to 1 for 1spp input images, and reduces to less than 0.1 for 16spp input images. It is similarly less than 0.1 for TAA input. Figure 5 also shows that our output correlates well with (1-SSIM) when the only artifacts are aliasing related. The detector is able to do so despite having no access to a ground truth antialiased image. However, Table 1 shows that the metric does not correlate well with SSIM or PSNR when other artifacts are present. For instance, outputs of temporal antialiasing (TAA) have a low SSIM and PSNR values because the algorithm tends to trade off image sharpness for antialiasing. Our metric only measures aliasing and hence provides a similar output for TAA images and 16spp images, even though the former is significantly less sharp. In this respect, the aliasing detector is more useful when we simply want to identify aliasing artifacts in an image sequence.

5 CONCLUSIONS AND FUTURE WORK

In this short paper we have presented the first steps towards using deep neural networks as tools for analyzing image quality of rendered images. We have shown that it is possible to train a network to identify visual quality artifacts in images without looking at ground truth images. Specifically, we have demonstrated how a trained network can detect aliasing artifacts in input image sequences. We have also shown that the these networks have two desirable properties. First, they are easy to train by simply showing examples of images containing the chosen artifacts as well as those free from those artifacts. Second, their output is well correlated with the magnitude of the artifacts they are trained to detect, as well as with conventional metrics when those are the only artifacts present. It also appears that such networks learn to effectively ignore other artifacts present in their inputs, which is useful when we wish to selectively measure image quality.

One of the limitations of our approach lie in the reliability of the output. Even at 16spp input with no visible aliasing, we notice some false positives in the output (Figure 1), although when averaged, the metric appears to be much more reliable. We also expect false positives when rendered image features resemble aliasing. For example, a flickering light might appear as aliasing to the detector. Another limitation of this approach is that it is currently very expensive with more than 70 million parameters. Running inference requires several hundred invocations, which can be slow in practice.

Going forward, we would like to explore the ability for our network to learn to detect other visual artifacts that commonly affect modern renderers, like Monte Carlo noise, ghosting, and low-frequency boiling. We would also like to investigate if we can train a single network to simultaneously detect and classify multiple artifacts. Finally, we wish to explore leaner configurations of such networks which can detect rendering artifacts in real time.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

Tunç Ozan Aydin, Martin Čadík, Karol Myszkowski, and Hans-Peter Seidel. 2010. Video Quality Assessment for Computer Graphics Applications. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10)*. ACM, New York, NY, USA, Article 161, 12 pages. https://doi.org/10.1145/1866158.1866187

Nir Benty, Kai-Hwa Yao, Tim Foley, Anton S. Kaplanyan, Conor Lavelle, Chris Wyman, and Ashwin Vijay. 2017. The Falcor Rendering Framework. (07 2017). https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).

Jorge Jimenez, Diego Gutierrez, Jason Yang, Alexander Reshetov, Pete Demoreuille, Tobias Berghoff, Cedric Perthuis, Henry Yu, Morgan McGuire, Timothy Lottes, Hugh Malan, Emil Persson, Dmitry Andreev, and Tiago Sousa. 2011. Filtering Approaches for Real-Time Anti-Aliasing. In *ACM SIGGRAPH Courses*.

Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward A Practical Perceptual Video Quality Metric. https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652. (2016).

Rafat Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. 2011. HDR-VDP-2: A Calibrated Visual Metric for Visibility and Quality Predictions in All Luminance Conditions. *ACM Transactions on Graphics* 30, 4, Article 40 (July 2011), 14 pages. https://doi.org/10.1145/2010324.1964935

O. Nalbach, E. Arabadzhiyska, D. Mehta, H.-P. Seidel, and T. Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen Space Shading. *Comput. Graph. Forum* 36, 4 (July 2017), 65–78. https://doi.org/10.1111/cgf.13225

NVIDIA. 2017. Open Research Content Archive (ORCA). (July 2017). http://developer.nvidia.com/orca

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). arXiv:1409.1556 http://arxiv.org/abs/1409.1556

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.