Image Enhancement Applied to Dynamic Frame Generation

Mark Wesley Harris
University of Colorado Colorado Springs
wharris2@uccs.edu

Jugal Kalita
University of Colorado Colorado Springs
jkalita@uccs.edu

Abstract

Image enhancement is a well-studied problem in Computer Vision. Since performing detailed enhancements analytically produces poor results, researchers are now turning to machine learning to solve this and other complex image processing problems. Here we attempt to discern the best suited architecture for enhancing images. The focus is not only on upscaling, but also on removing noise and other visual artifacts from an input image. We are thus interested in how networks can learn to remove previously unseen anamolies from images while retaining clarity. Two generative models were investigated, Generative Adversarial Networks and Transformers. Our evaluation showed that the former improved images with 1.3 times more accuracy on average.

Introduction

Image enhancement is the process of improving an image's quality, resolution, and clarity. From its many applications in fields such as surveilence, cinematography, and social media, researchers are currently interested in finding new ways of improving image enhancement techniques. Leading research and development companies like Google and Facebook are keen on creating the newest state-of-the-art image enhancement systems (Dahl, Norouzi, and Shlens 2017). Image enhancement involves not only increasing image resolution (called super-resolution), but also replacing noise and other anomalies with believable substitutes. Here we discuss our evaluations of the DCGAN, SRGAN, and Sparse Attention Transformer architectures applied to image enhancement.

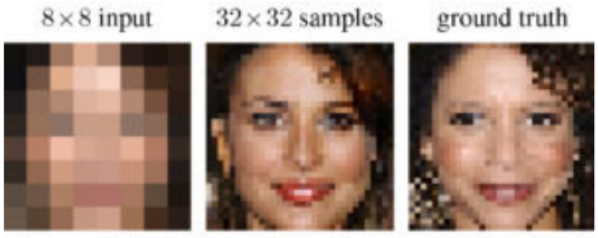


Figure 1: Results of Google's super-resolution network (Dahl, Norouzi, and Shlens 2017).

Related Work

Our research focuses on two generative models, Generative Adversarial Networks (GANs) and Transformers. Both architectures represent broad classes of networks. Concerning the GAN framework, we look at the DCGAN (Goodfellow et al. 2014) and SRGAN (Ledig et al. 2017) models. The Transformer architecture evaluated was the Sparse Attention Transformer (Child et al. 2019), which is very new research that is still under development. Background information on each network is discussed below.

Generative Adversarial Network

The Generative Adversarial Network (GAN) was first proposed as a way to train a model to produce more realistic images from noise. The vanilla GAN model works as essentially a random image generator. The network is made up of two sub-architectures: a generator, G, and a discriminator, D. GANs are able to generate new images that have similar qualities to those in the dataset on which they are trained (Goodfellow et al. 2014). A generalization of the GAN architecture is shown in Figure 2.

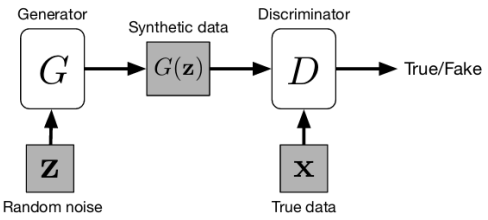


Figure 2: Basic Generative Adversarial Network architecture, with a generator G and discriminator D (Huang, Yu, and Wang 2018).

The model works by training both the generator and discriminator in tandem. G is trained to progressively generate more realistic images, while D is trained to recognize differences between real and fake inputs (Huang, Yu, and Wang 2018). This relationship can be expressed in the format of a "two-player min-max game", shown in Equation 1.

$$\min_G \max_D V(D,G) = E_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + E_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \quad (1)$$

$p_{data}(\mathbf{x})$ represents the true data distribution, and $p_z(\mathbf{z})$ represents the distribution of noise. Optimizations and variants are based off this simple concept, of which include the cGAN, DCGAN, and SRGAN architectures.

Transformer

Transformers were first proposed as a way to use attention mechanisms more efficiently. Attention is a technique that references past data during each iteration of training. The Transformer relies "...entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution" (Vaswani et al. 2017). The Encoder and Decoder for the Transformer design are shown in Figure 3. An input is first encoded into the dimensional space the transformer works with, and the result is then decoded back as the output of the model.

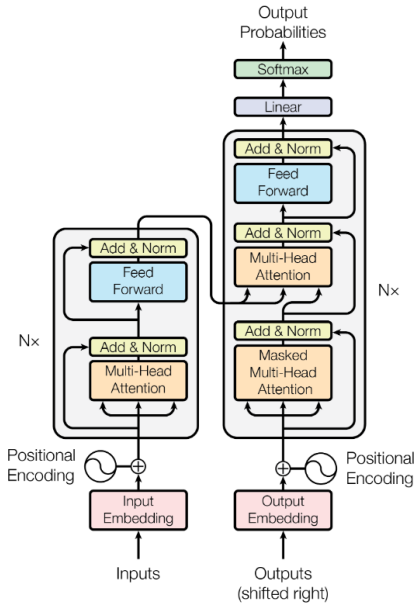


Figure 3: Transformer architecture (Vaswani et al. 2017).

The Transformer uses a series of attention functions to map between two sequences. An attention function is a "...mapping [of] a query and a set of key-value pairs to an output" (Vaswani et al. 2017). The mathematical expression of the attention function is shown in Equation 2. d_k is the dimension of keys and queries, and Q, K, V are matrices of queries, keys, and values, respectively. The function uses the dotproduct operator, so that the output is computed as a linear combination of weights (Vaswani et al. 2017).

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

We can model the joint probability of a sequence, $\mathbf{x} = x_1, x_2, \ldots, x_n$, as the product of conditional probability distributions parameterized by a network θ (Child et al. 2019). The final expression is shown in Equation 3.

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}; \theta) \quad (3)$$

Transformers are able to "...model arbitrary dependencies in a constant number of layers," and are useful for natural language processing and image generation (Child et al. 2019). While the Transformer shows potential as a powerful machine learning technique, it is a recent concept and still has many inherent problems. One of the major problems with the architecture is that its required resources scales with $O(n^2)$ for sequence length n. Researchers theorize that "...to improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r" (Vaswani et al. 2017).

The Sparse Transformer architecture was developed as a means to shrink the computational resources for large sequences of data. Child et al. introduced sparse factorizations on the attention matrix in order to speed up processing. They approximated the dense attention operation by combining several cheaper attention operations. This new method resulted in a faster attention-based architecture that could be trained on longer sequences of data (Child et al. 2019).

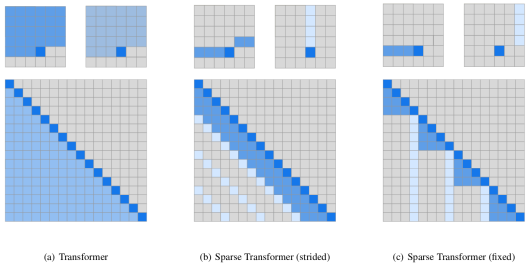


Figure 4: Optimizations of Attention (Child et al. 2019).

Figure 4 shows a visual representation of the two optimizations experimented with, strided and fixed. The Sparse Attention Transformer architecture reduces the resource cost to $O(n\sqrt[p]{n})$. The architecture is also simpler than other autoregressive models that perform similar functions, including upscaling and enhancement (Menick and Kalchbrenner 2019).

Approach

We now discuss our considerations for the GAN and Transformer implementations. The first approach studied was the Deep Convolutional GAN (DCGAN). The DCGAN is similar to a regular GAN, but uses convolutional and convolutional-transpose layers in D and G, respectively (Radford, Metz, and Chintala 2016). Pose Guided Person Image Generation (PG^2) proved that a variant conditional DCGAN could be used to remove anomalies and provide

higher resolution for output images (Ma et al. 2017). Without specifics on how the model was altered for image enhancement, it was difficult to obtain good results from our initial implementation.

Instead of fine-tuning our imperfect DCGAN, we implemented a Super Resolution GAN (SRGAN). The SRGAN was first developed in Single Image Super Resolution (SISR) research, and showed a drastic decrease in loss measurements compared to "NN, bicubic interpolation, and four state-of-the-art methods" (Ledig et al. 2017). the model architecture of the SRGAN was directly applicable to the image enhancement problem, and did not suffer the same set-backs we encountered with the DCGAN. We retain the presumption that the SRGAN architecture could be altered into a better performing DCGAN, however in order to fully test all architectures we decided to only focus on our initial SRGAN implementation.

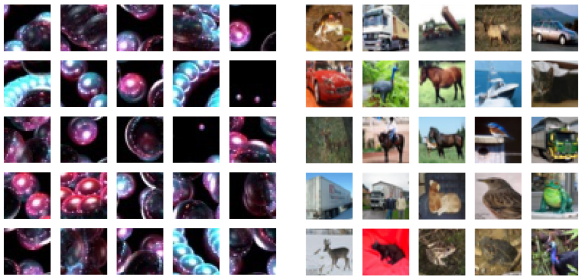
The last architecture we studied was the Sparse Attention Transformer. Source code was taken from the Sparse Attention project (Child et al. 2019). The architecture used completely different mechanisims than the GAN models, requiring a separate runtime environment, alternate Python libraries, and a Linux operating system. After getting the initial Docker environment set up on a fresh Linux build, we focused on correctly formulating the tensor inputs of the attention function. The implementation utilized multi-headed attention with a minimum of 4 layers, which gave us the extra challenge of ensuring our inputs were split appropriately for the network.

As an overview of each architecture, The DCGAN and SRGAN implementations were written for PyTorch, while the Transformer architecture utilized TensorFlow. A python module of utility functions was created and shared between the programs via GitHub, so that evaluations of altered images would be consistent and comparable for all architectures. A dataset of images was generated for the purposes of training and testing the developed networks. The standardized CIFAR-10 dataset was also used to further support our analysis. Each dataset contained approximately 60,000 32 x 32 pixel images, random samples of which are shown in Figure 5.

Datasets

The first dataset we used was generated from 300 frames of source animation (Harris 2019). We refer to the extracted images as "frameblocks" – each frameblock was chosen to represent significant change between two source frames. Frames were processed sequentially, which resulted in a total of 56,337 generated images. The CIFAR-10 dataset was also used, in order to help evaluate how our models compare to current research. The CIFAR-10 data consists of 10 different categories of images, with 60,000 images total. Examples of the categories include dogs, horses, frogs, and automobiles.

During training and evaluation, input images from each dataset were pre-processed in order to produce low resolution versions containing controlled amounts of random anomalies and noise. Equation 4 expresses how images were altered. First, a random sample of uniform noise, I_N, was



(a) Generated dataset. (b) CIFAR-10 dataset.

Figure 5: Random samples from each dataset evaluated.

generated in the shape of sample image, I. A linear combination was applied to add α amount of the input image with $1 - \alpha$ amount of the random noise (where $0 \leq \alpha \leq 1$). Afterwards, a Gaussian blur was applied with stride β. We decided to evaluate several different senarios of varying image quality. The top row of each sample in Figure 8 shows altered samples of an image from the CIFAR-10 dataset with varying parameters of α and β. The OpenCV Python library was used to perform the same alterations accross all model architectures, so we know the results are consistent and repeatable.

$$I' = G_{\beta,\beta}(\alpha I + (1 - \alpha)I_N) \tag{4}$$

Evaluation

A major criticism of GANs and other generative models is the "lack of a robust and consistent evaluation method..." (Richardson and Weiss 2018). As opposed to other machine learning models, GANs do not optimize any kind of objective function, and operate instead on a learned latent space which cannot be analyzed analytically. Thus, in order to evaluate the implemented models, we must find a way to numerically compare the generated images to their respective targets.

Researchers have defined many different means of comparing two images. Some of this research is focused on image context, such as if two images contain the same person or setting. This type of analysis would not benefit our project, since the image needs to be exact – simply containing similar features is not enough. Thus, we turned to the measurements of Mean Squared Error (MSE) and Peak Signal-To-Noise Ratio (PSNR), as both matrics are commonly used to evaluate super resolution algorithms (Yang et al. 2010). However, since MSE has been found to overlook high-frequency details, and PSNR involves complicated calculations, we decided against using these methods as a means for evaluation at this time (Ledig et al. 2017).

In place of MSE and PSNR, we chose to use the concept of pixel distance to compare two images. Pixel distance is often referred to as L^2 distance. The expression for L^2 between two images, I_1 and I_2, is shown in Equation 5 (Foley et al. 2013).

$$\sum_{i=1}^{w \times h} ||\mathbf{P_i}^{I_2} - \mathbf{P_i}^{I_1}|| \tag{5}$$

Pixels are treated as vectors, and images as 3-dimensional matrices. $\mathbf{P} = \langle r, g, b \rangle$ represents the red, green, and blue color values for pixel P, respectively. Note that the distance between two pixels $\mathbf{P}$ and $\mathbf{Q}$ is defined in the following expression:

$$||\mathbf{P} - \mathbf{Q}|| = \sqrt{(P_r - Q_r)^2 + (P_g - Q_g)^2 + (P_b - Q_b)^2}$$

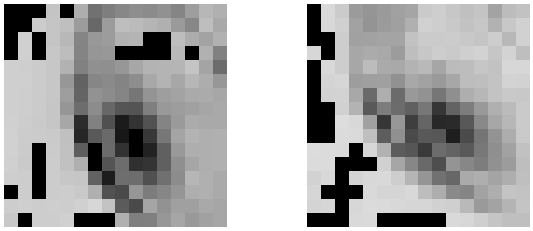
Each of the 60,000 images in either dataset have resolutions of 32 pixels in width (w) and 32 pixels in height (h). Thus, processing a single image pair would consist of summing 1,024 distance calculations. We found this to be very draining on computational resources – the square-root operation in particular was problematic. In order to save time training and evaluating, we chose to replace traditional L^2 distance with a simplification that uses bit-wise XOR. Since the XOR operation accurately captures differences between color values without costing much in processing, we found it to be a credible and efficient replacement of L^2 distance (Serra 1983). For a visual comparison of standard L^2 to our implementation, see Figure 6. The modified distance calculation we used for our evaluation is shown in Equation 6.

$$\sum_{i=1}^{w \times h} \lceil (\sum_{x \in r,g,b} \mathbf{P_i}(x)^{I_2} \oplus \mathbf{P_i}(x)^{I_1}) \rceil^{255} \tag{6}$$

The sum of color values ($\sum_{x \in r,g,b}$) is stored in the corresponding 8-bit pixel of a black and white image. Values are capped at 255 to avoid erroneous results with overflow in evaluation. Percentage of enhancement (e.g. likeness of an image to the target) is calculated using a ratio of the pixel sum S_p and that of a completely black image, $T_p = 255 \times w \times h$.

$$E = 1 - \frac{S_p}{T_p} \tag{7}$$

If the value of image enhancement is close to 1, then there is very little difference between the generated image and the target. Conversely, a value close to 0 implies great difference between the two images. Visually speaking, a darker image signifies greater difference than a lighter image.



(a) Example of standard L^2 calculation.

(b) Example of our distance calculation.

Figure 6: Visual comparison of pixel distance metrics.

Challenges

Since the base DCGAN architecture was created for image generation from noise, we found no feasible way to inject a conditioning image during training. Several methods were attempted to fix this problem. We first looked at manipulating the Z-latent vector. This is the vector trained by the GAN to map features from one image to another. Since the latent vector is unknown until training occurs, it was impractical to initialize it without a way to convert the condition image into latent space. After several attempts, this method was abandoned.

The next approach proved to be more useful than manipulating the latent vector directly. The DCGAN was made to generate $G(\mathbf{x})$ of the expression $I_E = X - G(\mathbf{x})$. The theory behind this approach was to train the model to generate the enhancement by inversing the alterations made in pre-processing. We found that with the way the generator is implemented, however, back-propagation acts on the image generated and not the calculated image, I_E. Thus, the generator eventually became confused and quickly spiraled out of control thereafter. When comparing the DCGAN results to those of the SRGAN, we decided the DCGAN implementation did not provide enough significance to support evaluation. However, the model did produce good images from noise. Examples of generated images are shown in Figure 11.

Results

The SRGAN and Transformer models were trained on 75% of the dataset, with shuffling, for each combination of parameters (α and β). The remaining 25% of images were used for testing. Figure 7 shows the calculated enhancement during training. The average enhancement for each distribution is drawn as a horizontal dashed line. Green and blue colored lines denote measurements for the CIFAR dataset, while yellow and red lines are measurements for our generated frame-block dataset.

Results of our testing data are presented in Table 1. We found that the SRGAN showed much better trends than the Transformer. As shown in the graphs, the SRGAN model increased in accuracy over the course of training. The Transformer's performance was unexpectedly erratic, which explains why some images were better learned than others. The SRGAN model handled each dataset very differently, shown by the pronounced distinction between to the two distributions. By comparision, the mean accuracies of the Transformer are much closer together.

As a way of evaluating each model's capabilities for images with unknown amounts of noise and blurring, we also trained them on images with random combinations of noise and blurring. We were surprised to find that these results differed only slightly from those of the least-altered set of images. This proves that generative models are appropritate for solving real-world enhancement problems, where images are not usually affected by anamolous data in the same way or amount. Overall, our generated dataset performed much better than the CIFAR-10 dataset. This is what we anticipated, since all images in our dataset had the same qualities.

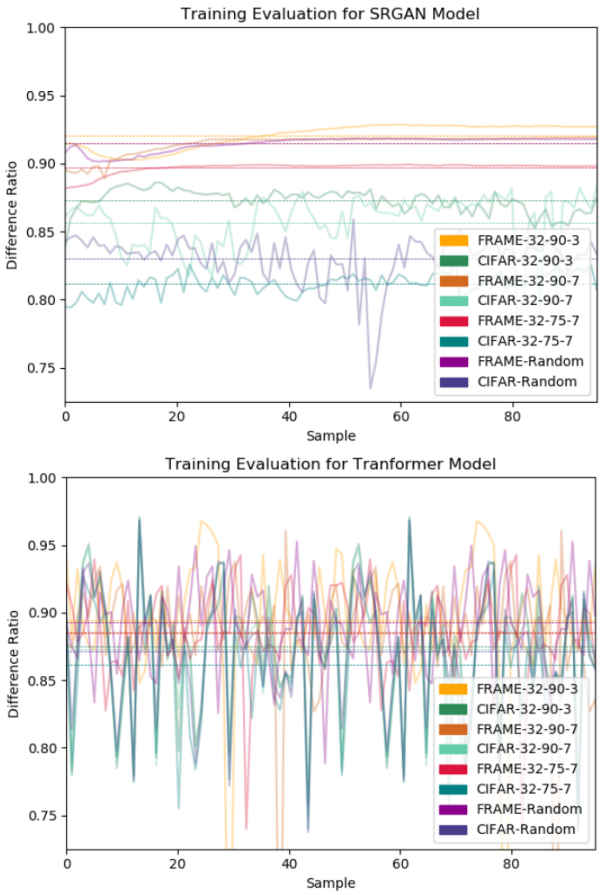


Figure 7: Results during training of the SRGAN and Transformer models.

Subject, lighting, color, and negative space were found to influence the accuracy of image enhancement.

The averages captured in Table 1 demonstrate the disparity between training and testing performance. Both the SRGAN and Transformer suffered a dip in accuracy between training and testing, however the Transformer's accuracy dropped around 4 times more than the SRGAN's. This result again shows that our Transformer model was over-fitting to some data. The average SRGAN accuracy on testing data was 0.84291, while the Transformer had an average accuracy of 0.62945. Thus the SRGAN was on average 1.3 times as accurate as the Transformer architecture.

We found that noise more heavily impacted results than blur for either model. This effect is demonstrated by the samples shown in Figure 8a, where the most distorted image is not nearly as close to the target image as the enhancements from less noise. By comparison, the Transformer outputs shown in Figure 8b are much lower quality, but demonstrate the same trend. We found the poor performance was due to the high variance of the Transformer's accuracy. The Transformer had a clear trend of over-fitting to certain types of images. Some images were nearly perfect, while others were highly inaccurate. The image we chose as the sample

Training Evaluation				
(α, β)	**(0.75, 7)**	**(0.9, 7)**	**(0.9, 3)**	**Range**
Frame Training Dataset				
SRGAN	0.89687	0.91459	0.92046	0.91450
Transformer	0.86831	0.88479	0.89707	0.89409
CIFAR-10 Training Dataset				
SRGAN	0.81159	0.85660	0.87253	0.83036
Transformer	0.87515	0.87191	0.86196	0.86804
Training Averages				
SRGAN	0.85423	0.88559	0.89649	0.87243
Transformer	0.87173	0.87835	0.87951	0.88106

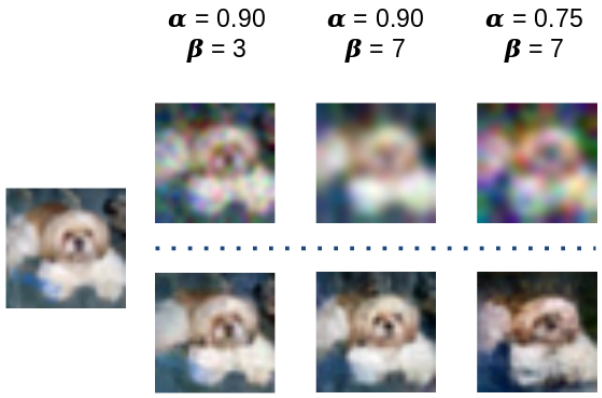
Testing Evaluation				
(α, β)	**(0.75, 7)**	**(0.9, 7)**	**(0.9, 3)**	**Range**
Frame Testing Dataset				
SRGAN	0.84983	0.87031	0.87105	0.85876
Transformer	0.65538	0.66616	0.67249	0.66494
CIFAR-10 Testing Dataset				
SRGAN	0.78727	0.83580	0.85327	0.81697
Transformer	0.59138	0.59293	0.59749	0.59504
Testing Averages				
SRGAN	0.81855	0.85305	0.86216	0.83786
Transformer	0.62327	0.62960	0.63504	0.62989

Table 1: Evaluation for the SRGAN and Transformer models on each set of parameters.

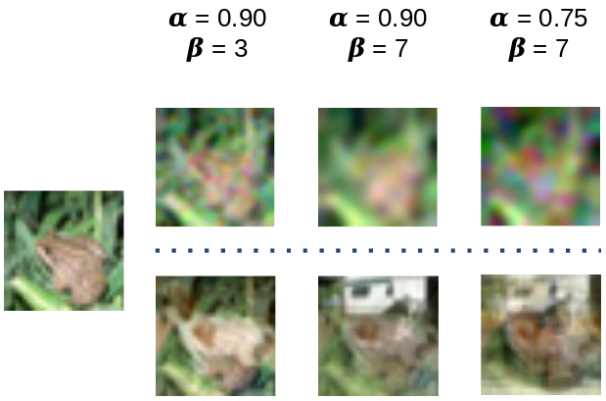
in Figure 8b was of median quaility. Example outputs for the DCGAN, SRGAN, and Transformer models are shown in Figures 9, 10, and 11. The XOR comparison image for each input pair is shown where applicable, comparing the architecture outputs (left) to the real source images (right).

Conclusion

Image enhancement is a difficult problem to solve. With the advent of machine learning, however, researchers have been able to produce more realistic and detailed enhancements from very crude images. Here we discussed our work regarding the evaluation of several machine learning architectures. Our analysis is not exhaustive, and there are still plenty more models we will test in the future as applicable to this problem. The most promising of these networks include the Convolutional Neural Network (CNN) and Conditional GAN (cGAN).



(a) SRGAN results on input.



(b) Transformer results on input.

Figure 8: Example outputs of a single input image for each combination of parameters.

Out of the models implemented, the SRGAN showed the best performance accross the board. While the DCGAN and Transformer models faired poorly in our analysis, we note that with some further development they could conceivably obtain results similar to those acheived by the SRGAN. The added capabilities of Sparse Attention Transformers in particular is promising for enhancement and many other image processing problems. We look forward to learning of the exciting discoveries that will be made in Computer Vision through the research and application of GANs, Transformers, and other generative models.

References

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. *CoRR* abs/1904.10509.

Dahl, R.; Norouzi, M.; and Shlens, J. 2017. Pixel recursive super resolution. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 5449–5458.

Foley, J. D.; van Dam, A.; Feiner, S. K.; and Hughes, J. F. 2013. *Computer Graphics: Principles and Practice (3rd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, 2672–2680. Cambridge, MA, USA: MIT Press.

Harris, M. W. 2019. Cs5800 demo 1. https://www.youtube.com/watch?v=UmGhuq_Zpr4.

Hong, Y.; Hwang, U.; Yoo, J.; and Yoon, S. 2019. How generative adversarial networks and their variants work: An overview. *ACM Comput. Surv.* 52(1):10:1–10:43.

Huang, H.; Yu, P. S.; and Wang, C. 2018. An introduction to image synthesis with generative adversarial nets. *ArXiv* abs/1803.04469.

Ledig, C.; Theis, L.; Huszar, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; and Shi, W. 2017. Photo-realistic single image super-resolution using a generative adversarial network. 105–114.

Ma, L.; Jia, X.; Sun, Q.; Schiele, B.; Tuytelaars, T.; and Van Gool, L. 2017. Pose guided person image generation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 405–415. USA: Curran Associates Inc.

Menick, J., and Kalchbrenner, N. 2019. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*.

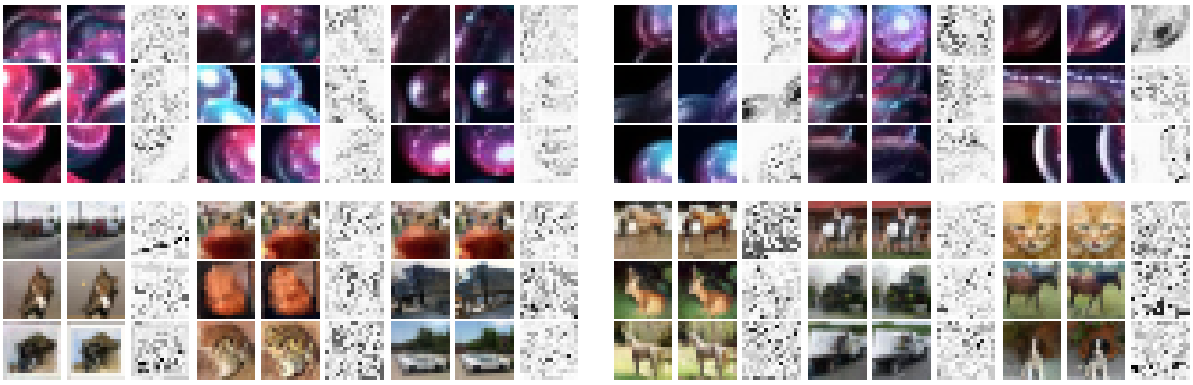
Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*.

Richardson, E., and Weiss, Y. 2018. On gans and gmms. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, 5852–5863. USA: Curran Associates Inc.

Serra, J. 1983. *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc.

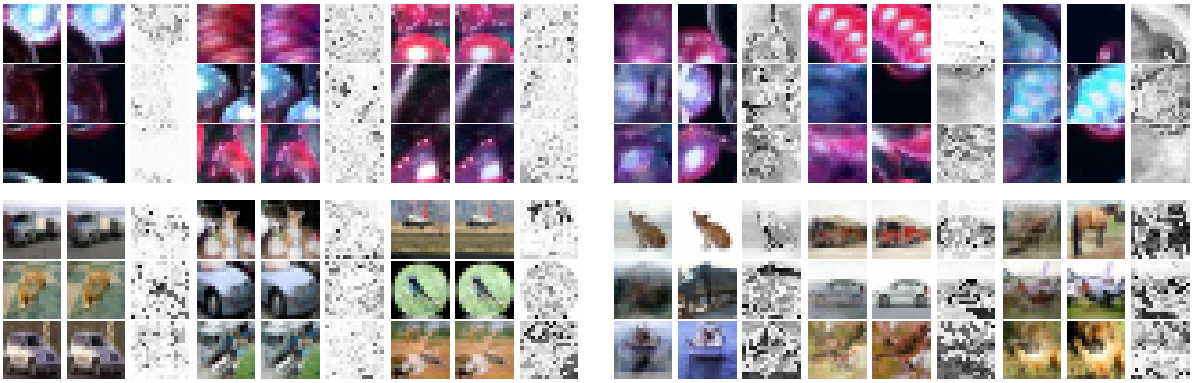
Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Yang, J.; Wright, J.; Huang, T. S.; and Ma, Y. 2010. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing* 19(11):2861–2873.



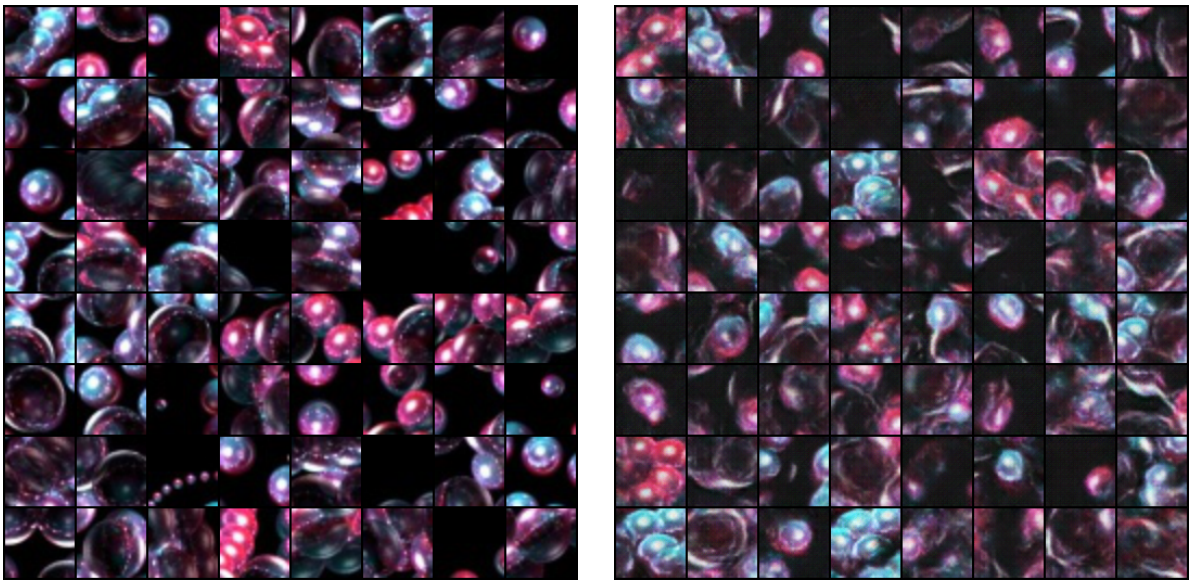
(a) SRGAN training results. (b) SRGAN testing results.

Figure 9: Super Resolution GAN evaluation.



(a) Transformer training results. (b) Transformer testing results.

Figure 10: Generative Sparse Transformer evaluation.



(a) DCGAN real samples. (b) DCGAN generated samples.

Figure 11: Deep Convolutional GAN evaluation.