StarGan V2 for Generating Realistic Characteristic Changes (ICML 2021)

Abstract

Using generative adversarial networks to create new or modified images has a lot of uses today. It can be used to aid in future machine learning by creating new datasets from pre-existing ones. They can also create very realistic pictures of objects that do not exist, and fill in the blanks in a sketch to create a photorealistic match, which would be hugely useful in a design context. They can also transfer the style of one painting or image to another, and generate realistic looking pictures from a text description, which is a boon in creative and artistic endeavors. Perhaps most interesting and the focus of this project, however, is photorealistic renderings of faces that have been imposed with different characteristics.

This would definitely be useful in the contexts of the beauty products industry: imagine being able to realistically see how you would look with different colored hair or products or clothes instantly before you make a purchase. Furthermore, it has life saving importance in the context of medical imaging, from modeling anatomies, cells, or being able to identify tumors with greater accuracy and with more attention to detail. Eventually, this may be useful in simulating models of humans or animals in virtual meeting or virtual reality contexts.

Therefore, for it to be useful in a real world setting, it should fulfill two requirements: have a great diversity of types of images and many different subjects, and have scalability over multiple characteristics. There are other research papers that are able to achieve one or the other, but StarGan v2 is able to accomplish both with a high degree of visual convincingness. It works using the CelebA dataset and the Animal Faces dataset, which was not available for StarGan v1. In terms of implementation, StarGan v2 was able to accomplish this with a mapping network that uses Gaussian noise to generate style codes, and a style encoder that learns to distinguish the style codes embedded in an image. Furthermore, in our implementation of the project, we have introduced two critical changes to the code. First, we have changed the use of Pytorch into Tensorflow. Secondly, we have replaced one of the layers with a modulated convolution to enable us to input a larger image size. The results of this experiment were mostly successful, as seen from the discussions below.

1. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation

1.1. Review

As a general overview, this paper starts from a place where the research in the field has shown how models can take images of faces from one domain (a certain attribute like hair color or gender), and translate it into another single domain (Choi et al., 2018). However, this lacks the functionality to be generalizable to more than one domain. StarGAN overcomes this lapse in functionality by enabling multi domain image to image translation, meaning it is better at changing the input in multiple attributes.

Previously, the generators that create the output images will work from one domain to another, i.e., from blond hair to dark hair. This is not efficient or effective because it will require a generator between every combination of domains. Additionally, each generator has less access to learn about common qualities from the whole data set, like face shape, because each generator only has the training set of one domain. It is less able to utilize the full training set and learn about common attributes that all faces have.

Instead, StarGAN uses a single generator that flexibly translates between all domains. It uses a label vector for each image that identifies its domain and is able to switch between any randomly decided target domain. Additionally, StarGAN employs the use of a mask vector to allow training and testing between datasets. So far, no other study has shown the ability to train using two different datasets with different domains. They have previously needed the image to be labeled for the domain the image learns from, but with

the mask vector, the researchers are able to change which domains the generator focuses on and which it ignores. It can learn facial expression attributes from one dataset of faces and apply it to another dataset whilst learning from both. StarGAN uses both advantages to train a more convincing facial attribute transfer than other existing models.

This model uses generative adversarial networks to increase its effectiveness. This has another network, a competitive discriminator, that tries to get better at telling which images are fake and which are real while the generator tries to get better at fooling it. This vastly improves the quality of the image to image attribute transfer.

Conditional GAN has both the discriminator and the generator with class information and makes the model more effective.

1.2. Strengths

This was a well written paper with a high degree of clarity. It was understandable and often would connect the statistics and implementation back to the image to image functionality. The examples of the test image outputs were organized and clearly laid out.

This was quite a landmark paper for formulating a new and innovative approach to creating convincing image generation of faces, which is difficult to do considering how attuned humans are to recognizing faces. Through its use of a single multi-domain generator to train all characteristics simultaneously, it solved both an efficiency problem with the computational intensity of running numerous GANs concurrently, and created much better, crisper images with fewer imperfections than other methods. Overall, this is a strong paper for the field of image generation and expanding the use cases for generative adversarial networks.

1.3. Weaknesses or Other Thoughts and Questions

At first it was not obvious how a starGAN can have much better results than previous cross domain models when they are working with the same volume of data in the dataset. It was eventually more clear on a second reading that the advancement was the labels that allow a single generator to improve on the whole data set, rather than many generators that only saw a portion of the data set in the form of a single domain.

For us, it is not clear how the label vector allows the single generator to flexibly switch between all domains, when it was not possible before. It makes each image identifiable with a certain domain, but is this sufficient to replace cross domain generators? It seems quite simple for a researcher in an earlier paper to attach a label to pre-labeled images. We believe the authors should have made a stronger case for this advantage for using StarGAN.
The experimental results showed an obvious favor for StarGAN, though not as unequivocally as we would have assumed.

2. Image to Image Translation with Conditional Adversarial Networks

2.1. Review

This paper made a big splash in the non-scientific community, as the pix2pix software that this was released with was easy and generalizable enough to use that many people were able to quickly demonstrate their own use of the model and post outputs on social media (Isola et al., 2017). It is the high quality and open-ended nature of this model that makes it convenient to use and a new breakthrough in image to image predictive translation.

Again, this image to image translation tool uses convolutional neural networks (CNN) to input some data, pixels, and evaluate it to minimize a cost function then, minimize that cost output in order to learn. The problem here is that there is still a lot of work to be done to design an effective cost function in order to get the result we want. From here, this CNN is the generator and an adversary. The discriminator CNN tries to discern whether the generator output is a fake or not. As the discriminator improves its ability to distinguish, the generator creates more high-quality outputs.

The main improvement this paper made was the use of Conditional GAN, specifically in the context of a general image to image translation. The conditional GAN has the input and an additional classification label to improve the function of both, using minimal extra input to create a directed transformation on the image. This is done unspecifically, such that the same program can extrapolate in many different ways.

This paper also uses a PatchGAN, which simply lowers the cost score on image size patches, so it is not as fine grained in its evaluation.

The experiments detailed in this paper range from a wide

array of tasks; from graphics tasks to vision tasks, i.e., creating aerial photos, coloring images, taking photos from sketches, and semantic segmentation.

2.2. Strengths

This paper was quite effective at producing high plausibility to the human observer. It was very thorough, and it demonstrated its usage in the real world on social media, which is very encouraging. The authors wrote this paper in such a way that was very coherent and made the objectively complex information understandable. Overall, this is a promising paper in the development of human level quality in performing certain tasks.

2.3. Weaknesses or Other Thoughts and Questions

The explanation of cGANs was somewhat vague. This may be a consequence of the generalizability of the cGANs model, but more examples would have helped. We found ourselves needing to look up the specific uses of cGANs in practice to gain a fuller understanding of its applicability.

Furthermore, many of the transformation qualities were tested only using perceptual studies from AMT. The authors note that obtaining quantitative results for image to image transformation has historically been difficult and a large part of the criteria of success for this project is subjective to human plausibility. However, relying mainly on human level grading does not always seem optimal. Furthermore, there was no vigilance testing for these human tests, so the metrics may be skewed with bots or bad actors.

3. Image Style Transfer using Convolutional Neural Network

3.1. Review

Being able to render semantic content from images has been a very difficult problem. However, using convolutional neural networks, this paper has been able to lift the style of one image, and transplant it over the structure of another image. For example, it uses Van Gogh's Starry Night, and redraws a normal photograph in the style of this painting.

There have been previous attempts to transfer style over the content of another image. All of these methods, however, simply take samples of textures from one image and intentionally overlay them around the main features of another image. This is an imperfect solution and does not recreate the image in a new style, but just overlays it. Using the convolutional neural network to train and learn the style, the program was able to recreate scenes in the style of famous artworks or even photographs of other cities (Gatys et al., 2016).

The authors were able to accomplish this by separating the content image, which is the objects and their arrangements, from their style images, which are the colors and textures of the image. First, the researchers cleaned and processed the images to have the same size, and then loaded them in the pre-trained VGG16 convolutional neural network. Their fundamental distinction from other experiments, which arises from identifying the layers that identify content versus style, are separated to allow them to work independently.

They are then optimized for minimizing content loss, so that we preserve the subjects of the picture. Also, the style loss is minimized so that our final output looks like the target style. Finally, variation loss is minimized, so that we can denoise the final output. The gradients are set up with the L-BFGS algorithm and the losses are optimized. This gives us the final output of a new image, with the contents and picture of one image, redrawn in the style of another.

3.2. Strengths

This paper was very able to display all the nuances of the program and seems to be one of the first using CNNs in this way. There was specifically a focus on discussion and open ended questions that encouraged more exploration. The paper was very organized and laid out the formulas well.

This landmark paper has the most use cases in computer vision, and finding and labeling objects in pictures. The major breakthrough in this paper was how the researchers were able to discover that the feature maps in the deeper layers of the neural network had very useful information about what the picture actually contains. They also found that the style could be extracted from the first layer of the convolutional neural network, which is what allowed them to apply styles to other images. Overall, the authors did well in detailing how they were able to build their system.

3.3. Weaknesses

The content was a bit difficult to follow at times. The paper was slightly disorganized thus creating more confusion. The fact that there is no conclusion is both a good thing, as it encourages more research and is an open ended question, but also does not summarize the whole paper at the end. It would have been nice to gain an insight into

how the researchers were going to improve upon their original implementation or the model's applicability and so forth.

4. Project Description

4.1. Implementation

Using StarGan v2, we are aiming to learn a mapping between two visual domains, where domains are the set of visual characteristics that a labeled group shares. For example, the "male" domain shares beards and short hair. Traditionally, generators were used for each domain to domain mapping. This carries a high computational cost and is inefficient as there could be many different domains, which means multiple mappings for a single generator.

StarGan v1 was able to overcome this by learning all domain mappings at once, and using a single datapoint for multiple mappings. However, this still falls short as this version is not flexible between particular characteristics because it relies on a single one-hot vector attached with each data input that was fixed thus, limiting the options.

StarGan v2 overcomes this by generating diverse images over multiple domains and departing from rigidly assigning styles to domains. This makes it much more generalizable and versatile to different inputs. On top of this, we implemented a change that was made to replace one of the layers from the project with a modulated convolution. This is done so that we can use images with the size of 512x512 pixels, rather than just images of size 256x256 pixels. Having the greater size is a big benefit, as the use cases for 256x256 pixel images are more limited and difficult to see and compare than a 512x512 pixel image.

4.2. Framework

In detail, there are a few things that StarGan v2 does differently from other papers and even Stargan v1 (Choi et al., 2020). It uses 2 inputs: a source image for extracting the content or identity of the object; usually features like face shape, position, facial structure, and expression, and a reference image to impart the visual style or appearance, color, texture, etc. onto the source image. There is also the target domain, which is implicitly tied to the category of the reference image, though it does not need to be. This is the biggest departure from the previous papers, as every module in StarGan v2, other than the generator, indiscriminately creates an output for each domain, regardless of its inputs. This means that an input of a cat can be imparted with the visual style of a dog, while fitting into the domain of a wildcat.

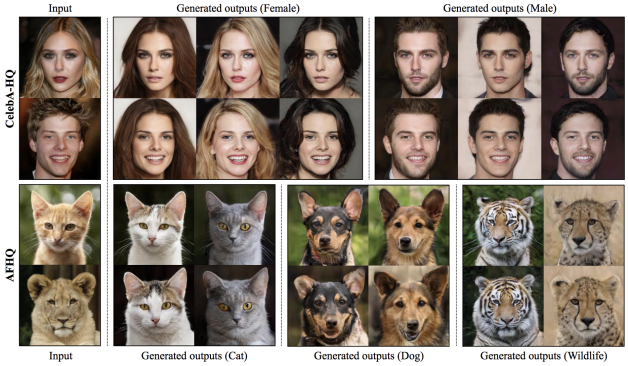
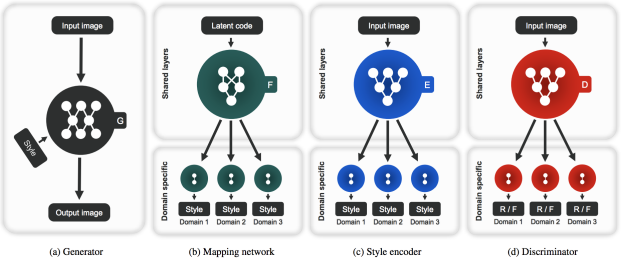


Figure 1. An example image from the Stargans v2 paper demonstrating taking a source images facial structure and poses, and overlaying the style, or hair, color, textures, of someone else over it.

The conceptual domain, more of a social construct defined by the world or researchers, can be totally divorced from the visual style, a representation of attributes learned by a model independent of a domain. This is a promising sign for the future of machine intelligence, as this separation of style from socially constructed, narrow domains, like gender, can be learned with more nuance and allows for a richer understanding of the world. It opens a framework for a less rigid definition of people that machine intelligence is usually associated with.

In order to replace the one hot vector domain label with a specific, varying style code, StarGan v2 makes use of 2 more convolutional neural networks: a mapping network that takes Gaussian noise and transforms it into a style code, and an encoder that learns how to extract style codes from an image. Both have multiple domain output branches, which is what makes them good at generalizing between domains with a high degree of diversity. They work in conjunction with a single generator that takes an input image and a style code, either from the mapping network F or the style encoder E.

An adaptive normalization instance (AdaIN) is used to inject the style code into the generator. The mapping network uses a latent code z to produce a style code associated with each particular domain, but only the output of the domain of the reference image is chosen and used in the generator. This is the same way the encoder works: by having an output for each domain, the encoder and mapping network can train the connections between all domains for each image, while only one relevant domain matching the reference is chosen. Like other adversarial networks, the discriminator trains to distinguish real images from the generated ones.



(a) Generator (b) Mapping network (c) Style encoder (d) Discriminator

Figure 2. An overview of the framework of the project, with the generator G, mapping network F, encoder E, and discriminator D. Notice how all components except for the generator produce an output for each domain.

4.3. Training Objectives

The adversarial objective is for the mapping network F to learn how to give the correct style code s while the generator G tries to create a convincing image using this style code. This means the adversarial loss is $\mathcal{L}_{adv} = E_{x,y}[\log D_y(x) + E_{x,\widetilde{y},z}[\log(1 - D_{\widetilde{y}}(G(x,\widetilde{s})))]$.

We don't want the generator to start simply neglecting the style code in order to prioritize a convincing image, so we also have a style reconstruction loss objective to minimize the difference in style between the given s and the output image: $\mathcal{L}_{sty} = E_{x,\widetilde{y},z}[\|\widetilde{s} - E_{\widetilde{y}}(G(x,\widetilde{s}))\|_1]$.

Next, we need to regularize generator G with a diversity sensitivity loss in order to foster a range of diverse image outputs: $\mathcal{L}_{ds} = E_{x,\widetilde{y},z_1,z_2}[\|G(x,\widetilde{s}_1) - G(x,\widetilde{s}_2)\|_1]$. This forces G to explore the image possibility space and discover new useful image features.

Finally, in order to make sure the generated image preserves some of the content and structure of the original image, we need to use a cycle consistency loss: $\mathcal{L}_{cyc} = E_{x,y,\widetilde{y},z}[\|x - G(G(x,\widetilde{s}),\hat{s})\|_1]$. This will minimize the difference between the original image and the output image.

This means the full objective function is min G,F,E maxD (Ladv + λsty * Lsty λds * Lds + λcyc * Lcyc), and each of the lambdas are hyper parameters that tweak the weights of each loss function on the output.

4.4. Extension

There are two main ways that we changed the base code to substantially extend the code. First, I changed the implementation of the experiments from using pytorch to using tensorflow. To do this, I became very familiar with the code and how it was implemented originally. We spent a lot of time understanding the paper, the methods and reasoning for choosing certain processes, and what the results of those were. Only then were we able to intelligently alter the code with the new package without working against ourselves in the process. It would ultimately take more time and result in a drop in quality were we to rush through the changes without having a good grasp on the code.

We slowly replaced uses of the PyTorch module with a Tensorflow implementation. To accomplish this, we carefully replaced only one function at a time and tested that the code still functioned before continuing on. Rather than use the entire dataset for this testing, we split the data into a subset comprising of only a few images to train on. This way, we could confirm it still worked without needing to spend the time and effort retraining the entire dataset of thousands of images. We would continue to do this until the project was implemented with Tensorflow. Once it was still confirmed to work well, we were able to test with training with the full dataset.

The next way we modified the experiment was by using a modulated convolution to replace the AdainResBlk, which is used by the decoder in the generator (Hramchenko, 2021). The generator takes the structural information from the source picture and passes that content information, along with the style code information from the other convolutional networks, as the input to the AdainResBlk convolution modules. It is made up of the adaptive normalization instance AdaIN modules mentioned earlier. We instead replaced this network with the modulated convulsions from another paper, StyleGan 2, with the block GenResBlk in order to be able to take the input size of the picture with a convolution that matches it.

We also replace the original forward function of the generator with some lines that directly interface with the GenResBlk from from StyleGan 2, which require inputs in RGB image stream. After this, we can start training on some 512x512 size images from the cycleGan and pix2pix repo, since the original code does not provide this.

After this, the model was able to train on 512x512 pixel images, though the dataset needed to be replaced with the fake images from the other projects, and we could begin the testing and comparing of results to the other benchmarks.

4.5. Results

The result was more or less successful. The rerun experimentation of the paper was successful in recreating the expected results with the expected loss functions. Afterwards, using the modular convolution, the input and output size of the images were able to be significantly increased. This is a benefit for use in more high definition pictures, as is the norm nowadays, and more common picture sizes. There are not as many uses for a generated image that is smaller than a thumbnail.

In particular, we want to observe the difference between our implementation, the original research teams implementation, and the implementation of other teams that are trying to solve a similar problem. There is the also the benchmark of comparing the generated images to real pictures, and scoring those real pictures against our generated images as if they are also generated images.

4.6. Discussion of Results

We can see how successful the experiment went with two ways to evaluate the quality of the images other than visually. One is the Frechet Inception Distance, which measures the difference between the high level view of two images, a real example from a domain, and the generated images. Therefore, a lower number is better for the FID and though it varies between domain mappings, the mean FID is 45.226.

The other metric to evaluate the model is the learned perceptual image patch similarity, or LPIPS, that measures the differences between two small areas of images between generated images, and is ideally maximized in the desired outcome of varying and diverse images. This value is .38.

As we can see from the figure below, the LPIPS number is fairly good with this experiment of only 3000 iterations and the images are nearly ideally varied. This may be because it is less difficult to create highly varied images rather than convincing images. The FID is not nearly as good as the original research, however. Because the FID should optimally be lower, it still surpasses 2 of the other teams with different methods that the StarGans v2 team compared to in their paper. It should also be noted that when the original StarGans v2 team were able to produce their results with Latent guided synthesis, or using the latent codes, the images were of even higher quality, and had a lower FID score than real images, which is impressive. Additionally, it had the ability to generate new hairstyles realistically and overcome the hurdle of generating ears.

Method	CelebA-HQ	
	FID	LPIPS
MUNIT [13]	107.1	0.176
DRIT [22]	53.3	0.311
MSGAN [27]	39.6	0.312
StarGAN v2	**23.9**	**0.388**
Real images	14.8	-

Figure 3. A comparison against our refrence guided implemntation of the FID and LPIPS scores, along with other teams compared in the paper.

StarGAN v2	**13.8**	**0.453**
Real images	14.8	-

Figure 4. StarGan v2 compared against real images in latent guided synthesis of new images, demonstrating a high degree of realism.

4.7. Conclusion and Future Insights

We can see that this paper has many practical and impressive uses in the future. As a more flexible framework that separates the domain classification from the style representations, it represents the shift to the direction of flexibility and nuance that is much more useful. It can be used for cosmetic purposes, life saving medical imagery, and perhaps in computer vision and 3D modeling.

References

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Hramchenko, V. Modifying StarGAN V2

using Modulated Convolutions. https://v-hramchenko.medium.com/modifying-stargan-v2-using-modulated-convolutions-13dc5796cd6e, 2021. [Online; accessed 19-November-2021].

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.