Frame Rate Upscaling with Deep Neural Networks

Ted Xiao
Machine Learning at Berkeley
Email: tx@berkeley.edu

Raul Puri
Machine Learning at Berkeley
Email: raulpuric@berkeley.edu

Gautham Kesineni
Machine Learning at Berkeley
Email: gkesineni@berkeley.edu

Keywords—Frame rate, interpolation, computer vision, video, Convolutional Neural Networks, Generative Adversarial Networks

I. Introduction

Frame interpolation is a computer vision task that is largely performed on real world video data to increase the number of frames per second. Video streaming could benefit greatly from this technique, as it could allow for streaming of lower frame rate video that could then be interpolated to generate more frames and produce smoother high-framerate video. Currently approaches leave a lot to be desired in that the images produced are blurry and look like simple averages of already existing frames due to linear interpolation techniques. This is passable in real video (non-animated), as the blurry image helps evoke a sense of motion. However, 2D animated videos have sharp, defined individual frames; blurry interpolated images do not fit with the overall style of the videos, and are easily detected by the human eye.

Furthermore, enhancing these techniques to photorealistic levels could have huge potential in animation. By computing the surrounding frames and interpolating the middle ones, computational effort to produce both live and pre-rendered animations could be greatly reduced. What we attempt to achieve in this project is to use deep learning to enhance the quality of interpolated frames. We want to develop a robust algorithm for both real world and 2D video frame rate upscaling.

II. Previous Work

Traditional frame interpolation usually separates the image into components to measure the motion vectors of the image. These vectors are used to find the location of different blocks at different points in time. This generally works well but is not ideal. The algorithm knows nothing about the way objects move in the real world and assumes all motion is linear. When frame rates are very low, the artifacts of this method start appearing. For this reason, we believe there is great promise in deep learning (DL) methods for this problem.

Proper frame interpolation requires an understanding of the subjects and the motion of the subjects within frames. Deep Convolutional Neural Networks have shown promise in numerous computer vision (CV) tasks, and shows similar promise in this task. They have been used previously for this task on real world video and with basic success on 2D animation in [1], whose work we hope to extend and improve upon.

III. Dataset

There is no shortage of data for this problem, which is a common issue for many domains. Since this project focuses on a generative application of Deep Learning, we were able to leverage existing videos from YouTube as training data. For a given video V with n frames, we are able to generate a training dataset of size $n-2$, where the input features are frames i, $i+2$ with label frame $i+1$ for $i \in [1, n-2]$. We scraped 3 videos from YouTube to train, resulting in roughly 3500 frames.

Specifically, we scraped two animated videos and one real-life video. They are:

1) "Moving Circles": https://www.youtube.com/watch?v=FTFHaZ
2) "Lion King": https://www.youtube.com/watch?v=K5lEJlbEgz4
3) "Obama Interview": https://www.youtube.com/watch?v=xXH5agV7skw

IV. Preprocessing

We followed two avenues of preprocessing. First, we experimented with using both grayscale and RGB frames as input to measure performance for both. Since the grayscale preprocessing is a simpler input format, we expected to see better performance; our goal was to get performance on RGB to the same standard. We also normalized all pixel values by dividing by 255 as is standard.

Second, we utilized a sliding window approach to generate data. First, we split up a video into two halves such that frame $X_i, X_{i+2}, X_{i+4}, x_{i+6}, ..$ is our input and $X_{i+1}, X_{i+3}, X_{i+5}, X_{i+7}, ..$ are our labels.

For a window of size $n = 4$, where n is always an even integer, we would try to predict frame X_{i+3} from X_i, $X_{i+2}, X_{i+4}, X_{i+6}$.

V. Areas of Inquiry and Prior Work

We have identified three axes of experimentation for this project: loss functions, type of neural net model, and image super-resolution (eliminating blur effects).

A. Loss Functions

Initially we planned on experimenting with the following four functions: L1 norm, L2 norm (MSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity Index Measurement (SSIM). These loss functions are based on work from [1] and [2].

$$PSNR(Y, Y') = 20\log(s)10\log(MSE(Y, Y'))$$
$$SSIM(Y, Y') = \frac{(2_{\mu Y'\mu Y} + c_1)(2_{\sigma YY'} + c_1)}{(\mu^2_{Y'_;} + \mu^2_Y + c_1)(\sigma^2_{Y'} + \sigma^2_Y + c_2)}$$

Finding an appropriate loss function is critical to training deconvolutional networks. However, due to time constraints we narrowed the scope of our inquiry to just mean squared error (MSE).

B. Model Goals

Based on our analysis of past work, there are three high level goals for our net to achieve with regards to frame input:

1) How to overcome a Neural Network's inherent inability for stochastic prediction (nets tend to be better at performing averages)
2) How to keep pertinent information from the input for that stochastic prediction
3) How to best capture, learn, and utilize optical flow information to perform our frame generations

With regards to the first goal one could modify the input block in the architecture of [1]. Instead of simply processing two input data points and combining the processed output as input into the convolutional block of our autoencoder-like structure, one could leverage a sliding window RNN/LSTM component in order to mitigate behavior resulting in summation/averaging of input information which the current architecture utilizes.

With regards to the second goal: prior work in [1] and [3] indicate that utilization of residual connections can help preserve key information about the input frames to be utilized during generation.

Lastly, prior work in [2] shows that use of a VAE, a.k.a. Variational Autoencoder, (with or without residual connections) can help learn a representation of information from the input frames, along with pertinent information about observable optical flow, to help predict the transition frame between the two input frames.

However, for the scope of our experiments, we focused primarily on the last goal and how to best utilize techniques like the VAE without residual connections for capturing information about optical flow and generating transition frames.

C. Image Super-Resolution/ De-blurring Motion

Another avenue in producing crisp, upscaled videos is to eliminate blur by leveraging existing work in image super-resolution ([2], [3], [4]). There are two separate techniques on this front. After taking output imagery from the convolution (or deconvolution in the case of a VAE) block one could achieve superior image generation by refining the image with the use of either Multi-frame Super-resolution (MFSR) or with the use of Generative Adversarial Networks (GANs), as introduced in [5].

For MFSR, one can take the generated output image frame and the original input images all concatenated into one featurization and then feed it into an MFSR block to get the final output image.

Regarding GANs, we can use them as an adversarial component to the specified network. By attaching an adversarial convolution component to the end of our model that distinguishes interpolated and real frames, we can hope to train the network to remove artifacts created in the interpolation process. Through this additional step, one can achieve better accuracy in the realism of predicted frames.

Another avenue for exploration in this realm is to train a deconvolution component adversarially on a encoding block of frozen pre-trained classification networks such as VGG. This will create a strong deconvolution component that works well within the pre-trained space. If the convolutional/encoding network is made of pre-trained components, we expect this deconvolution network to work very well in conjunction.

For the scope of our experiments we focus mainly on GANs as a new avenue of exploration, as prior work has established the success of producing crisp frames given a series of surrounding frames.

VI. Evaluation

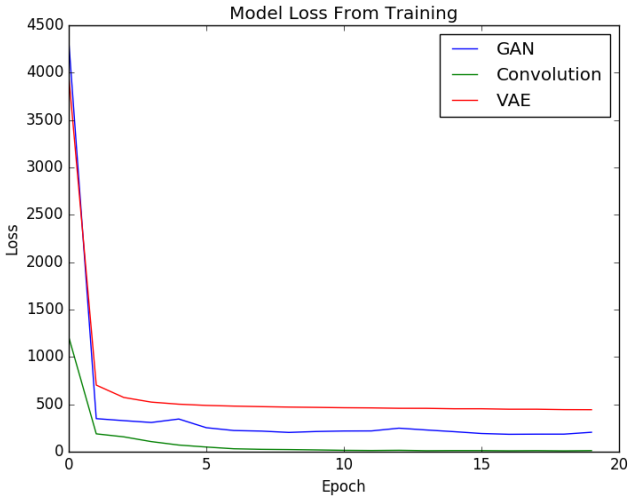
With our MSE loss function found that the loss function allowed our neural networks to generalize well to multiple videos and upscale a wide corpus of test videos. We used existing evaluation methods (for instance, as seen in [1]) to evaluate the success of our different models. Specifically, we attempted to use main evaluation measure was Interpolation Error (IE), defined as the root-mean-square (RMS) difference between the ground-truth image and the estimated interpolated image:

$$\text{IE} = \left[\frac{1}{N}\sum_{(x,y)}(I(x,y) - I_{GT}(x,y))^2\right]^{\frac{1}{2}}$$

In addition, we also used qualitative measures of determining the success of different models, as we found that the human eye discerned interpolation results as well as if not better than many error metrics. We find this due to the visual nature of computer vision tasks, especially as applied to frame interpolation. This is standard practice in other similar fields of study such as image stylization, where the quality of the intended result can't solely be expressed in a mathematical value.

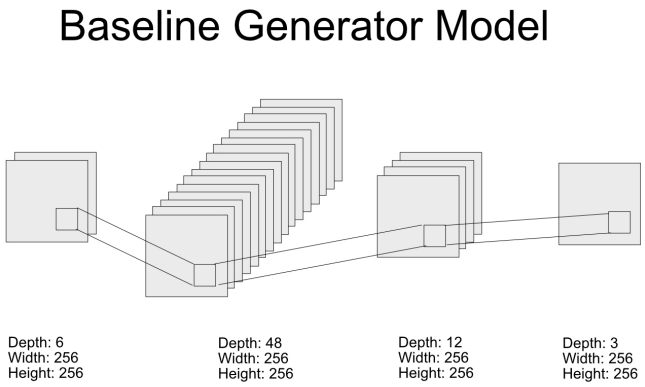
VII. Models

We tried several methods in our approach: Simple Convolution (Baseline), Deep Convolution, Variational AutoEncoders, and GANs. More details on these along with the results can be found in this poster. Below is the loss over time while training each of these models.



A. *Baseline Model*

Our baseline model was a simple 3-layer convolution.

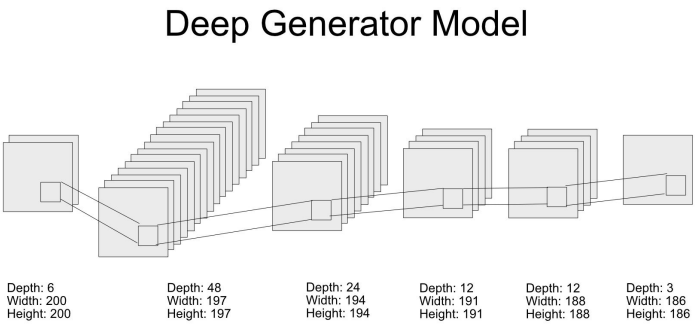


A convolution layer applies several learned convolutions to each input image. A convolutional neural network stacks these layers so that later convolutions learn on the outputs of earlier layers. Although these are typically used for image recognition because of their ability to ignore spatial information (convolutions slide over the input), they can also be applied to other tasks. Because convolutions allow for translations, the network should intuitively be able to learn the movement of static objects in video.

The biggest change we made here from a standard 3-layer convolutional neural network was to use $\tanh$ instead of ReLu activation at the last layer and to normalize the inputs as described in the preprocessing section; however, these are all pretty standard techniques for generating images via deep learning.

B. *Deep Convolution Model*

This was an extension of our first model. We increased the number of layers to 6. The biggest change is that we swapped the $\tanh$ activations with ReLu and used a linear activation for our last layer. We found that $\tanh$ would favor the values at 1 or -1 while a linear activation would treat all equally, as pixel values should be.

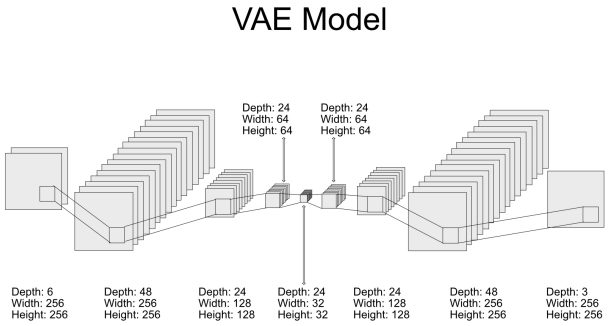


We only applied 6 layers because we were limited by the amount of memory available on our hardware. Deeper models should perform better at this task and capture faster movement. Given more time and stronger hardware, we would have explored even deeper convolutional models.

C. *VAE Model*

Variational AutoEncoders (VAEs) consist of an encoding block made of convolution layers to learn feature representation (encoding) of the image, and then a deconvolution (decoding) block to take the feature representation and generate an image from it. VAEs force the inputs to go through a bottleneck of a lower dimensional latent space so that the model learns to abstract information. Ideally, the image would learn motion in its latent space and use that to interpolate the appropriate frame in its decoding block.

We tried using Variational AutoEncoders (VAEs) in our experiments. We focused on a simpler implementation of VAEs that do not rely on residual connections as that was natively supported within Keras. We attempted to also apply residual connections but were not fruitful in our efforts for various reasons. Due to the compressive nature of VAEs our results generally ended up being rather blurry as shown in the results section.

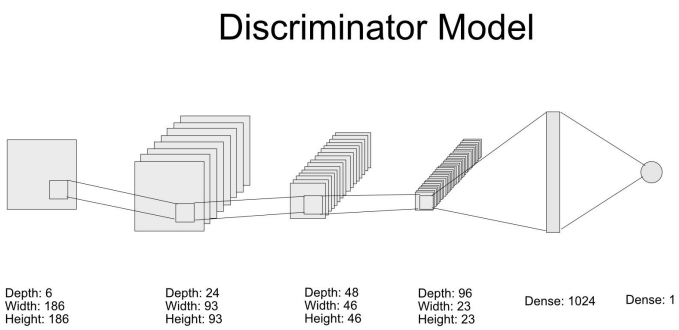


D. *GAN Model*

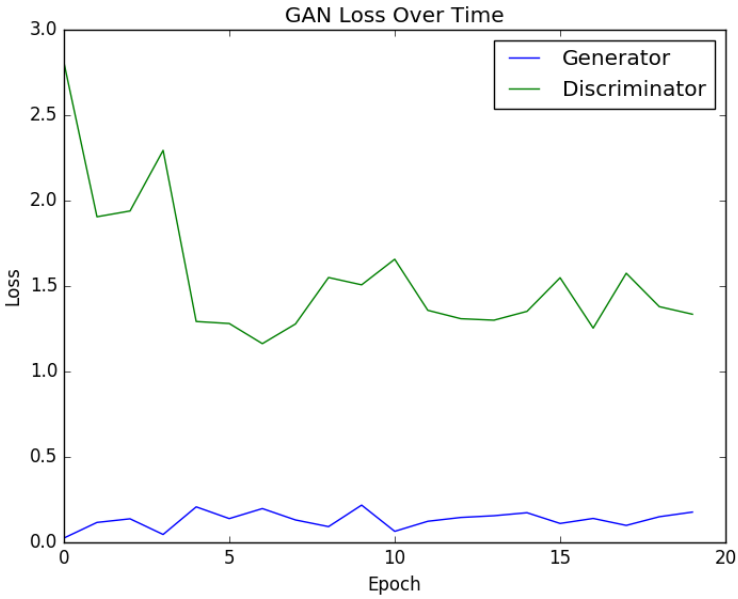
Generative Adversarial Networks are typically composed of two parts: the generator and discriminator. The generator takes as input random numbers that represent a lower dimensional latent space and use them to create an image that fools the discriminator. The discriminator, on the other hand, will receive images made by the generator and real images. It attempts to find which images are real and which were generated. These two networks train adversarially so a good

indicator that the system is working is that the losses of both networks remain close.

The Generative Adversarial Network (GAN) model is an extension of our Deep Convolution Model. We adopt the Generator-Discriminator architecture introduced in [5]. However, unlike typical GANs, we supply the input as the before and after frames and attach a normal discriminator component. We applied both MSE and GAN updates to the generator component to converge faster.



The Discriminator architecture is shown above. The generator architecture is exactly the same as our Deep Convolution Model in earlier sections. The GAN network saw a disproportionate convergence for the Discriminator and Generator components. This is not ideal but we were unable to improve the results in our experiments.



VIII. Results

The results of the different models can be seen on the next page. Qualitatively, we do indeed find that our models interpolate successfully to some degree. Additionally, the visual performance of the various models match the plotted loss history seen in Section VII.

For a video demonstration of our Deep Convolution Model upscaling these videos from 10FPS to 20FPS, view our demonstration at:

https://www.youtube.com/watch?v=8HtA6iyJkHo

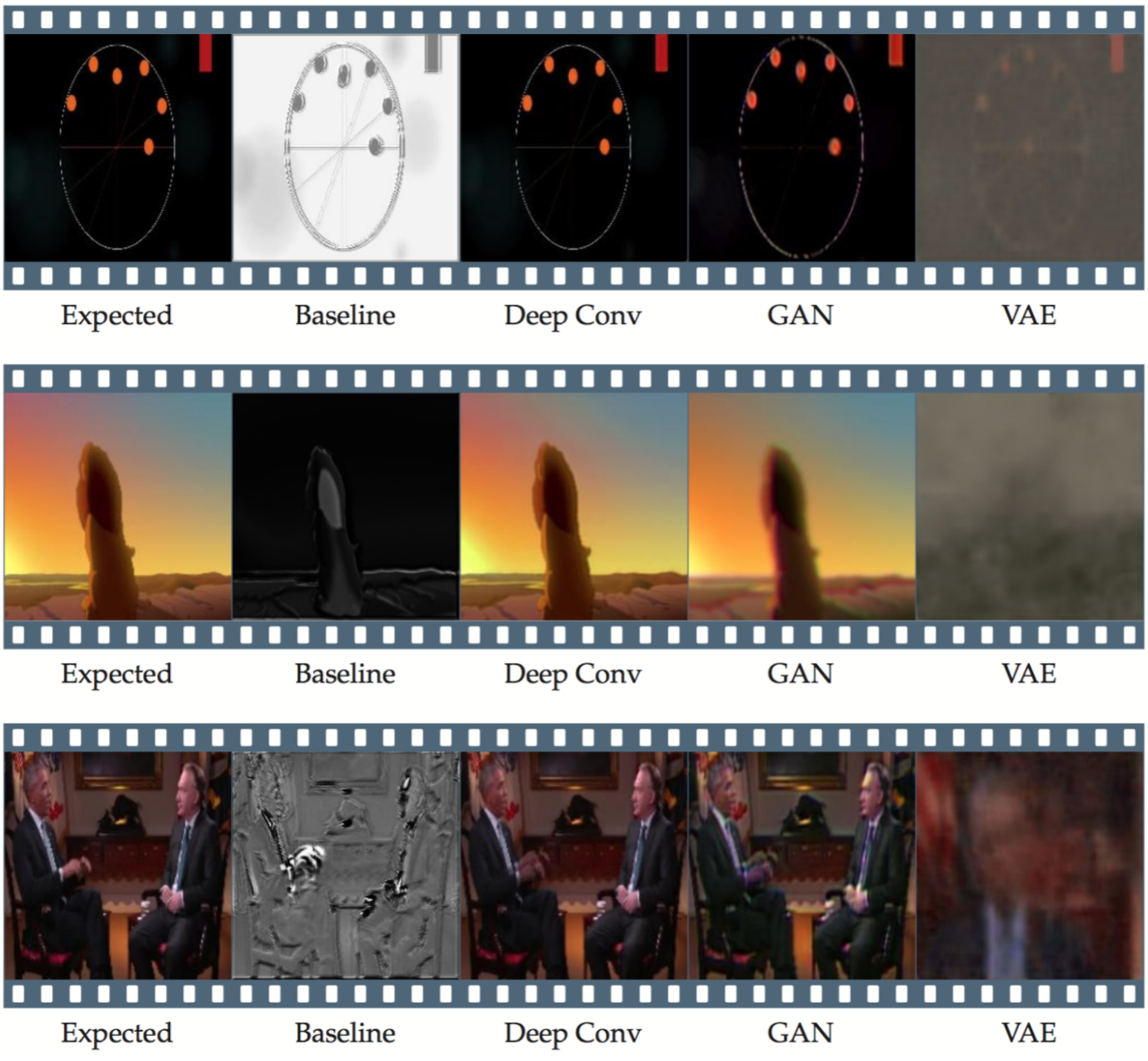
The results of the models can still be understood from single static frames, as shown on the next page. A general note is that the Baseline 3-Layer Convolution Model has grayscale preprocessing, as discussed in Section IV; thus, a perfect reconstruction of the expected RGB image is not possible, but general trends can still be observed.

We see that for the "Moving Circles" video (shown in the first row), the Baseline, Deep Convolution, and GAN models all generate an image similar to the expected frame. The Deep Convolution generated image is a little sharper than the GAN generated image, while the baseline model seems to have some kind of averaging and edge detection artifacts. The VAE generated image has a circle in the center, which may be a remnant of training, where a circle would often be in the center of the frame.

For the "Lion King" video (shown in the second row), the Baseline, Deep Conv, and GAN models generate an image roughly similar to the expected frame. Again, the Deep Conv image is the sharpest generated image. The generated GAN image is fairly blurry, both for the foreground and background of the image. The VAE generated image is extremely blurry and has a significant amount of uncertainty, shown visually as noise.

Finally, for the "Obama" video (shown in the third row), the Deep Conv and GAN models all generate images roughly similar to the expected image. However, the moving parts of the video (Obama's hands) are not correctly interpolated - we see some kind of averaging/overlay instead of a prediction of the intermediate hand position. These models do fair better than the VAE however, which generates a very downsampled zoomed-in image of Obama's face. This is because the VAE is trained on the corpus of the entire interview, during which many frames are of a zoomed in focus shot of Obama's face. The VAE seeks to reconstruct the video by simply memorizing various frames during the compression step, resulting in very inaccurate generated images on most frames.

Across the various test frames, we find that the Deep Convolution Model performs the best. The GAN performs well, but is blurrier than the Deep Convolution Model. We hypothesize that the GAN's Discriminator requires more tuning, as it did not successfully discriminate generated images from real images; if it had discriminated better, there would have been significantly less blur in the generated images. Another possible explanation for this behavior is that the downsampling (maxpool) component of the discriminator causes it to lose information about the generated image; thus, allowing blurry images to pass as real images since they have similar feature maps after the 2x2 maxpool layer. The baseline model is essentially performing grayscale linear interpolation, with some interesting learned features (such as learning filters for edge detection). We suspect that the baseline model performed worse than the Deep Convolution Model because there simply weren't enough layers; the network was deep enough for interesting filters to be learned, but it was not complex enough to fully learn the representation of a generated image.



IX. Tools

We used Keras to implement our neural networks, allowing us to including support for Theano and TensorFlow as a core framework for applying our models. We ran our experiments on a variety of hardware, but most significantly we utilized a cluster of 8 NVIDIA K-80s.

X. Lessons Learned

1) Make sure all data sets are reproduced on all machines - As we transitioned in between the multiple team members' machines we worked on we occasionally would find a similar, but incorrect YouTube video. This would lead to misleading MSE plots, making it harder to debug experiments.
2) Version control is crucial - we ran into some issues with not saving older models but thanks to version control, we were able to roll back on those problems.
3) Utilize coarse-grain search for hyper-parameters - test many different hyper-parameter configurations before committing to a few and training for a long time. Testing numerous hyper-parameters this way allows you to get a general sense of what works, and what doesn't, much sooner.

XI. Conclusion

The task attempted in this project was ambitious: perfect frame rate upscaling via frame interpolation is a task that has proven difficult to generalize to all types of video (such as animation), especially with traditionally available techniques of signal processing. Through the tools used, successes experienced, and failures experienced we believe that attaining a

general deep learning framework for frame rate upscaling is feasible.

Through the remaining areas of exploration (detailed in the following section) we intend to continue pursuing this project, so that one day their might be a powerful and effective frame rate upscaling tool that generalizes to the entire corpus of publicly available video.

XII. Future Work

In section V, Areas of Inquiry and Prior Work, we detail several axes of experimentation and several possible experiments. However, for the sake of this project we limited the scope of our inquiry for each of these axes of experimentation. Based on the performance of our models the two areas we believe are most crucial for improving performance and ensuring our models generalize better are adding residual connections to our VAEs, and utilizing MFSR to enhance the quality of our output images.

As we saw empirically in our results, the VAE resulted in either extremely blurry images or missed key features of the images. The issue of blurriness is due to the fact that each deconvolution block has upsampling, which causes the last layers to result in blurry images. Using residual connections between convolutional and deconvolution blocks will allow us to convey information from earlier layers in the net before the information was ever downsampled to the dimensions of the deconvolution layer in question. This should allow the VAE to generate crisp results.

Most of the related work we saw heavily used VAEs, so we were disappointed to see that it only generated results based on a few key features of the entire video set (eg. always generate Obama's face given any input frames with him in it). However, this problem is highly related to the problem of information loss in VAEs. Because of this we believe that residual connections will also help alleviate this issue in addition to the problem of blurry VAE-generated images.

The second area for improving performance is related to the flickering effect in our results video. In the video, one can notice a distinctive flickering effect in the upscaled video during the "Lion King" segment. This is due to the fact that the generated images all tend to have some tint or distortion compared to the real images, leading to the flickering effect observed between generated and real images. This type of problem is perfect for MFSR to tackle, as it will leverage information from adjacent frames in the input stream to to help correct things like hue and make them similar across the input and output images.

We hypothesize that the generated images are, as a distribution, considered to be fairly realistic when viewing them from a Generative-Adversarial lens. However, when compared with real images, there are still artifacts from the generation step that aren't captured through the loss functions or the Discriminator Network. We propose postprecessing steps to overcome this: we supply more temporal information to the discriminator network (to better penalize tinted images), consider KullbackLeibler divergence between generated images and real images, as well as attempt super-resolution methods seen in [6].

XIII. Team Contributions

All three members of the team contributed equally (33% each) but in different ways. We've highlighted what each person worked on below, but we did have lot of overlap. The amount of time spent on something did not always equal the amount contributed, as would be expected in a research environment.

The team shared the burden of researching previous works, writing the final paper, creating the presentations, drafting the poster, and presenting during the midpoint and final.

Gautham focused on implementing and experimenting with different models, solely using what Keras provides. He trained and tuned several of the models on his local machine. He produced many of the results found in the midpoint and the deep convolution model and GAN in the final presentation. He also created the demonstration video used in the poster session.

Ted focused on theory, fine tuning models, and synthesizing the generated videos and metrics from the models. Initially, Ted worked on extending prior work and the theoretical opportunities for improvement. Then, Ted worked on optimizing the various models to scale up training data on external clusters. He produced many of the results found in the final presentation, focusing on the baseline, deep convolution, and VAE models.

Raul focused on building upon Keras to make more complex models that the platform does not natively support. In addition, Raul worked on the preprocessing generator to extract training datasets from YouTube videos, a surprisingly difficult part of the project. His pipeline allowed us to do initial, though unsuccessful, experimentation with residual connections, pretrained networks, and LSTMS.

Acknowledgment

The authors would like to thank Professor John Canny and the staff of CS294 Designing, Visualizing and Understanding Deep Neural Networks for an incredible learning experience this semester.

References

[1] Gucan Long, Laurent Kneip, Jose M. Alvarez, and Hongdong Li. Learning Image Matching by Simply Watching Video. *Arxiv*, pages 1–18, 2016.

[2] Alex Greaves and Hanna Winter. Multi-Frame Video Super-Resolution Using Convolutional Neural Networks. 2016.

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. *arXiv:1501.00092v2*, pages 1–14, 2015.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses Supplementary. *Arxiv*, pages 1–5, 2016.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, and Mehdi Mirza. Generative Adversarial Networks. *arXiv preprint arXiv: ...*, pages 1–9, 2014.

[6] Daniel Glasner, Shai Bagon, and Michal Irani. Super-Resolution from a Single Image. page 9, 2009.