PokeDog CycleGAN- Experiments in Encouraging High Level Feature Change in CycleGAN Architecture
Author: Avidesh Marajh

Anonymous CVPR submission

Paper ID *****

Abstract

Cycle-Consistent Adversarial Networks(CycleGAN) are a variation of GAN architecture which effectively performs unpaired image translation. This architecture has seen great results in translating between classes which have low level feature differences such as styles or patterns. It has, however, seen great difficulty in high level feature changes, specifically geometric changes. In this paper, I shall outline attempts at improving the CycleGAN architecture in order to better achieve geometric changes. Using the problem of unpaired translation between images of dogs and images of dog-like entities from the popular media franchise, Pokemon, I shall demonstrate several proposed methods for a novel CycleGAN architecture which can achieve geometric changes. These methods includes additional terms for the loss functions of both the generators and discriminators as well as random grayscaling of input images. These methods achieve unsatisfactory results in geometric changes to the source image but do increase the fidelity of outputs.

1. Introduction

Pokemon is an iconic part of many young adults' childhoods. Known for its adorable character designs and distinct art-style, people have interacted with this franchise through video games, anime, trading card games and various other merchandise. It comes as no surprise that it is the highest grossing media franchise of all time, leading its runner-up by 20 billion dollars. Be it nostalgic fans or eager pet parents, many people would certainly find joy in seeing their beloved pets in the style of these lovable Pokemon. This project, inspired by my dogs Onix and Riolu, seeks to achieve just that.

In this paper, I present my findings in attempting to apply Cycle-Consistent Adversarial Networks(CycleGAN) to this problem of unpaired image translation between dogs and Pokemon. Generative Adversarial Networks are generative models which consists of a generator and a discriminator which learn simultaneously. The generator learns to generate outputs of a specified class and the discriminator learns to distinguish between fake generator outputs and real datapoints from the dataset. They are driven by an adversarial loss, such that each model minimizes its loss by outperforming the other.

The CycleGAN architecture is a method for unpaired image translation which utilizes two discriminators and two generators. For two classes of images, X and Y, GeneratorXY takes an input of a real image from class X and produces a fake image of class Y. Conversely, GeneratorYX takens a real input from Y and produces a fake image of X. The discriminatorX aims to discern the outputs from GeneratorYX from real datapoints of X and discriminatorY aims to discern the outputs from GeneratorXY from real datapoints of Y. What truly defines the CycleGAN architecture is its multi-term loss calculation. The discriminators are still driven by adversarial loss from their corresponding Generator, but the Generators are driven by, in addition to adversarial loss, a cycle-consistency loss and an identity loss. The identity loss penalizes the generators for a metric of difference between an output y' for an input image y of class Y to the generatorXY (ideally it would perform no change for an input y). The cycle-consistency loss penalizes the generators for a metric of difference between an input x of class X and an output x', where x' is the output of the generatorYX for an input y' and where y' is the output of the generatorXY for the input x.

These loss terms define the unique performance of CycleGAN's which allow them to perform image translation tasks with little information loss. That is, the model preserves high level features and performs minimal change. Correspondingly, the CycleGAN architecture excels in translating low level features(shapes and patterns) and is unsuitable for geometric changes and high level feature changes(recognizable artifacts or large components). This, works against our task of dog to Pokemon conversion since,

while Pokemon are mostly based on real animals, they usually have distinct shapes and features. My reasoning for selecting the CycleGAN architecture is its affinity for unpaired image translation. Had I used a paired image translation method, such as styleGAN, it would simply imitate the style of a provided Pokemon input onto the input image of a dog. The use of unpaired image translation allows for the creation of a Pokemon which has modified but recognizable features of the inputted dog, This will allows for the creation of a more original seeming Pokemon which closely resembles the inputted dog.

As such, this project served as an experiment in methods for encouraging high level feature changes in CycleGAN architecture using a variety of original methods. Some of these methods include, grayscale transformation of input images at a decaying random rate, a variety of new loss terms for both the discriminator and generator, and alteration to the architecture which involves replacing the two discriminators with a single classifier.

While none of these methods showed significant improvements in feature changes, they did show higher fidelity outputs than the base CycleGAN architecture.

2. Background

2.1. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks (Zhu et al. 2223-2232)

This paper introduces the CycleGAN architecture which is employed in our model. Naturally our model faces many of the limitations addressed in this paper. Most notably, this architecture is shown to be ineffective for problems which require geometric changes. This does limit the fidelity of dog/Pokemon translations as the Pokemon are not always quadrupedal and may have various additional features not common to dogs.

While the author makes no suggestions for addressing this issue, I attempt to encourage large feature changes through the use of additional loss terms.

2.2. PokeGAN: P2P (Pet to Pokemon) Stylizer (Hedge et al.)

This project aimed toward a very similar objective as this one. That is, it aimed to transfer the styles of Pokémon onto various pictures of animals. Similar to my own, their model applied blue and green tints on output images in order to generate 'Pokémon'. This is explained by the authors as being due to these being the most commonly occurring colors in the anime. This is very similar to the issues encountered by our baseline model in which a green tint was being applied to outputted dog images. This issue was left unaddressed by the authors and was dismissed as an outlier only for inputs with little color variation.

This is addressed in my project through the use of a random grayscale transformation with a decaying probability.

2.3. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks (Bousmalis et al. 3722-3731)

This paper also addressed the issue of excessive augmentation of irrelevant feature in Style Transfer problems. This issue is apparent in our baseline model by the application of filters over the entire image, not only augmenting the dog/Pokémon but also greatly altering the background. The authors addressed this problem using a content-similarity loss which uses an image mask which excludes features which should not be changed and adding the mean squared error between real and generated images to the total loss. This was extremely effective but required a pre-existing mask which can isolate the features which are required to change, which is not possible for our problem as our input images feature entities in different poses.

In my project, I implemented a diff loss function which is meant to create a minimax objective which minimizes uniform change and introduces large, isolated changes.

2.4. From GAN to WGAN (Weng)

The defining feature of GAN is the adversarial simultaneous training of two different models, a generator and a discriminator. This can lead to issues, however, when the discriminator overtakes the generator in performance restricting the generator from attempting from learning effectively. In this paper, they discuss multiple solutions to promoting stable learning. In addressing the problem of discriminator dominance, they suggested softer labels for the discriminator loss, eg. 0,1,0.9 instead of 0,1.

In my project, I attempt to address this issue using a class loss, which provides a secondary learning objective for the discriminator, making it more difficult for it to converge.

2.5. Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data (Almahairi et al.)

This paper suggests a method for diverse outputs in GANs. Their method of introducing a noise vector as additional input to the generators is very promising but I believe it is unsuitable for our problem which has a very diverse dataset and requires a focus on higher fidelity than diverse outputs.

For my project, I utilized random grayscaling in order to prevent the model from overfitting to the color mappings of the images in the datset.

3. Dataset

For this project, I utilized two sources of data:

3.1. Dog dataset

For my dog images, I employed the Stanford dogs dataset. This dataset is built using images and annotations from ImageNet and is made for large machine learning projects such as this. This dataset contains 20,580 images of dogs from 120 categories, with a variety of poses and backgrounds.
http://vision.stanford.edu/aditya86/ImageNetDogs

3.2. Pokemon dataset

For my Pokemon images, I scalped the anime screenshot of gallery from a Pokemon wiki site, Bulbapedia. Using a web request python script to iterate through their unorganized, but well labelled, gallery of Pokemon images, I downloaded all images which mentioned the name of Pokemon from a personally curated list of 50 dog-like Pokemon. This resulted in a dataset of 940 images. For this dataset, I relied on images from the Pokemon anime, rather than the video games or trading card games, due to the variety of poses and backgrounds which accompany screenshots from the anime. The code used for this can be found in the link below.
https://github.com/Deshy23/Dog2Pokedog/blob/main/scrape.py

3.3. limitations

By machine learning standards, my Pokemon dataset is very small and is greatly dwarfed by my dog dataset. This scarcity in available data greatly increase the difficulty of the task.

4. Methods

My networks are constructed using the following blocks:
Feature Map Block: A single 2D convolutional layer with a kernel size of 7 and padding size of 3.
Contracting Block: A single 2D convolutional layer with a kernel size of 3, padding size of 1 and a stride of 2. This is followed by an instance norm and a ReLU activation.
Expanding Block: A single 2D convolutional layer with a kernel size of 3, padding size of 1 and a stride of 2. This is followed by an instance norm and a ReLU activation..
Residual Block:Two 2D convolutional layer with a kernel size of 3, padding size of 1. This is followed by an instance norm and a ReLU activation.

4.1. Baseline

For my baseline model, I used a standard implementation of the CycleGAN architecture, as follows:
Generators:Each Generator consisted of a feature map block which receives 3 input channels and 64 output channels, followed by 2 contracting blocks each of which doubles the number of channels. This feeds into a series of 8 residual blocks each of which have 256 channels, input and output. This is followed by 2 expanding blocks, each of which halve the number of channels followed by a feature map block which takes an input of 64 channels and an output of 3 channels. This is finally subject to a Tanh activation function.
Discriminators: Each Discriminator consisted of a Feature Map Block with 3 input channels and 64 output channels, 3 Contracting Blocks, each of which double the number of channels and finally, a 2D convolutional layer which outputs 1 channel.
Each discriminator was optimized separately as defined by an adversarial loss function.
The generators were optimized together by a hybrid loss function of the adversarial loss(with weight 1.0), the identity loss(with weight 0.1) and the cycle consistency loss(with weight 10)
Adversarial losses used the Binary Cross Entropy as its criterion and Identity and Cycle losses used L1 loss.
This model was used as a baseline due to its verifiable performance when applied to the horse to zebra problem.

4.2. Diff Loss

My first modification of the architecture is an additional loss term for the Generator. In creating this loss term, my intention was to create a term which encourages the model to perform notable geometric changes. To do this I sought to disincentivise small uniform changes and encourage large localized changes. This was formulated as a quotient of the L1 loss and the Mean Squared Error as follows:
For an inputted image of class X, x, to a GeneratorXY, and the corresponding output y' and where L_1 is the L1 loss and MSE is the Mean Squared Error,

$$L_{diff} = \frac{L_1(x, y')^2}{MSE(x, y')} \tag{1}$$

The objective of this function is to minimize the L1 loss and maximize the MSE loss between real inputted images and fake outputted images. This is due to the nature of these loss functions, where the L1 loss penalize changes uniformly but the MSE loss harshly penalizes having few pixels which have large differences. Therefore, this diff loss will be minimized by having localized changes in the image rather than uniform changes such as a tint. That is, the image should have few pixels which have large changes rather than many pixels with small changes.

4.3. Class Loss

One of the main problems observed to be faced was the imbalance in discriminator and generator performance. This is a common failure mode for GANs and to prevent the discriminator from reaching convergence so rapidly, an

additional term was added to its loss. This class loss, for a DiscriminatorX, takes an input of a real image of class Y, y, and a generated image of class X, x', where x' is the output of the GeneratorYX for an input y. It then finds the discriminators prediction for each term and computes the L1 loss between these terms. That is,

$$L_{class} = L_1(disc_X(x'), disc_x(y)) \tag{2}$$

This loss function seeks to minimize the difference in prediction for an image of class Y and its corresponding fake image of class X. This is to increase the complexity of the discriminator's task to prevent it from converging prematurely while also encouraging it to base its decision on the identification of remnant features of class Y rather than exploiting other things such as the image quality of the generator outputs.

4.4. Retention Loss

As an alternative to the Diff Loss, I also devised the retention loss, an additional loss function for the Generator. This loss function seeks to similarly encourage changes to geometric features by punishing the Generator for maintaining features from the input images. For this it leverages the Discriminator as follows:
For a GeneratorXY with some output y' and the DiscriminatorX where BCE is the binary cross entropy,

$$L_{ret} = BCE(disc_X(y'), 0) \tag{3}$$

This loss function, using similar logic as the above Class Loss, encourages the Generator to ensure its output y' is not similar to real images from class X.

4.5. Classifier

Thus far, my contributions have involved additional loss terms to shift the optimization goals of the model. This change however, makes a large architectural change. Instead of utilizing 2 discriminators for each class, I utilize a single Classifier which aims to output 0 for images of class X and 1 for images of class Y. The logic behind this change is to have a Classifier which learns some set of underlying common features between the two classes and the differences between them and encourages the Generators to utilize this underlying mapping. This also eliminates the failure mode of having the discriminators converge before the generators and thus prematurely end learning, as the the classifier faces a more complex task than the individual discriminators.
The Classifier uses the same architecture as a single Discriminator as described in the baseline model(3.1). The loss function for the new model are as follows:
Generator adversarial loss: Let $val_X = 0$ and $val_Y = 1$:
For x',y' where x',y' are generated images from GeneratorYX and GeneratorXY, and BCE is the Binary Cross Entropy,

$$\begin{aligned} L_{gen,adv} = &BCE(Classifier(x'), 1 - val_Y) \\ &+ BCE(Classifier(y'), 1 - val_X) \end{aligned} \tag{4}$$

Classifier adversarial loss For x,y,x',y' where x,y are real images from X,Y and x',y' are generated images from GeneratorYX and GeneratorXY, and BCE is the Binary Cross Entropy,

$$\begin{aligned} L_{classifier,adv} = &BCE(Classifier(x), val_X) \\ &+ BCE(Classifier(x'), val_Y) \\ &+ BCE(Classifier(y), val_Y) \\ &+ BCE(Classifier(y'), val_X) \end{aligned} \tag{5}$$

4.6. Random Grayscaling

In order to force the model to place a greater focus on geometric features, I added a random grayscale transform with a decaying random rate,

$$p = 0.25e^{-\frac{curr}{total}} \tag{6}$$

Where curr is the current step of the epoch and total is the total number of steps in the epoch.

5. Experiments

After initial testing I found that the Class loss and Retention losses give unremarkable results and actually worsened output fidelity. I trained the following models demonstrating the efficacy of random grayscaling and the diff loss.
Results will be demonstrated using the following pair of images from each class: See Figure 1

5.1. Baseline

The results are shown in Figure 2.

5.2. Diff Loss

The baseline model was modified to include the Diff Loss term(3.2) with a lambda weight of 1.
The results are shown in Figure 3.

5.3. Grayscaling

The baseline model was modified to include the random Grayscaling(3.6).
The results are shown in Figure 4.

5.4. Random Grayscaling and Diff Loss

The baseline model was modified with the addition of the random Grayscaling(3.6) and the Diff Loss(3.2).
The results are shown in Figure 5.

Figure 1. Testing Images -
top: Golden Retriever(real dog), bottom:Arcanine(real Pokemon)

6. Risky Experiments

Due to Colab timing out and revoking my GPU access(and also making me lose my trained models!!!), the following models were trained locally and thus were trained for only 4000 steps. My risky experiments centered on altering the CycleGAN architecture as highlighted in 3.5. For this purpose I trained this model with Random Grayscaling and the Diff Loss. I also trained a baseline model with the Random Grayscaling and Diff Loss for the same number of steps for comparison.

6.1. Random Grayscaling and Diff Loss

The baseline model was modified with the addition of the random grayscaling(3.6) and the diff loss(3.2).
The results are shown in Figure 6.

6.2. Classifier with Random Grayscaling and Diff Loss

The classifier model(3.5) was modified with the addition of the random grayscaling(3.6) and the diff loss(3.2).
The results are shown in Figure 7.

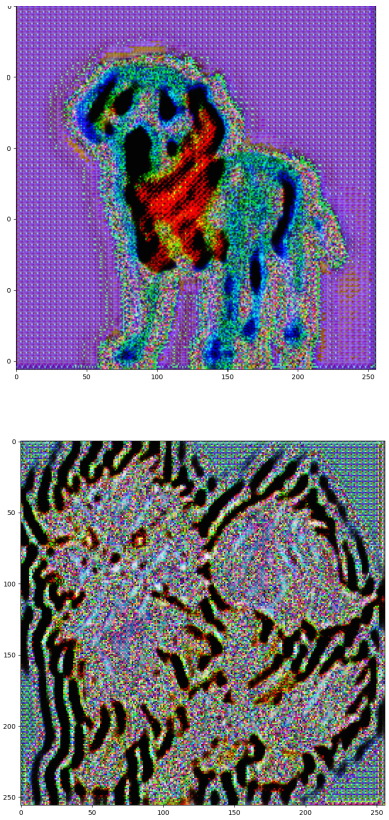

Figure 2. Base Model Output(10000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

7. Summary of Experiments

I performed initial tests on different combinations of my methods and concluded that the Class Loss and Retention Loss were not worth including in further experiments. I then trained 4 models for 10000 steps, focusing on improving the baseline model using Diff Loss and Grayscaling. These experiments were greatly successful and performed quite positively. For my risky experimentation, I trained my modified classifier CycleGAN architecture with Grayscaling and Diff loss for 4000 steps(due to time and resource constraints) and compared it to the base model with Grayscaling and Diff loss trained for the same number of steps. While I am unfortunately unable to see this model perform after extensive training, its early results show pleasantly positive outputs. This is an excellent result for such a risky experiment and is worth further investigation.

8. Analysis

The baseline model(Figure 1) produced extremely unsatisfactory results, performing very little change to the images and producing a heavily blurred quality with a distorted ef-

Figure 3. Base Model w/ Diff Loss Output(10000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

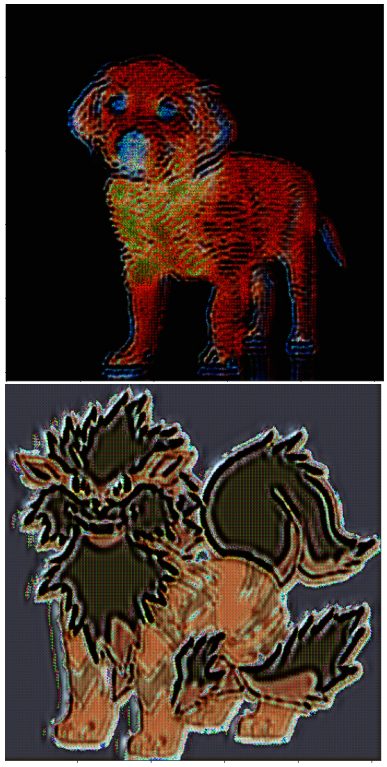

Figure 4. Base Model w/ Grayscaling Output(10000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

fect on it background. Likely this is due to the model's attempts to replicate the animation style of the anime by placing sharp patterns in the background. The Diff Loss example(Figure 2) made small strides toward geometric feature changes. In the generated dog, it reshapes the mouth of the Pokemon, such that you no longer see the small teeth and makes it seem more like an open mouth of a dog. Its dog to Pokemon conversion adds a lined pattern which masks the realistic fur of the dog.

The Grayscaling example(Figure 3) produced higher quality outputs which did not necessarily have as drastic changes but seemed less distorted. Notably, it rounded the lines of the fake dog's frame to make it less cartoon like and sharpened the fake Pokemon's frame for the opposite effect. It also attempts to the reduce the prominence of the Pokemon's stripes in it fake dog output.

The combined methods(Figure 4) produced results which retained benefits from both. The Pokemon to dog conversion once again added the open mouth to the fake dog as well as beadier dog eyes. In the dog to Pokemon conversion, the generator added clearly bounded patches of color onto the dog's body, much like a Pokemon's design.

While the combined methods were more effective overall and produced outputs with the a balance of quality and drastic changes, the combined method also failed at things its components succeeded at, such as reducing the prominence on the fake dog' stripes.

The results of these methods on the fake dog image have been compiled in Figure 8 for easy comparison.

Similarly the results of these methods on the fake Pokemon images have been compiled in Figure 9.

In my further experiments which highlights the classifier model, the output of the classifier model(Figure7) is comparable to the base model(Figure 6) with the same number of steps in quality. This is a pleasant result for a novel architecture as it has not reduced the quality of the model. However, notably, the classifier model after 4000 steps seems to have produced a Pokemon with more defined outline and more solid patches of color than the base model after a similar number of steps. Upon reference to Figure 5, it seems that Figure 7 has identified similar features and made similar changes. Interestingly, it has done this more than Figure 6, which is the same model as Figure 5. While this is an unverified hypothesis, I would speculate that the Classifier model is able to converge more rapidly than the Base model.

9. Conclusion

While we have failed to see significant geometric change using the CycleGAN architecture, we have made very small strides toward improvement. While this could likely be

Figure 5. Base Model w/ Grayscaling and Diff Loss Output(10000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

credited to shortcomings in my method and may be possible through some careful tuning of parameters and architecture, the objective of geometric change may be out of reach for CycleGAN. Beyond geometric change alone, this translation task is also quite tricky due to its disagreement with the underlying concept behind unpaired translation. That is, unpaired translation preposes some underlying mapping between the classes. For Pokemon which are only loosely inspired by animals, there likely does not exist such a mapping between them and dogs. This is unlike the horse to zebra problem where there is a far more realizable strategy for mapping them to each other.

However, for a difficult unpaired image translation application, with a very limited dataset, my proposed Diff Loss and Random Grayscaling methods produced arguably higher fidelity outputs than the baseline architecture. Whether or not they have potential for geometric changes, these methods are worth further study for usage in difficult image translation tasks.

Finally, with very little study, my proposed Classifier CycleGAN architecture model shows potential as a more effective variation on the standard CycleGAN architecture.

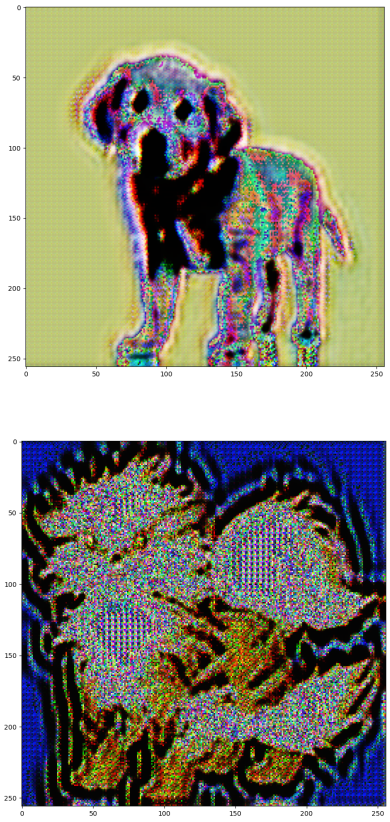

Figure 6. Base Model w/ Grayscaling and Diff Loss Output(4000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

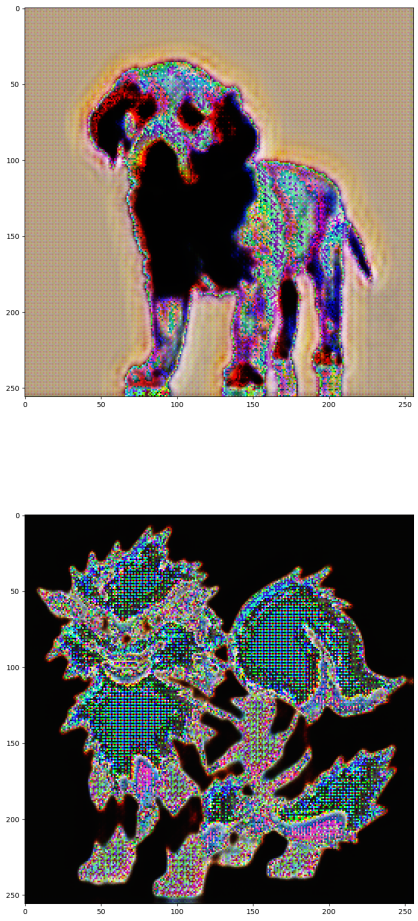

Figure 7. Classifier model w/ Grayscaling and Diff Loss Output(4000) -
top: Golden Retriever(fake Pokemon), bottom:Arcanine(fake dog)

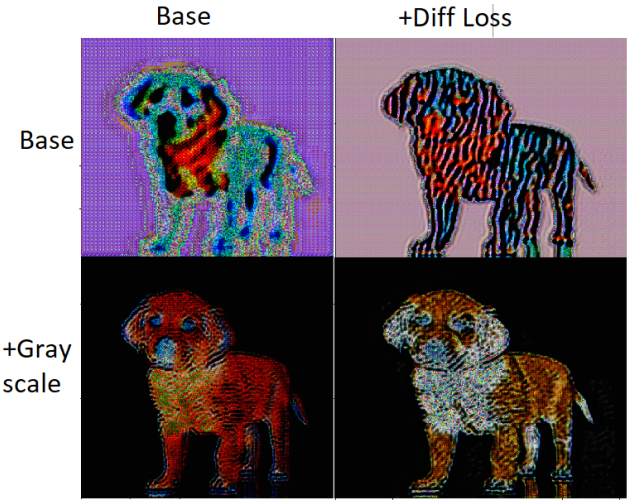

Figure 8. Grid showing results of adding Diff Loss and Grayscaling to the baseline model on generated pokemon

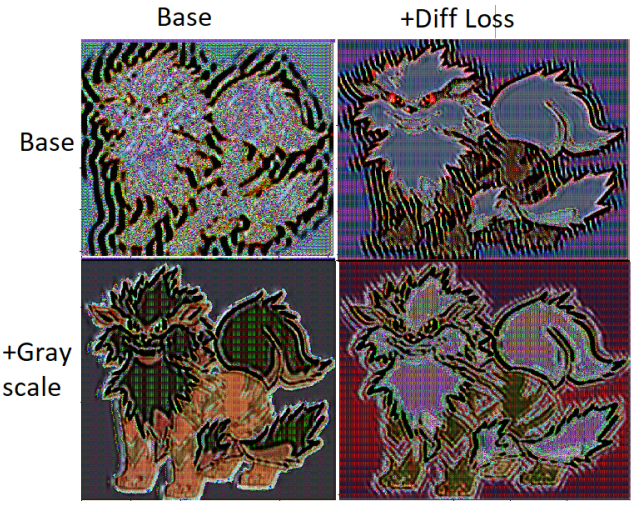

Figure 9. Grid showing results of adding Diff Loss and Grayscaling to the baseline model on generated dogs