Transformer Based Pix2Pix

Alex Albors Juez
University of Washington
aalbors
aalbors@uw.edu

Logan Garwood
University of Washington
lgarwo
lgarwo@uw.edu

Abstract

Generative Adversarial Networks (GANs) have become one of the best model frameworks in deep learning for the task of image generation. They have been shown to learn to generate life-like results in diverse creative domains. Pix2Pix is an image generation variant using GANs in which the model translates the style of an input image. A traditional Pix2Pix GAN usually consists of two convolutional neural networks, the generator and discriminator, playing a game in which they try to beat each other. The generator's job is to generate a new image conditioned on its input image - typically done through the classic U-Net architecture. However, since the advent of the transformer, the deep learning community has found attention to be "all you need". In this paper we introduce a new generator architecture that incorporates the use of transformers instead of a traditional U-Net. The main idea being that transformers excel in translating language sequences, so perhaps they will be well suited for Pix2Pix image translation. We iterate on our baseline to explore the relationship between network depth and residual connections on the success of the GAN. We compare these model designs to a convolutional baseline to measure success using a variety of metrics. A success by transformer GANs in Pix2Pix could reinvigorate the lately diminishing appeal of generative adversarial networks in image generation due to the advent of diffusion models. Additionally, we believe that the results of our model could suggest an opportunity to modernize other deep learning models with the use of transformers.

1. Introduction

Generative adversarial networks (GANs) have been used to generate authentic looking images across a variety of domains. They take advantage of an adversarial game between two agents, a discriminator D and a generator G (in practice these are Neural Networks). The generator produces synthetic images that are meant to simulate a real dataset while the discriminator tries to determine if the produced image is real or created. Through this game, both agents improve until eventually the generator produces images that are so believable, the discriminator cannot tell the difference between the dataset and the output. Due to the GAN's ability to learn a loss function that adapts to the data, they have been successful in a multitude of tasks that traditionally would require different kinds of loss functions. One of these has been image-to-image translation, which seeks to translate an input image from one domain to another domain given input-output image pairs as training data. Tasks within this domain include image colorization, image inpainting, in which we complete missing parts of a given image, and image segmentation, which annotates individual pixels as belonging to a specific class or instance. With regards to image segmentation, given an input image, GANs output segmentation masks representing pixel-by-pixel boundaries and shapes of each class, which correspond to different objects or regions in the input image. We present Pix2Trans2Pix2GANs, an improved version of the popular Pix2Pix GAN used for paired Image-to-Image translation tasks [5]. We introduce our new architecture approach to Pix2Pix GANs which replace the classic U-net convolutional encoder-decoder with a modified, transformer based generator instead. Transformers have swept the world of deep learning and are being implemented in a variety of mediums and model types. We are hoping that a transformer based generator will be an upgrade from a convolutional generator and will produce higher quality image translations.

2. Related Work

The field of image to image generation has evolved rapidly in the past years due to the advent of GANs and their ability to implement different losses through their discriminators, which were first introduced by Goodfellow et al [3]. Although they showed a remarkable ability to generate high-quality images, they were not yet ready to tackle image translation due to their unconditional nature: unpaired training data and the nature of the GAN architecture itself did not allow for the generator to learn to produce an image given a certain condition. In the case of image segmentation, this would correspond to the real life image we'd wish to segment. This issue was quickly resolved with the introduction of conditional GANs, proposed by Mirza et al [7] and quickly adopted in the image translation community by Isola et al's Pix2Pix GAN [5], which accomplished remarkable results by introducing a U-Net with residual connections as their generator architecture and focusing on looking at local, high-frequency structures in their discriminator. Shortly after, Wang et al [11] introduced upgrades to Pix2Pix through their model Pix2PixHD by including an additional generator, letting one focus on the generation of coarse features, and another on the fine details. Furthermore, due to the improved generator, they also augmented the discriminator's receptive field by adding two other discriminators, each focusing on different image scales. Such architecture augmentations however, considerably increase the model complexity and training difficulty. Since the introduction of Transformers by Vaswani et al [10], attention layers have become ubiquitous in the deep learning community. They have been successfully implemented in Vanilla unconditional GANs [6], where the authors take a pure transformer-based approach completely free of convolutional layers. They also introduced multiple discriminators to simultaneously capture semantic contexts and low-level textures. Transformer GANs have also been explored for high-resolution image generation tasks, such as [12], but their computationally expensive training remains a nuisance.

3. Methods

3.1. Pix2Pix GAN

Pix2Pix uses conditonal generative adversarial networks which is a variant of the original generative adversarial network. Conditional GANS learn a mapping from some observed image x to a generated image. The goal of the generator is to fool the discriminator by producing output images that appear real. The discriminator has access to a real image x and a real or fake image and learns to produce the probability that it is real given x. Therefore if (x, y) is a pair of training data representing a real input and target image, then the objective function of a conditional GAN can be expressed as

$$V(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log(1 - D(x, G(x))]$$

where G tries to minimize this function and D tries to maximize it [5]. Observe that the first term is large when the discriminator correctly predicts that a real image is real and the second term is large when the discriminator correctly identifies a generated image as fake.

In general, the parameters of a generator and discriminator can be arbitrary but in practice they are most often deep convolutional neural networks. In such a convolutional framework, the generator downsamples the input image through a series of convolutions into latent space and then applies transpose convolutions out of latent space back to pixel space, learning along the way the desired Pix2Pix translation. This downsampling and upsampling architecture is referred to as U-Net and is a common encoder decoder method. Then, the discriminator convolves the real input and real or fake output down and returns the probability that the second image is from the dataset as opposed to fabricated by the generator.

We follow the same overall training workflow as Persson [9] but make our own transformer-based generator instead of the U-Net. In subsequent sections we will explain the architecture in more depth.

3.2. Dataset

After surveying a variety of datasets, we decide to train our models on Cityscapes [2], a large-scale benchmark containing over 5000 street scene images with fine pixel-level annotations on over 50 different cities and a total of 30 different classes. After thorough training on datasets for different image translation tasks, we've found the task of image segmentation on Cityscapes to be the most appealing for our purposes: the target image simplicity allows the generators to remain competitive throughout training. For comparison, we tested our models for image inpainting on the Animal Faces Dataset [1] but saw the discriminator outpower the generator, and eventually leading it to mode collapse. We address this challenge in the discussion section.

3.2.1 Data Augmentation Techniques

To enrich the variety of the dataset, we employ a variety of image augmentations. We first ensure color chanels are represented in the $[0, 1]$ range. With regards to the input images, we perform horizontal flips with probability 0.5, and apply Color Jittering with probability 0.2. We also normalize across all color channels so that images have pixel mean $(0.5, 0.5, 0.5)$ and standard deviation $(0.5, 0.5, 0.5)$ as well (one coordinate per channel). We employ Python's Albumentations library to do so.

3.3. Baseline

Our baseline GAN generator came from the original Pix2Pix paper and was implemented by Aladdin Persson on GitHub [9]. In fact, we use parts of Persson's general GAN pipeline throughout the project. Their GitHub repository came complete with code that saved us a lot of time by implementing methods that were not at the focus of the project but are still necessary for success such as configuring the GAN settings, processing the dataset, and saving results.

The baseline generator is a deep convolutional neural network that follows the traditional U-net architecture as seen in Figure 2. It consists of six layers of convolutions, Batch norm, ReLU, and dropout. Then a bottleneck convolution with just ReLU. From the bottleneck they up-sample via 7 layers of transpose convolutions and again using Batch norm ReLU and dropout on the way up. Finally, the last transpose convolution uses three channels to transition into RGB space and then a hyperbolic tangent activation function in the end.

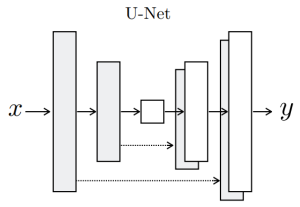

Figure 2. The standard U-Net encoder-decoder with long range residual connections.

The baseline model also takes advantage of residual connections between encoding convolutional layers and decoding transpositional layers. We will discuss the importance of these residual connections more in Section 3.5.

This model performed admirably out of the box and we will use the results from this model to test our hypothesis that a transformer based generator can outperform the traditional U-net convolutional generator.

3.4. New Architecture 1

Our first transformer based architecture operates within the same GAN training workflow as implemented by Persson although we created our own generator from scratch. Generally speaking, this generator replaces the convolutional encoding layers of the U-net with transformers instead [Figure 1]. The architecture takes some inspiration from Jiang et al [6].

First, the generator partitions the 256×256 images into 1024 patches, each of size 8×8 pixels [4]. Then the patches are encoded into a sequence in two ways. Each patch is projected out of pixel patch space by a fully connected layer from $\mathbb{R}^{192}$ into $\mathbb{R}^{64}$ and the result is added to the patch's positional embedding.

The encoded patches are then sent through four transformer blocks each consisting of a multi-headed attention layer, dropout, layer norm, a two layer perceptron $\mathbb{R}^{64} \to \mathbb{R}^{32} \to \mathbb{R}^{64}$ and then another dropout and layer norm. The transformers use eight attention heads and a dropout rate of 0.1

From this latent space we reshape the transformed sequence back into the patchified shape of $(N, 1024, 8, 8)$, where N is the batch size, in order to prepare to construct an image from the transformed information. We then upsample back to a tensor of the same shape as the input which results in the generated image. The upsampling is done by five copies of transpose convolution, batchnorm, and leaky ReLU. Each transpose convolution uses a kernel size of 5 a stride of 2, zero-padding 2, and an output padding of 1. The number of output channels of a single transpose convolution is set to half of the number of channels from the prior block. All of this is to ensure that the depth of the encoded tensor is halved with each convolution while the height and width are doubled until we reach a size of $(N, 32, 256, 256)$. Then a final linear layer is applied to project down to three channels representing the RGB values of the generated image.

In addition, similar to most Pix2Pix GANs we utilize residual connections during upsampling. However, the residual connections of this first architecture only establishes connections between directly adjacent transpose convolutional blocks during decoding. Note that the original Pix2Pix paper [5] uses long skip connections in their U-Net to allow information to bypass the encoder-decoder bottleneck. This first architecture does not do that because we have removed the convolutional encoding layers in place of transformer blocks, meaning the size of our encoded images does not approach a "bottleneck" in the typical sense of the

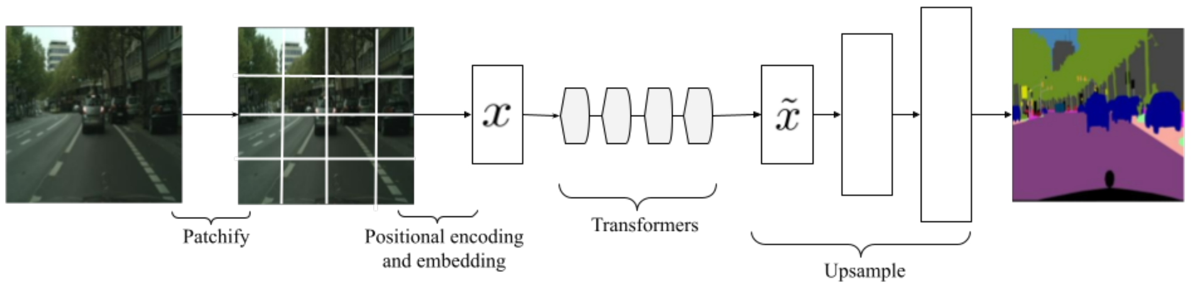

Figure 1. First architecture

word. Our other architectures will explore workarounds to this situation.

3.5. New Architecture 2

As Isola et al. [5] mention in the original Pix2Pix paper, image translation is a task in which residual connections across a U-Net makes sound sense. This is because we want the generated image to share a lot of broad structural similarities to the input image. The underlying hypothesis is that if the generator decoder can reference information from the encoding process, then the output images will be more similar to the input image. This is why they implement their generator with long range skip connections which transmits information from encoding convolutions across the bottleneck to the decoding transpose convolutions on the other side of the U-Net.

The problem with implementing this in our transformer based GAN is that the encoder-decoder does not follow a U-Net shape and does not even taper to a bottleneck in the same way that traditional U-Nets do. Recall that in the first transformer architecture, we patchify the image and embed the patches into a sequence which are fed through transformer layers. However the dimensions of the embedded patches do not change while being transformed. Instead, we take the transformed tensor and we reshape it so that each transformed patch is stacked to create a tensor of shape $(N, 1024, 8, 8)$. Note that we manually reshape instead of convolving down to this shape. So where do the residual connections come from? We chose to feed data from the first several transformer layers' output into the decoding transposition convolutional layers [Figure 3].

To do this, the input of the first five transformers are fed into the five transpose convolution layers during image construction. The rationale is identical to that of the U-Net although now instead of a connection between symmetric convolutional layers and transpose convolutional layers as in the U-Net, we have an asymmetric connection between transformer layers and the transpose convolutional layers.

We also chose to deepen the network by adding more transformer layers to increase the effect of our architecture change compared to the baseline. Due to the residual connection layout described above, we must have at least five transformer layers; we chose to use ten.

3.6. New Architecture 3

While we found that the preliminary results of architecture 2 signified improvements over architecture 1, it still fell short of the baseline implemented by Persson. To try and match this performance, we decided to construct architecture 3.

While transformer GANs on sequences of image patches have shown to perform well in the past for vanilla GANs [6], we wondered what aspects of our new application to Pix2Pix conditional GANs would translate poorly and thus need modification. In architecture 1 and 2 we adopt a similar patch embedding framework as Jiang et al [6]. As mentioned in the description of architecture 1, each patch of pixels is projected through a fully connected layer into $\mathbb{R}^{64}$ and also positionally embedded. However, we realized that a fully connected layer from pixel space to embedding space could struggle to preserve the local spatial information of the pixels. This is why we decided to borrow from the typical U-Net GAN and introduce convolutions into our third architecture.

Architecture 3 is a copy of architecture 2 with an updated patchification process. Now instead of the patch embedding function projecting the patches from pixel space into $\mathbb{R}^{64}$, it will project the patches from some spatially aware latent space into the same projection dimension. To do this, we implement a new patchify function which first, partitions the image into a sequence of 1024 patches each with size 8×8 pixels. Then, unlike in previous architectures, we apply three convolutional filters with ReLU in-between. The convolutions have a kernel size of 3 with a stride and padding of 1 so that at each step the patch has size 8×8. The kernels have depth 64, 128, and 256. We then apply adaptive average pooling on each patch giving us a single value for each filter which results in a 256 dimensional vec-

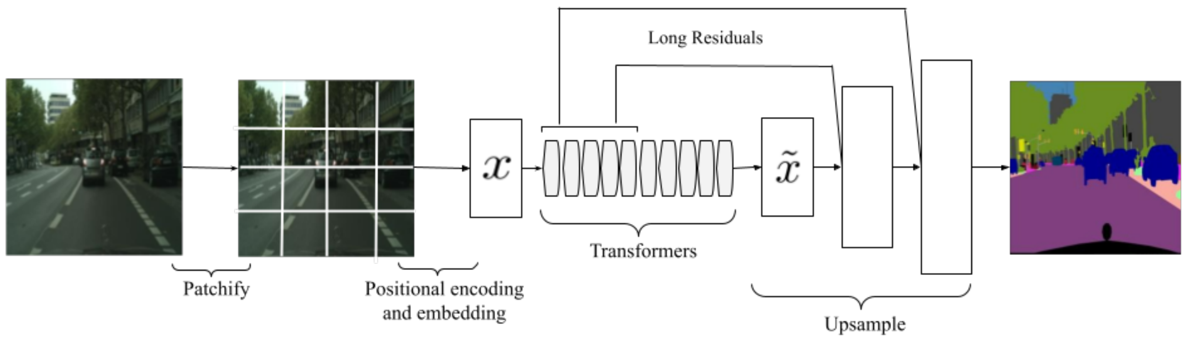

Figure 3. Second architecture

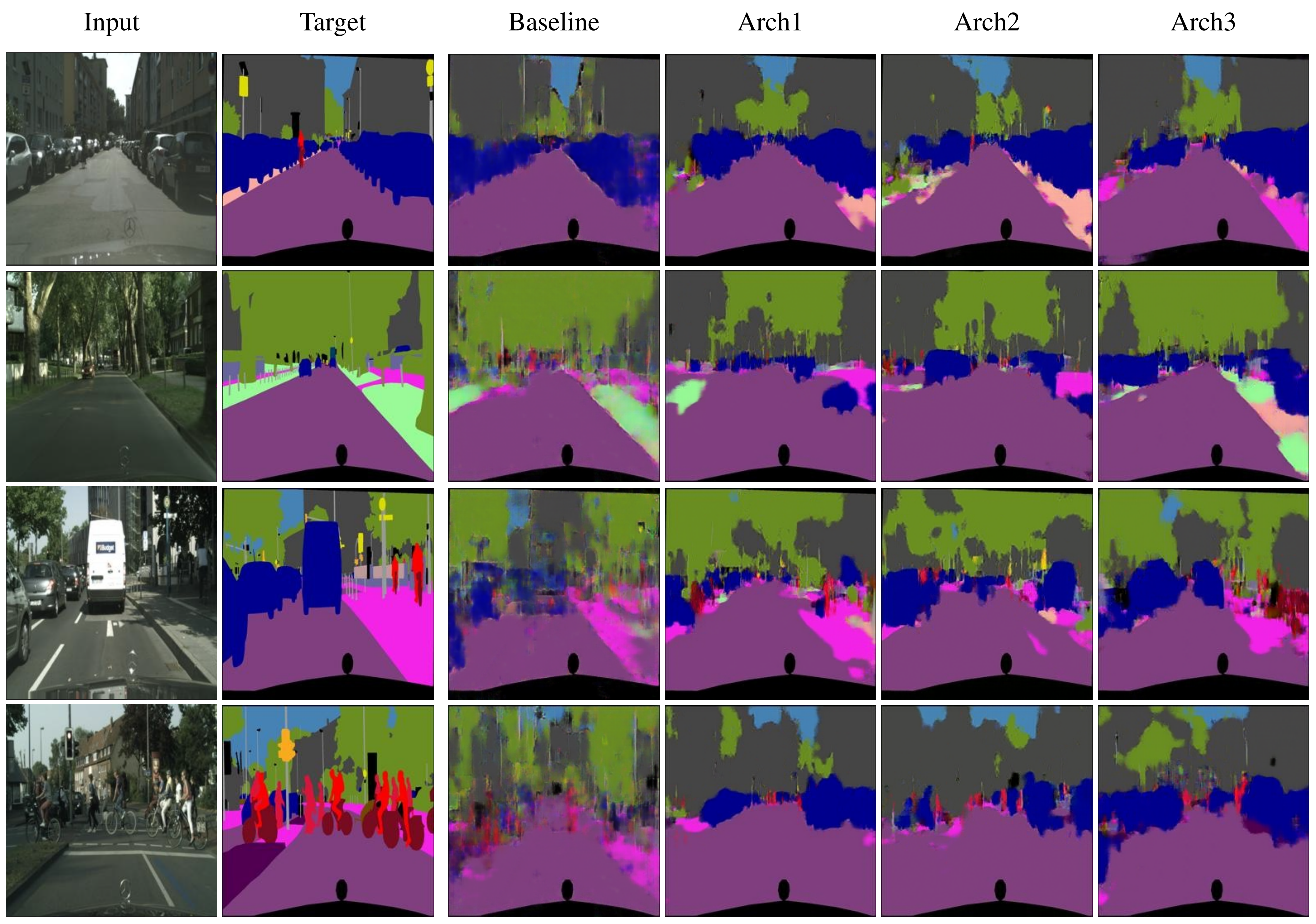

Table 1. Generated images on validation data. Target and input column represent validation data pairs. Each model output is shown in the following columns. The colors of each image correspond to types of objects in the Cityscapes dataset.

tor for each patch. We then project this vector down into 64 dimensional space and feed it through the rest of the patch embedding which is identical to architecture 2 (more linear layers and then positional encoding).

4. Experiments

4.1. Metrics and Training

Even though we can evaluate the models subjectively by assessing the realism of the results, we employ a more objective evaluation metric through the use of FID scores, which quantify the distance between the model's generated images and a reference test dataset. Specifically, the Fréchet Inception Distance measures how similar the distribution of the real, target images is to the distribution of the generated images. To do so, we run our trained generator through a validation dataset to get generated images. Then, an InceptionV3 model converts these to lower-dimensional features, along with the target images. Finally, we fit a multivariate gaussian with mean μ and covariance Σ on each set of features. Namely, $\mathcal{N}(\mu_\text{g}, \Sigma_\text{g})$ is the best fit Gaussian to the features corresponding to the generated images, and $\mathcal{N}(\mu_\text{t}, \Sigma_\text{t})$ corresponds to the distribution representing the features of the target images. Then, the FID score is

$$\text{FID} = \|\mu_\text{g} - \mu_\text{t}\|^2 + \text{Trace}\left(\Sigma_\text{t} + \Sigma_\text{g} - 2(\Sigma_\text{g}\Sigma_\text{t})^{1/2}\right)$$

FID scores have become the standard for evaluating GAN performance, and tend to produce more consistent results than other popular metrics, such as Inception scores. Both are thoroughly covered by Choi et al [1]. Moreover, we will also keep track of the L^1, generator, and discriminator losses to compare the training efficiency on each model. Due to the increased complexity of the proposed models, particularly Pix2Trans2Pix2GAN v3, we expect them to take longer to produce satisfactory results, but believe their performance will be superior in the long term. We train each model for 500 epochs, since GAN training is expensive and highly unstable. We use the Adam optimizer for both the discriminator and generator updates, and use batch sizes of 16 images each. The total loss is a weighted sum between the classical conditional GAN loss 3.1 plus the L^1

loss between the generated and target image, with weights 1 and 100, respectively. As hyper-parameters, we have experimented with others and have found those to work best.

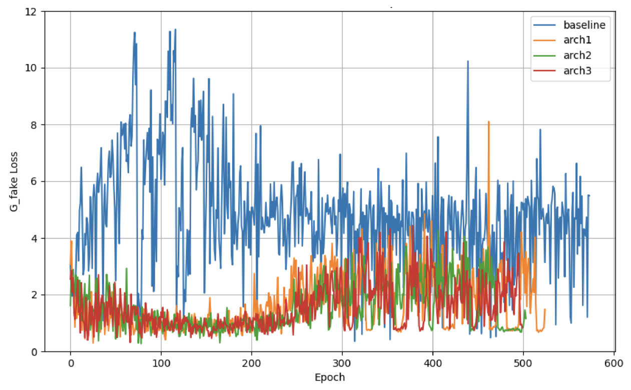

Figure 4. Generator losses throughout training epochs

5. Discussion

5.1. Baseline Comparison

Through our experiments, we confirmed our hypothesis that a transformer based generator can outperform the standard convolutional U-Net generator. While the baseline generator achieved a minimum combined reconstruction and adversarial loss of about 7.5 in 500 training epochs, each of our transformer based architectures achieved half of that or less [Table 2]. Also it can be seen in the training losses [Figure 5] that the baseline model's reconstruction loss plateaued earlier and higher than the transformer reconstruction loss suggesting that the attention of the transformer architecture allows it to find more room for optimization better than the U-Net.

The same figure also highlights an interesting finding which is that in the first 100 epochs or so, the convolutional model experiences relatively lower L^1 losses than the other architectures. We will discuss this more later but for now it makes intuitive sense that the more spatially-aware convolutional model would exhibit short term success over a more sophisticated transformer model dealing with patch sequences.

Another interesting difference between the baseline and transformer generators during training is that the adversarial losses of the transformer models are much more stable than that of the baseline [Figure 4]. In general, adversarial loss can be difficult to interpret in GANs because the discriminator and generator are both learning at the same time so any improvements in one will come at the cost of the other. This results in the losses oscillating from epoch to epoch. However, as the discriminator and generator both learn to perform their respective duties better, their adversarial losses should stabilize and converge to some positive value. This value will correspond to the loss in the ideal situation when the discriminator predicts every real image correctly and will correctly classify a generated image only half of the time. The stability of the adversarial loss in our transformer based generators suggests that the variance in the "believability" of a generated image is less than in the convolutional generator images. This means that our architectures will more consistently generate similar quality images while the baseline model is more likely to once in awhile predict extremely poor images (notice the spikes in the baseline adversarial losses of [Figure 5].

Beyond training, the transformer based generators also scored much lower FID scores on the validation set than the baseline [Table 2]. The validation set consisted of 500 randomly selected images from the Cityscapes dataset and the models did not have access to this data while training. The FID score (Fréchet Inception Distance) compares the distribution of the set of generated images for each model to the distribution of the target images. A lower FID score means that the distribution of the generation image set more closely resembles that of the ground truth set. All of the models in this study achieved high FID scores when compared to typical GAN literature but we believe this is due to the limited training time and budget combined with the sensitivity of GAN training to hyper-parameters. However, because the FID scores were calculated for each model after a common training period, we choose to focus on the relative difference in the FID scores between the four architectures.

Finally, it is worth noting that the baseline model has about 54 million parameters while the transformer architectures all have less than 30 million. This means that not only did our transformer based architectures show improved metrics in loss, FID, and stability but it also did so while using half the number of learnable parameters. This is likely because the U-Net encoder-decoder has several con-

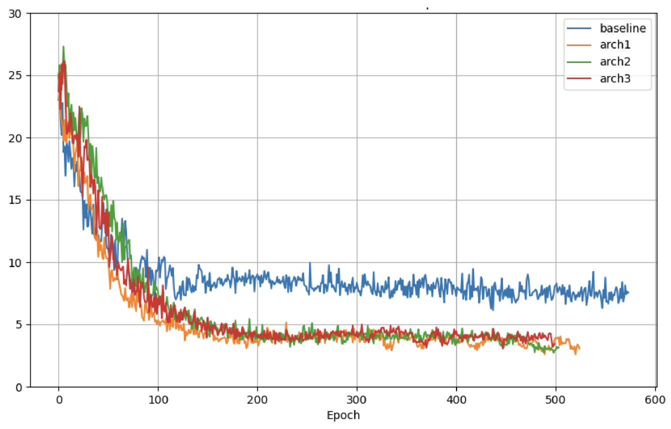

Figure 5. L^1 losses throughout training epochs

Model	FID	Min Loss	Variance	Parameters (M)	Train time (h)
Baseline	212.24	7.57	3.63	54.4	2.2
Arch1	130.39	3.30	0.91	23.9	3.5
Arch2	124.76	3.40	0.63	28.5	3.2
Arch3	134.67	3.90	0.64	28.9	6.1

Table 2. Metrics and training statistics

volutional filters which contain many more parameters per layer than our respective transformer encoders. Recall that the baseline generator encoder used 6 convolutional layers while architecture 2 and 3 each used 10 transformer layers. Therefore the bulk of the baseline parameters come from the fact that convolutional weights store more raw parameters than transformer layers. However, the number of parameters may not be the best way to measure the relative complexity of the architectures because the training time of our models were much longer. This is likely due to the difficulty in calculating the gradient in a transformer based architecture as opposed to a mathematically simpler convolutional network.

5.2. Architecture Analysis

Now that we have established that transformer based generators can outperform the baseline, lets compare the three transformer architectures and evaluate whether our design iterations showed the intended improvements.

In FID score, minimum loss, variance, and number of parameters, architectures 1, 2, and 3 show essentially the same results. According to our statistics of Table 2, the model quality is reversed from what we expected and our intended improvements did the opposite. The numbers are so similar though that this could be a result of the particular validation data sample. However the point remains that our iterations on architecture 1 (namely: deepening the model, adding long residual connections, and convolving patches) did not significantly improve the model performance. This is also verified by the similarity in quality of generated images from Table 1. Not only was the performance roughly the same, but the training time for architecture 3 took much longer than any other model.

In summary, the difference between transformer based generators and the standard convolutional generator is certainly noticeable, but there is no noticeable difference in quality between the three architectures we proposed.

5.3. Generation on Validation

Beyond the numeric ways to evaluate the performance of the various models, it is also important to explore the qualitative results as well. To this extent, we will consider the generated images of each model on the validation set of Cityscapes images with our own eyes. Table 1 shows a small sample of the validation set that represents some of the key aspects found throughout the dataset. Each input image was fed into each generator and the outputs are shown in the table. Note that although we only show four images in the table, the patterns we identified (and will describe below) are based on peering through hundreds of validation examples.

Initially while training, we thought that the baseline would outperform transformer based models because the convolutional model tended to produce better looking images earlier in training than the transformer models. This is seen in the lower L^1 loss by the baseline in the first several epochs (Figure 5). However, the quality in generated images by the transformer models soon caught up.

We noticed that in the Cityscapes dataset, there are certain types of input images that are more difficult to translate. For example, every model seemed to generate more realistic images when the input contained many parked cars on the sides of the street or when the street contained very few cars. More difficult images are obviously those with extraneous features that are less frequent in the training data such as vans, cyclists, or pedestrians.

Although all the models perform best on the "easier" images, it is actually the baseline that seems to give the superior quality image translations here instead of the transformer generators. All of the other architectures show very similar quality across their generation on simple images although we noticed architecture 2 and 3 give slightly better reconstructions of the input image than architecture 1. A significant issue we've encountered with the transformer generators on the majority of the images, even in the simple ones with open roads, is the fact it hallucinates cars often. In most of the early to middle stage epochs of training, the transformer based architectures will generate cars on one or both sides of the street where there are none in the input image. Even in late stage epoch generations (like the ones shown in the table), there are many blue blotches symbolizing parts of cars. The explanation for this is simple, the training data has many images of busy streets with parked cars lining the sidewalk and so when the generator is learning to fool the discriminator, it will find that the image is realistic looking if it has many cars on the side. After longer training times, this hallucination effect is somewhat mitigated by the discriminator improving and using its ability to condition on the input as well as the L^1 loss becoming more of a factor as the adversarial loss decreases.

However, there is room for improvement on the outliers in the dataset distribution, which corresponds to the more complex images. This is largely due to the notorious problem that GANs tend to suffer from mode collapse. However, we noticed that the baseline model performed worse compared to the transformer based models on such images. The resultant output of the baseline model is characterized by widespread smudging, with scarce discernible shapes. This matches what we observe in the adversarial loss stability (Figure 4) of the baseline mentioned above. The transformer based models also encounter some difficulties, but they tend to produce clearer looking shapes (cars, sidewalks, etc.) with more defined outlines. They do however tend to hallucinate cars that don't exist even more so than in the simple images. Also, the extraneous feature such as a van or pedestrians will often not be shown in the generated output. The model chooses to ignore them and attempts to generate the rest of the image instead.

In broad terms, the baseline model demonstrates a capacity to generate highly convincing image translations for certain inputs. However, its efficacy diminishes significantly when confronted with outlier images. At the same time, the images produced by transformer-based architectures consistently exhibit mediocrity. Consequently, the superiority of transformer-based Generative Adversarial Networks (GANs) over convolutional GANs in the context of image translation is not evident.

5.4. Further Research

The success of our transformer based model over the convolutional baseline in the task of image translation serves as further evidence that transformers and attention can and should be used to update existing architectures in many areas of deep learning. However, the lackluster quality of generated images shows that there is still a lot of work to be done.

The images generated by all of these models are far from state of the art. Ideally we would like to train each model for much longer and experiment with more hyper-parameter combinations involving the relative weighting between the reconstruction and adversarial losses, the embedding dimension of our latent vectors, and the patch size.

We also would have liked to implement some GAN training tricks that we have since learned since completing the training. Some examples include training the discriminator with stochastic gradient descent while the generator is still trained on Adam that way the discriminator will hopefully improve slower than the generator giving the generator more time to experiment with different generations and hopefully mitigate mode collapse. Another similar idea would be to experiment training with dual discriminators and dual generators to improve robustness.

Finally, we initially began looking into transformer based architectures because we wanted to try and follow Sora's recent example of replacing a U-net architecture with a transformer related architecture. However, Sora uses a diffusion transformer based on the paper by Peebles and Xie [8]. We would have liked to implement a diffusion transformer architecture inside of a GAN. Perhaps this could be a further iteration of this project and the results can be compared to the four generators discussed in this paper.

5.5. Conclusion

In this study we presented three different architectures to improve the popular Pix2Pix conditional generative adversarial network. We hypothesized that a transformer based architecture would be well suited for a conditional Pix2Pix generator because the input image can be interpreted as a sequence on which a transformer can use attention to produce a generated translation. We iterated on our own model designs by identifying weaknesses and developing new ideas to address those weaknesses. Although the generated images are not quite state of the art, we provided empirical evidence that the transformer based generators can outperform the baseline convolutional generator with the training resources available to us.

Furthermore, we both learned an immense amount about deep learning content and project collaboration. We faced many frustrations such as changing ideas after the first proposal and learning that GANs are extremely tricky to train. In hindsight there are many things we would have done differently from the start. However, these setbacks ultimately taught us some of the most valuable lessons. We are not so unlike these models in a way. Through an oscillatory sequence of wins and losses, we learned to minimize the losses and adapt to meet new challenges.

References

[1] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 5

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014. 1

[4] Qi Guo, Sheng Gao, Xiao Zhang, Yafeng Yin, and Chao Zhang. Patch-based image inpainting via two-stage low

rank approximation. *IEEE Transactions on Visualization and Computer Graphics*, 24(6):2023–2036, 2018. 3

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018. 1, 2, 3, 4

[6] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up, 2021. 2, 3, 4

[7] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. 2

[8] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. 8

[9] Aladdin Persson. Machine learning with python. `https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/GANs/Pix2Pix`, 2023. Accessed: January 15, 2024. 2

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 2

[11] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2018. 2

[12] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. Styleswin: Transformer-based gan for high-resolution image generation, 2022. 2