Denoising Videos with Convolutional Autoencoders

A Comparison of Autoencoder Architectures

Tara Larrue
University of Maryland, College Park
tlarrue@cs.umd.edu

Yunchuan Li
University of Maryland, College Park
yli93@terpmail.umd.edu

Xiaoxu Meng
University of Maryland, College Park
xiaoxumeng1993@gmail.com

Chang-Mu Han
University of Maryland, College Park
cmhan74@terpmail.umd.edu

ACM Reference Format:
Tara Larrue, Yunchuan Li, Xiaoxu Meng, and Chang-Mu Han. 2018. Denoising Videos with Convolutional Autoencoders: A Comparison of Autoencoder Architectures. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, graphics researchers have harnessed the power of deep learning to solve a multitude of rendering and modeling problems. One important open problem is how to use machine learning to cut down on computation costs in photo-realistic 3-D rendering. In order to produce realistic images, film and design industries typically use ray tracing algorithms that simulate light rays interacting with model surfaces. The most common flavor of ray tracing uses Monte Carlo approaches to uniformly sample ray directions within a scene. The biggest disadvantage of this approach is the poor performance due to the large amount of light ray samples that is required to produce a clear image. If an insufficient amount of samples per image pixel is used, the resulting image will appear noisy due to the lack of light in low-sampled areas.

One way machine learning can improve ray tracing efficiency is to use reinforcement learning to progressively learn where light comes from in order to guide the light ray sampling strategy [2]. This method results in faster convergence to a noise-less image. Another approach is to use a neural network architecture called a convolutional autoencoder to denoise images rendered with a low sample count per pixel [1]. The latter post-processing approach is the focus of this paper.

A convolutional autoencoder is composed of two main stages: an encoder stage and a decoder stage. The encoder stage learns a smaller latent representation of the input data through a series of convolutional and down-sampling layers. The decoder stage expands the latent representation back to the original image. Such an architecture can be used to solve any image reconstruction task, including the restoration of a full resolution image from a noisy image (see Figure 1).

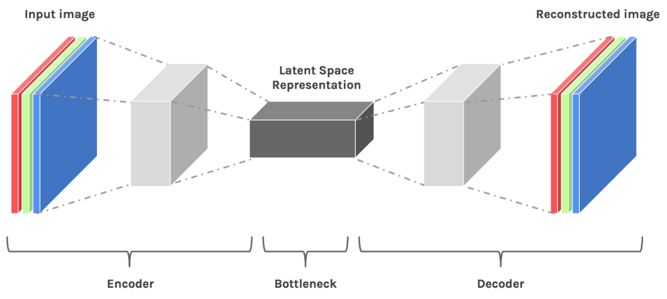


Figure 1: Basic convolutional autoencoder architecture.

While basic convolutional autoencoders have been shown to be effective at denoising images frame-by-frame, they are insufficient in reducing the temporal issues that occur when denoising image sequences. To minimize the temporal "flickering" that can occur when noise from one frame is different from the noise in the next frame, the autoencoder network must include a "memory" of previous images in a sequence so that sequential frames learn a similar latent representation. In a related neural network architecture, called a recurrent convolutional autoencoder, that memory is called a hidden state, and it is reused to compute future outputs. Still, the hidden state is anchored to the image plane. This means that if objects move beyond the convolutional receptive field within the image sequence, the network cannot effectively extract the feature evolution over time. In a 2017 SIGGRAPH presentation, researchers suggested that the addition of spatial warping of the hidden state could solve this problem [8]. Warping the hidden state based on the optical flow of the rendered scene from frame to frame will effectively allow the hidden state to "follow" moving features in the image sequence (see Figure 2.) The goal of this project is to compare how these three main autoencoder architectures: the basic convolutional autoencoder, the recurrent convolutional autoencoder, and the warped recurrent convolutional autoencoder perform in the application of denoising rendered videos.

In order to gather enough training data for this course project, we will use previously rendered videos found on YouTube and add artificial Gaussian noise to them. Although Gaussian noise does not mimic noise resulting from low sample-per-pixel Monte Carlo rendering, the task of removing Gaussian noise without blurring

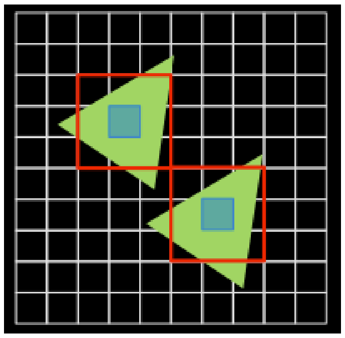


Figure 2: The hidden state in a recurrent convolutional autoencoder can be warped so that the convolutional receptive field will follow a triangle in an image sequence. [8]

is similar to the task of removing rendering noise. This pilot study should be able to demonstrate the effectiveness of each autoencoder to remove noise from rendered videos, even using this simplified training data set.

2 ARCHITECTURES

2.1 Baseline

The baseline convolutional autoencoder architecture used was based on an image-to-image translation architecture called "pix2pix" [4]. The goal of this learning algorithm is to simply translate one representation of an image into a different representation of the same image, given enough training data demonstrating the same translation. Authors of the algorithm were able to achieve this broad goal using a specialized network called a conditional generative adversarial network (cGAN). cGAN has two main pieces, the generator, which is similar to the basic autoencoder and has the job of translating an input into an output, and the discriminator, which has the job of classifying those generator outputs as real (ground truth) or fake (generated). The generator tries to "trick" the discriminator, while the discriminator tries to classify the generator's outputs correctly. The purpose of this architecture is not only to learn autoencoder parameters, but also to learn an appropriate loss function for the specific translation task. This further offloads some work to the machine, as the developer does not have to hand-engineer a loss function.

In order to minimize the amount of parameters our networks had learn, and therefore minimize training time, we decided to only use the generator portion of pix2pix with a simple L1 loss function. The pix2pix generator architecture is detailed in Figure 3 and was used as a starting point to each subsequent architecture designed for this project. This baseline architecture learns 355,905 parameters.

2.2 3-layer Recurrent Blocks

In a recent paper, NVIDIA researchers added recurrent connections to a denoising autoencoder network to improve the temporal stability of denoised image sequences [1].

We implemented a similar structure. In our architecture, each encoding block was a "recurrent convolutional block" consisting of 3 convolutional layers. The first layer processes the input features from the previous encoding block. The result is then concatenated to the previous hidden state, which undergoes the second convolution in the block and becomes new hidden state. The output of the third down-sampling convolutional layer goes undergoes batch normalization and becomes the final output of the encoding block. All convolutional layers are followed by one activation function (ReLU). The structure is shown in Figure 4a. This architecture has 3,609,024 parameters.

2.3 3-layer Recurrent Blocks with Spatial Transformer

A presentation on open problems in real-time rendering from SIGGRAPH 2017 discussed the addition of warping the hidden state in order to improve denoising recurrent architectures [8]. The proposed addition used pre-computed dense motion vectors between frames in image sequences to translate the hidden state between iterations in a recurrent block. Although we were not able to implement this type of warping during the duration of this project, as a first approximation to this warping, we added a spatial transformer into the recurrent block architecture after the concatenation with the hidden state. The architecture is shown in Figure 4b. Spatial transformers allow the spatial manipulation of feature maps in order for the network to learn features regardless of spatial variations [5]. It works by learning an appropriate affine transformation matrix to transform the hidden state. By adding a spatial transformer into our network after the hidden state has been updated with features of the new input frame, this will effectively warp the updated hidden state to align spatially with the new frame. This architecture had 6,656,466 parameters.

3 TRAINING AND TESTING DATA

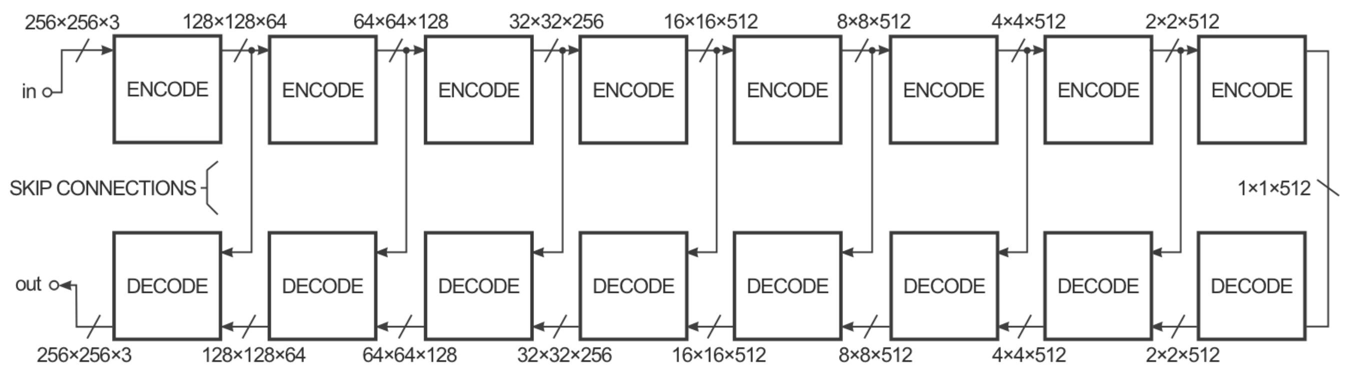
4 different video segments were downloaded from YouTube and used to train and test each of the autoencoders. Each video is a previously rendered 3-D model presented as a fly-through animation. 3 of the 4 the videos were used in training, and 1 video was completely withheld as a separate test set. In addition, 20 percent of each training video was withheld as additional, smaller test sets. The training videos are named Community [6], Church [7], and Bungalow [9]. A total of 2598 frames between the 3 videos were used to train each autoencoder. The biggest test video is named Kitchen [9], which consists of 1437 frames. Among the smaller test videos, Community consists of 298 frames, Church consists of 187 frames, and Bungalow consists of 206 frames. Each autoencoder was trained for 400 epochs.

To pre-process the training data, the YouTube videos were downloaded in HD definition (720p) and frames were extracted at 24 frames per second. Frames were then resized to 256x256 pixel images. Noisy images were produced synthetically by adding random Gaussian noise with a sigma of 30 to each frame.

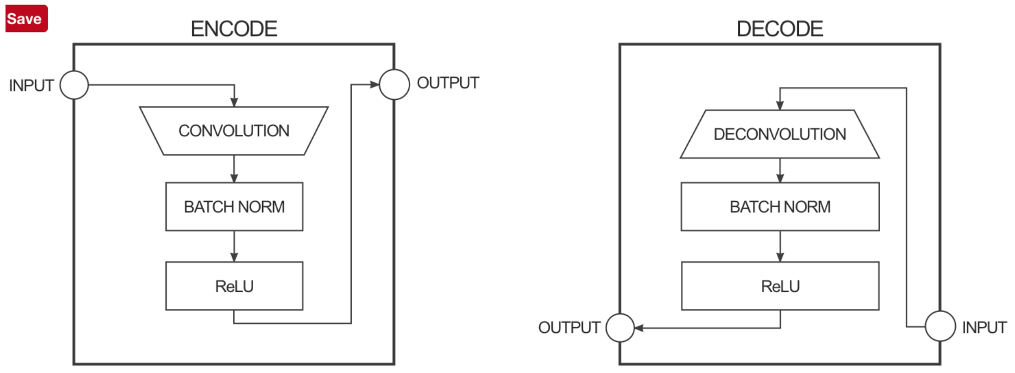
Training was performed with python Tensorflow implementations on a GeForce GTX 1080 GPU.

4 RESULTS

Statistical analysis including the mean square error (MSE), peak signal-to-noise ratio (PSNR), structural similarity (SSIM), sharpness loss, and temporal L1 loss from each of the architectures and test sets



(a)



(b)

Figure 3: The baseline architecture is a convolutional autoencoder based on "pix2pix," implemented in Tensorflow [3]. (a) the baseline architecture has 8 convolutional encoding layers and 8 deconvolutional decoding layers with skip connections, (b) every encoding and decoding block includes a convolution/deconvolution operation that downsamples/upsamples, batch normalization, and a ReLU activation function.

can be seen in Figure 5. These image quality assessments evaluate the temporal and spatial performance of the 5 cases, including the convolutional autoencoder (Baseline), 3-layer recurrent (Rec), 3-layer recurrent with spatial transformer (WarpRec), bilateral filter (BiF), and Noised frames (NF). BiF means denoising without using a learning algorithm. NF indicates the comparison between noisy input images and ground truth images.

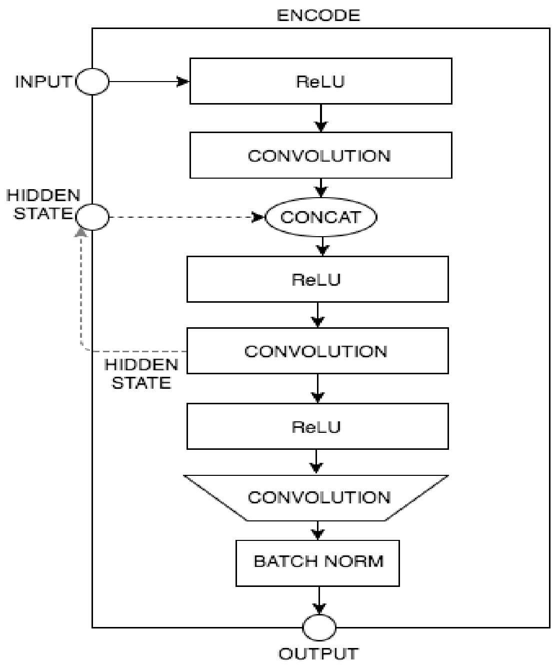
The result of sharpness appears in Figure 5c. The images processed by WarpRec are blurrier than the others. The result is in line with our subjective visual quality assessment. Furthermore, the deterioration of blurriness in the images might explain the poor performance of WarpRec in MSE and PSNR because MSE and PSNR measures the intensity difference between images, shown in Figure 5a and Figure 5b. The averaging between pixels due to blurriness increases the intensity difference between each pair of the output images and the target images.

Temporal loss is an important metric for this study, as it quantifies the temporal incoherence of the sequence. This is the flickering between frames that recurrent architectures have been shown to reduce. Figure 5e shows that there was some temporal loss reduction for all the cases, while the cases using the learning algorithm outperforms the others. However, the result is inconsistent with our expectation, in which WarpRec is the best case. One explanation for this might be that the blurry edges and the coloring change of locality of images in WarpRec cause significantly negative effects on the output images.

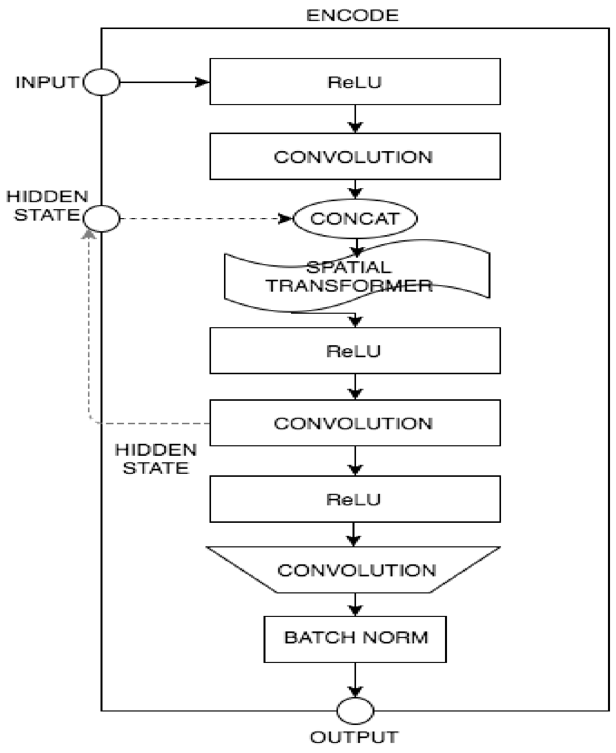
SSIM measures how similar two images are [10]. Figure 5d indicates that Rec provides the most promising result over the others. It is clear that the learning algorithm cases perform better than non-learning methods in both SSIM and Temporal L1 Loss. Unexpectedly, WarpRec is the worst case in MSE, PSNR, and sharpness loss.

5 DISCUSSION

One explanation for the poor results from our warped recurrent architecture is the lack of training time compared to the number of parameters this network had to learn. We trained each network with the same amount of training data and the same amount of

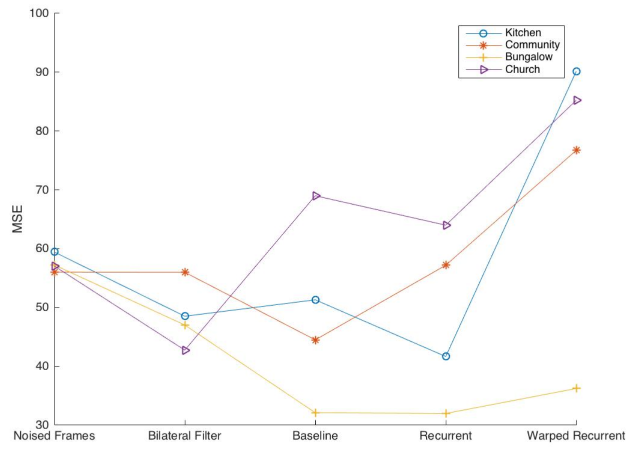


(a) Internal structure of a recurrent block. The output from the previous encoding layer becomes the input of the next encoding layer. The hidden state defined in the middle of the block is used in the same block at the subsequent time step.

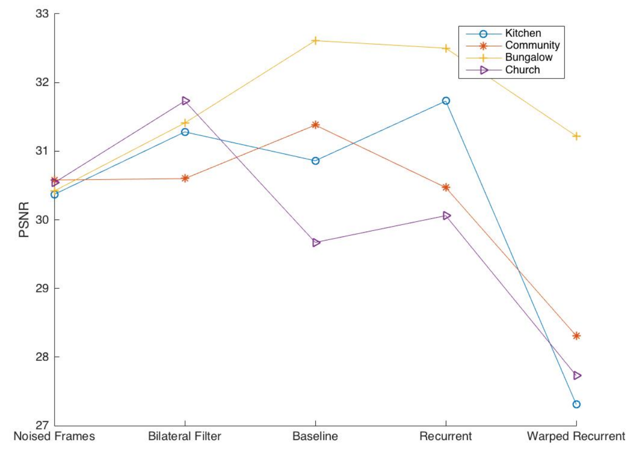


(b) Internal structure of a recurrent block with spatial transformer network.

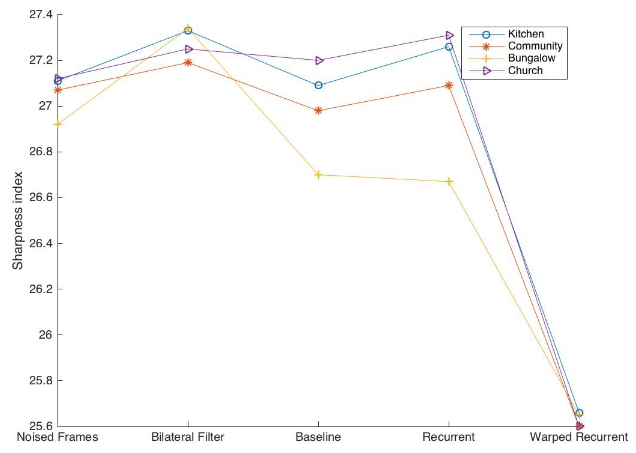
Figure 4: Recurrent block architectures.



(a)

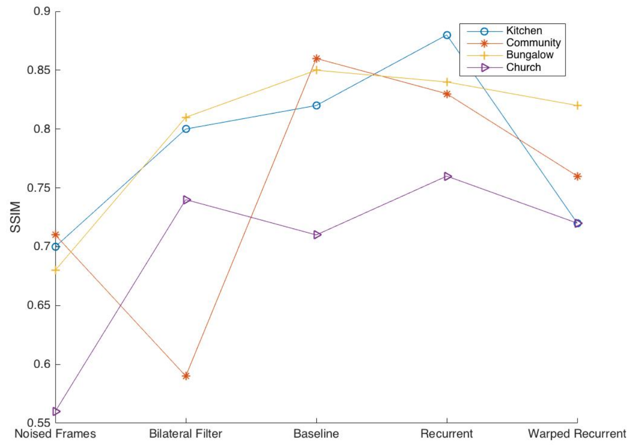


(b)

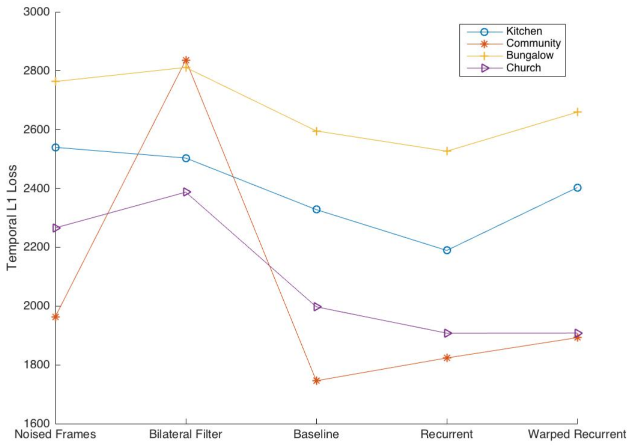


(c)

Figure 5: Statistical results of testing each test set with NF, BiF, Baseline, Rec, and WarpRec. (a) mean squared error of all frames, (b) peak signal-to-noise ratio of all frames, (c) sharpness loss of all frames.



(d)



(e)

Figure 5: (d) is structural similarity of all frames, (e) is temporal loss of all frames.

epochs. However, in order to make an apples-to-apples comparison between network architectures, architecture complexity needs to be considered when determining training parameters. On the more promising side, we did see artifacts present in results from our recurrent architecture correct by the warping.

One addition we would like to make would be to add the temporal L1 loss function to the network loss function. For each architecture, the loss function was simply the L1 distance frame-by-frame. This does not sufficiently account for temporal stability of the entire image sequence. This addition would likely make a significant difference in our results.

Lastly, we would like to add functionality to incorporate pre-computed dense optical flow maps into the warping architecture. The spatial transformer network can be modified to accommodate this type of spatial warping.

REFERENCES

[1] Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073601
[2] Ken Dahm and Alexander Keller. 2017. Learning Light Transport the Reinforced Way. *CoRR* abs/1701.07403 (2017). arXiv:1701.07403 http://arxiv.org/abs/1701.07403
[3] Christopher Hesse. [n. d.]. Image-to-Image Translation in Tensorflow. ([n. d.]).
[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR* abs/1611.07004 (2016). arXiv:1611.07004 http://arxiv.org/abs/1611.07004
[5] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. *CoRR* abs/1506.02025 (2015). arXiv:1506.02025 http://arxiv.org/abs/1506.02025
[6] João Marques. [n. d.]. Project fly-through LUMION 3D. ([n. d.]).
[7] Render3DQuickly. [n. d.]. 3D architectural animated fly through of Eagle creek church | 3d rendering video | 3 d walk through. ([n. d.]).
[8] Marco Salvi. 2017. The Future of Real-Time Rendering. In *Open Problems in Real-Time Rendering course*. SIGGRAPH, Los Angeles.
[9] TheRedCottageVideos. [n. d.]. 3d Virtual Tour of the Independence Bungalow. ([n. d.]).
[10] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing* 13, 4 (2004), 600–612. http://dblp.uni-trier.de/db/journals/tip/tip13.html#WangBSS04