Removing Atmospheric Turbulence From Video Using Recursive Neural Networks for Noise Kernel Prediction

Thomas Arrizza
University of Michigan: Ann Arbor
500 S State St, Ann Arbor, MI 48109
tarrizza@umich.edu

1. Introduction

Atmospheric turbulence is a physical phenomenon where changes in air density, temperature, humidity, and many other factors cause large bodies of air to move chaotically. When a scene is observed through this turbulence it appears distorted with a combination of a visual rippling, localized blurring, and variations in brightness. These effects are pronounced when either viewed over long distances, or local variations in temperature are high. Fields such as long range photography, astrophotography, high altitude surveillance or cases where temperature variations are high like desert environments, near asphalt, or exhaust from cars and buildings are particularly impacted by atmospheric turbulence. To remove this effect, typically researchers model the effect of the atmosphere as

$$y = [D_t(H_t(I))](x) + n_t(x) \tag{1}$$

where y is the observed frame, x is the true frame, D_t is the geometric distortion function, H_t is the blurring operator, and n is noise [3][10]. A simplification of this function combines D_t and H_t into a single composite function $Y = D(x)+n$ This function is unfortunately non-invertible and there have been many attempts to solve it.

In this project, I explore the use of a typical U-net denoising architecture augmented with a recurrent neural network (RNN) block. The addition of an RNN block allows the model to accumulate information from frame by frame and determine which features of the sequence are are atmospheric distortions and which parts are ground truth objects that are moving. Using the features accumulated by the RNN block, a U-net block can then remove noise and distortion while leaving the ground truth features intact. In addition to the RNN block, I restrict the architecture to only use causal information and the model is trained only using the current or past frames and not future frames. I also prioritize inference speed and model simplicity. This results in a real-time, causal, atmospheric denoising algorithm with better deployability in real world applications.

2. Related work

Removing atmospheric turbulence from images and video has been explored using a variety of methods by researchers in the last 20 years. Some methods model it has a point spread function, employing blind deconvolution to estimate the ground truth scene[7][2]. Others use a mix of image processing tools, like image warping using optical flow or Robust Principal Component Analysis, to take advantage of scene properties[3]. These often come with severe limitations. Often times the model doesn't work if the scene has moving elements as knowing when motion is a product of distortion or an object moving requires knowledge about the structure of the scene that may not be available. These methods also can be computationally expensive and are typically employed in offline methods.

More recently, deep learning architectures have been shown to perform well at removing noise from images and video more efficiently while maintaining high accuracy reconstructions of the ground truth scene[4][8],[1][9]. Architectures range from relatively simple CNN architectures[4] that reconstruct the scene directly, to adversarial architectures[8], with the most recent being those that use U-nets architectures[1]. The body of work related specifically to removing atmospheric distortions from video using deep learning is relatively small. One cause of this is training data is rare as both noisy examples and noise free examples of the same scene need to be provided for learning. One way of achieving this is to capture a static scene and either apply intense heat in front of a camera or move further away with increased zoom[5]. These methods don't transfer well to video if the subject of the scene isn't static, so the best solution is simulation.

Compared to research removing atmospheric distortions, removing Gaussian noise from both images and video has significantly more popular, with application not just in denoising, but now in image generation in diffusion model[6]. A recent successful method for noise removal in video is to feed multiple frames into a pyramid of U-net blocks. This

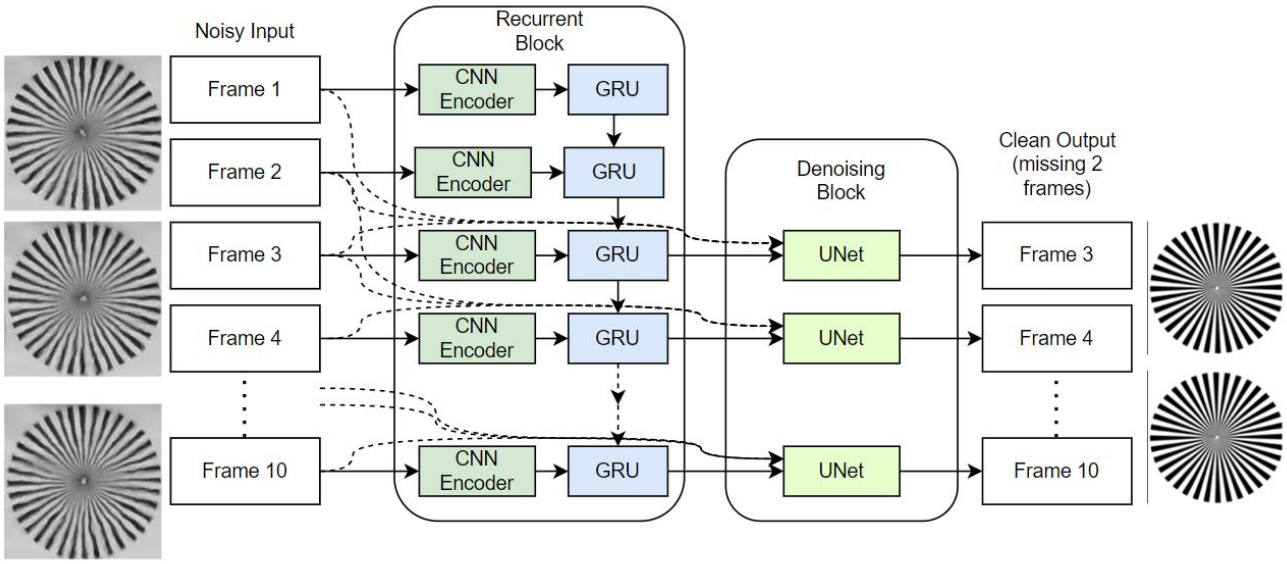


Figure 1: Overall Model Architecture

method, FastDVDnet[9], breaks a video sequence into sets of five frames, two before the current frame and two after. Then this is again broken up into three sets of three frames and each are passed to a U-net denoising block. The output of this block is again passed a final denoising block which predicts the noise residual of the current frame and produces a clean output. This gives better frame continuity and reconstruction accuracy while simultaneously enjoying high inference time. Due to these properties and the overlap in architecture with those that remove atmospheric noise, this model servers as the foundation for architecture in this project.

3. Method

To remove atmospheric turbulence from video, I implement a two stage model that first captures the composite noise kernel D by recursively sampling frames for the entire clip length, then uses these noise features along with the current and two prior frames to predict output clean frames. The architecture of my model can be seen in Figure 1. This two stage architecture allows the model to perform better frame-to-frame continuity while preserving a high individual frame signal to noise ratio. A clip length of ten is chosen as it is long enough that the short window performance of the denoising block and the long window performance of the recurrent block can be shown while reducing computational complexity.

The first stage is the recurrent block, which consists of a relatively simply convolution neural network (CNN) encoder that generates features for a two layer RNN. The architecture for the encoder CNN is very similar to the encoder of the denoising U-net and can be seen in Figure 2. Its purpose is to convert each frame into a feature map that reduces the dimensional of pixel space into a small latent space. This conversion should be independent of the frame in context of its video sequence. The RNN is then able to operate in the significantly smaller latent space and accumulate these features to determine D based on context from

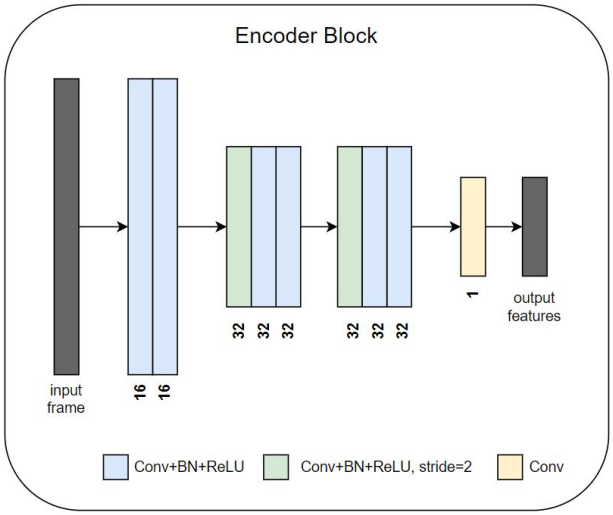


Figure 2: Encoder Architecture

the full sequence. The RNN in this implementation are two layers of gated recurrent units (GRUs). GRU was chosen only for its fast training capabilities. The long term memory benefits LSTM and GRU over a simple RNN are unlikely to make a large difference as the sequence length is relatively short, but if the clip length is increased significantly, the benefits may become more apparent. The output of the recurrent block is then unpooled to the original size of the input frame as a single channel of noise features and passed to the denoising block.

The second stage is the denoising block composed of a single U-net architecture, modified from the architecture used by FastDVDNet[9] for causal inference. This denoising block takes three input frames and the output of RNN blocked stacked as a total of 12 channels (3 for RGB plus 1 for the noise feature channel multiplied by 3 frames). The output is then subtracted from the input noisy frame so that the model learns to predict the noise residual rather than directly predicting the output frame. This was done for similar reasons to that of the original paper, where predicting the output frame caused the model be slow to converge and produce hazy, low quality images. In this architecture, the model requires two previous frames to predict an output frame, and so the first two output frames are discarded as they would either be generated from padding images that don't contribute with meaningful content for training. In a real world setting, these would be padded with duplicates of the first frame at inference time.

4. Experiments

4.1. Dataset

The dataset for this project is a subset of the Kinetics 400 dataset[1]. This dataset contains short, high quality videos of 400 different types of common human-action activities lasting at least 10 seconds. Originally this dataset is intended for use with video recognition tasks and is the smallest amongst the similar Kinetics 600 and Kinetics 700 datasets, with approximately 10 million clips. For space and computational limitations, this model is only trained using a subset of 500,000 clips separated into ten frame segments. Each clip is then normalized to [-1,1] randomly resized and cropped to 128x128 pixels for training. For evaluation and testing purposes, the dataset is further broken up into a %80-10-10 split for training, evaluation, and testing respectively.

4.2. Training Details

As part of training, each input frame is distorted in way to approximate the atmospheric turbulence function defined by Equation 1. In the following experiments 1 is simplified to only include H_T. Each clip is distorted using a Gaussian filter applied to patches of each frame. At each patch the variance of the filter and the kernel size is randomly chosen. This creates a non-invertible, spatially localized blurring function which simulates the atmospheric turbulence effect. The range of the noise is then varied between batches to simulate the effect of different intensities of turbulence.

[1]https://www.kaggle.com/datasets/rohanmallick/kinetics-train-5per

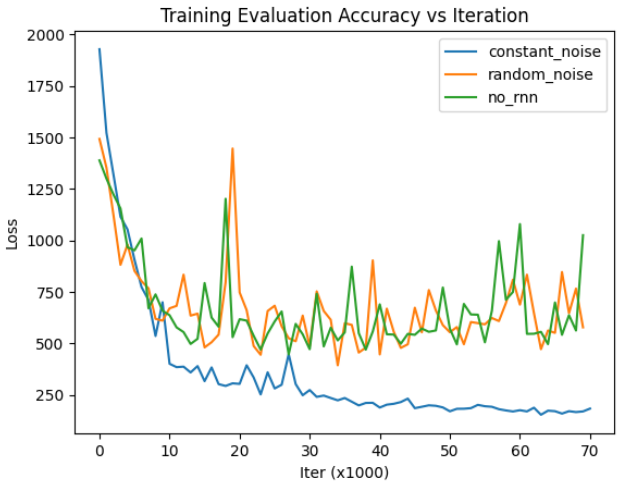
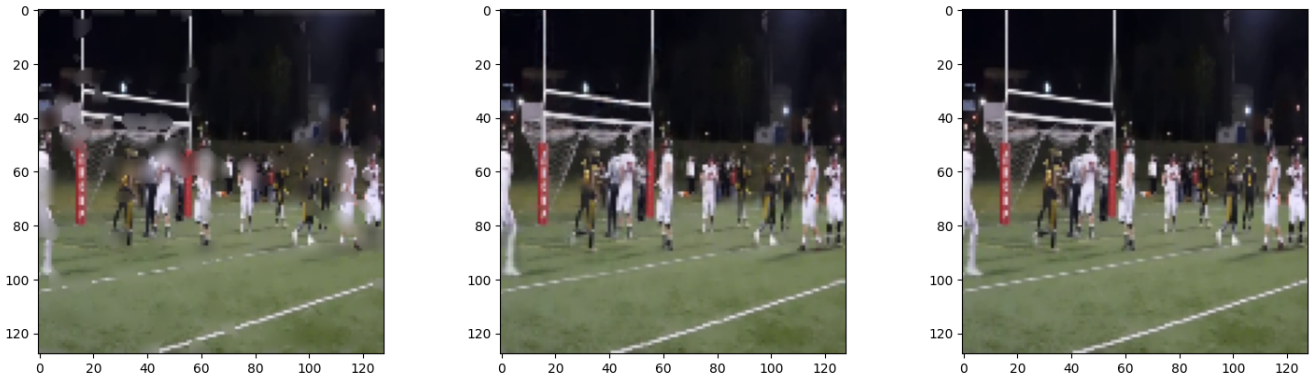


Figure 3: Training Loss: random_noise is the full model, constant_noise is the model evaluated when noise intensity is held constant, and no_rnn is the model without a RNN block.

The model was trained for 70,000 iterations over 5 epochs with a batch size of 16. In addition to the full model, two additional models were trained to evaluate performance:

- A model with the RNN removed
- A model where the distortion intensity is held constant

Removing the RNN demonstrates how its addition affects performance for both image quality and frame-to-frame continuity. Holding the distortion intensity constant demonstrates how the model fits to the noise kernel and whether the model is learning to denoise independently of the intensity of noise, or it has learned the underlying noise function. Training progress is shown in Figure 3. The instability for the full model and model where the RNN is removed stems from the nature of the loss function when the noise intensity from iteration to iteration. Higher noise iterations tend to produce higher loss curves, so plotting loss per iteration spikes at the same time as instances of high noise intensity. However, the average loss goes down throughout training, which is visible in the training of the model when noise intensity is held constant.



(a) Noisy Frame (b) Reconstructed Frame (c) Ground Truth

Figure 4: Qualitative results: Sampled from the first output frame using the full model.

Model	PSNR↑	SSIM↑	ST-RRED↓	Infer(ms)↓
Baseline	24.13	0.9388	0.291	-
No RNN	27.01	0.9552	0.196	**1.60**
Cnst. Noise	27.48	0.9572	0.118	1.81
Full Model	**29.44**	**0.9737**	**0.087**	1.80

Table 1: Quantitative results: Full model performs best in quality metrics and remain close for inference time. Baseline refers to reference noisy image.

4.3. Results

Model performance is evaluated using a mix of per-frame quality metrics, full clip continuity performance, and inference time. Evaulation is performed on 1,000 random samples from the test dataset and averaged. Each sample has random intensity of noise, with a maximum intensity slightly higher than during training for any model. Quantitative results can be seen in table 1 and qualitative results can be seen in figure 4. The full model performs significantly better than either baseline model for single frame performance. With constant noise, the model tends to over fit the level of training noise and struggles during testing when noise intensity varies. The quality also drops when the RNN is removed, demonstrating that it aids performance for each frame. ST-RRED[2] (Computes Spatio-Temporal Reduced Reference Entropic Differencing) is a measure of quality by frame differences and is a metric for full clip performance. Removing the RNN yields a large performance drop compared to the other models and demonstrates the RNN blocks importance to frame-to-frame consistency. The full model performs the worst in inference time, but still is very fast. The model is able to denoise frames at 574.1 FPS for the native 128x128 which could scale to 14 FPS at 480p (a subject for future work). Overall, the model performs significantly better without any ablations.

[2]Implemented with https://www.scikit-video.org/stable/modules/generated/skvideo.measure.strred.html

5. Conclusions

This project demonstrates that utilizing an RNN in conjunction with a more typical U-net denoising block yields a noticeable gain in performance for removing atmospheric turbulence noise from video. Our results suggest that the hidden layers of the RNN contains information that helps aid in modeling the atmospheric noise function. Not only does it aid in frame-to-frame continuity, but also improves the overall quality of the reconstruction for each frame. Unfortunately the scope of this project is limited due to time and computational resources. There several opportunities to expand on it in future work however.

An immediate extension to this work is adjusting the model to work with higher resolution images. This can be done by breaking higher quality images into more manageable chunks and stitching them back together, but fine-tuning needs to be done to ensure artifacts are generated at the border between chunks. This project also simplifies to only a single training/evaluation dataset with one type of noise. For a more rigorous model, additional datasets, like Otis[5], larger Kinetics datasets, or the DAVIS test set, can be used in conjunction with spatial warping noise can be used during training and evaluation. These additions are necessary for a better comparison to other competing work.

References

[1] Nantheera Anantrasirichai. Atmospheric turbulence removal with complex-valued convolutional neural network. *Pattern Recognition Letters*, 171:69–75, 2023.

[2] Nantheera Anantrasirichai, Alin Achim, David Bull, and Nick Kingsbury. Mitigating the effects of atmospheric distortion using dt-cwt fusion. In *2012 19th IEEE International*

Conference on Image Processing, pages 3033–3036. IEEE, 2012.

[3] Md Hasan Furhad, Murat Tahtali, and Andrew Lambert. Restoring atmospheric-turbulence-degraded images. *Applied optics*, 55(19):5082–5090, 2016.

[4] Jing Gao, Nantheera Anantrasirichai, and David Bull. Atmospheric turbulence removal using convolutional neural network. *arXiv preprint arXiv:1912.11350*, 2019.

[5] Jérôme Gilles and Nicholas B Ferrante. Open turbulent image set (otis). *Pattern Recognition Letters*, 86:38–41, 2017.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[7] Chun Pong Lau, Yu Hin Lai, and Lok Ming Lui. Restoration of atmospheric turbulence-distorted images via rpca and quasiconformal maps. *Inverse Problems*, 35(7):074002, 2019.

[8] Shyam Nandan Rai and CV Jawahar. Removing atmospheric turbulence via deep adversarial learning. *IEEE Transactions on Image Processing*, 31:2633–2646, 2022.

[9] Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1354–1363, 2020.

[10] Xiang Zhu and Peyman Milanfar. Removing atmospheric turbulence via space-invariant deconvolution. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):157–170, 2012.