EvolPokéGAN: Final Report

G054 (s1887493, s1871044, s1890135)

Abstract

We have adopted pix2pix to model the evolution pattern of Pokémon using in-game display images and full-body anime images. Generated samples are compared with samples from baseline models. Baseline models are DC-GAN and Conditional DC-GAN modelling visual pattern of Pokémon images directly. Since pix2pix model is specifically designed for surface appearance translation tasks, several modifications were designed to enable this model to capture structure transformation patterns.

The overall results indicate that samples from models trained on evolution datasets have more concrete shapes, solid colours and sharper detailed structures. However they still can not be considered as valid Pokémon. Mode collapse was encountered during the training on in-game display images dataset using the original training process in pix2pix project (Isola, 2017).

1 Introduction

GAN is the biggest breakthrough in the recent years of Artificial Intelligence (Sagar, 2018). As quoted by the celebrated computer scientist Yann Lecun, "GANs and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion." There is a tremendous growth in the application of GANs and one application is to generate Pokémon.

Numerous attempts like (PokeData, 2018) and (Han, 2018) have produced unsatisfactory results. They pointed out that the task of creating Pokémon is challenging due to the high variance between Pokémon. However, we believe that this variance is much smaller between evolution patterns and therefore, generating Pokémon based on evolution would produce better results.

We have initially documented in our interim report that we would like to model the evolution pattern of animal-like Pokémon in a detailed scope. However, we realise that this task is quite similar to the first task. The dataset will also pose a greater challenge as the current challenging dataset will shrink. Thus, we have decided not to research on that task. Therefore, we are going to research on the benefits of using the Pokémon type as conditions in the Conditional-GAN. We will also use different dataset to investigate if different styles of the same Pokémon have an impact on the GAN's performance. Lastly, the main research question of this paper will then be to verify if Pokémon evolution serve as the better pathway of generating new Pokémon.

2 Related work

Generative Adversarial Networks GANs are a class of generative modes that are based on game theory. It comprises of two models that pit against each other. A generator model tries to produce realistic fake samples, while the discriminator model tries to distinguish between real and fake samples. The GANs were introduced by Ian Goodfellow and other researchers from the University of Monreal in 2014 (Goodfellow et al., 2014). GANs' potential has been popularly studied due to its ability to mimic any distribution of data. One such example is the generation of Pokémon (PokeData, 2018). The GAN used by PokeData uses the dataset containing full-body images but does not consider learning the features that change during evolution.

Conditional Generative Adversarial Networks In basic generative adversarial networks, there is no constraint on the modes of data being generated (Mirza & Osindero, 2014). For example, if you generate an image of a dog using GANs, the generated image can be a running dog or a sleeping dog. This paper (Mirza & Osindero, 2014) proposed Conditional Generative Adversarial Networks. In this network, both generative model and discriminator model are given additional information called "conditionals". By giving conditions to the network, we can direct the network to generate a certain kind of data we want. In the paper, the author implements a conditional GANs to generate hand-written digits based on given class labels. Instead of generating a random digit, the system will generate a hand-written digit of "1" if the class label "1" is given as the condition.

3 Data Set and Task

Data Sets This work will use the images of Pokémon from the original Nintendo game series as the basic dataset, these images can be acquired from the official website which are free to use. There are images of different styles that can be found online, from which we choose two kinds of size and styles as datasets in this work: In-game display images, and painting style Full-body images for Anime use. The In-game display images are obtained from Pokémon Database. The painting style full-body images can also be obtained from the same website, but we chose to extract them from Pokédex as this website provides higher resolution images and would be beneficial for our networks. The

In-game display images and Full-body images are colour images and are of size 120×120 pixels and 475×475 pixels respectively. Since they differ in resolution and style, they should be used separately to train different models. Figure 1 displays the different styles of Pokémon images used.

Figure 1. Left: In-game image ; Right: Full-body image

For this work, we only use the Pokémon from Generation 1 to 6 as the in-game display images of Generation 7 Pokémon obtained from Pokémon Database have different colour backgrounds. Moreover, the patterns of Generation 7 Pokémon are too complex to learn. Thus, to enable comparison across the data sets, all the data sets will contain images of Pokémon up to Generation 6. Hence, the total size of each dataset is 721.

Figure 2. General evolution pattern

Data Preprocessing To model the evolution of Pokémon in Task2: Model the Pattern of Pokémon Evolution, we need to have a training set with pre-evolved Pokémon as training inputs and evolved Pokémon as training targets. Although most Pokémon evolution follows a similar pattern examples as shown in Figure 2 (similar shape & colour, bigger body, sharper edges, larger wings or larger horns etc), there are still some Pokémon which do not evolve to a similar shape. As Figure 3 shows, Caterpie has a completely different shape to Metapod. Butterfree even has a totally different body colour. This kind of evolution pattern will not be considered as a valid in our work.

Figure 3. Caterpie evolves to Metapod which then evolves to Butterfree. This evolution path do not follow the regular evolution pattern and thus can not be considered as a valid evolution sample.

Thus, the whole Pokémon dataset is divided into two sets. Those Pokémon that have a valid evolution path will be grouped into the training set. The remaining Pokémon will be grouped into the testing set. The training set and testing set of each data set will then have 596 and 125 images respectively.

For convenience in pairing images into training samples, each valid pair of evolution are concatenated into one long image with the pre-evolved Pokémon on the right and the evolved version on the left so that each training sample is a pair of evolution. During training, data loader will crop the samples into inputs and targets. Figure 4 displays an example of a training sample. Hence, each training set now has a total of 290 training samples.

Figure 4. Training Sample

Data Augmentation As mentioned in the Experiments, the main model trained on the In-game display images generates the most credible Pokémon. As such, data augmentation was conducted only on the dataset containing In-game display images to see if there is any positive impact on the model. Data augmentation is a process where a copy of the original image is taken from the dataset and it is transformed into another image through some combination of processing. Example of processing includes flipping and rotating images. Thus, the data set is artificially increased and this will help the model to better learn the features in the images.

Thus, we have chosen to augment the data by flipping and rotating the images in both clockwise and anti-clockwise direction by five and ten degrees. With this process, our data set has increased by 6-fold.

Task1: Improve Baseline Models with Conditional–GAN. Before carrying out the experiments modelling the Pokémon evolution, we need to set up two baseline experiments each with a different model. One with Deep-convolutional-GAN (Radford et al., 2015) to model the visual pattern of Pokémon directly, the other with Conditional-GAN (Mirza & Osindero, 2014) using types of Pokémon as training labels. As we couldn't find any previous work to model Pokémon evolution, our baseline experiments will only model the visual pattern of Pokémon directly. The final outputs will be visually compared to see which model can generate more reasonable Pokémon.

The first baseline experiment aims to reproduce the project PokeGAN (PokeData, 2018) which inspired our work, We only aim to achieve a similar quality of outputs as PokeGAN.

The second baseline experiment aims to achieve a better performance than PokeGAN as the model will learn the visual pattern of different types of Pokémon separately. It can be seen from the dataset that the shape and colour of Pokémon usually depend on their type, one type of Pokémon tend to have similar colour and features. Thus it can be expected that this experiment will have better performance. Our model for Pokémon evolution will also be a Conditional-GAN with pre-evolved Pokémon as input label so that it will be more sensible to use another Conditional-GAN

Task2: Model the Pattern of Pokémon Evolution. We will train a model on the dataset containing the images of all evolution pairs for each Pokémon. The outcome should be a model which can generate an evolved form for any Pokémon based on the evolution pattern learnt from the training set. Even for those Pokémon do not have any evolution in the original game setting, the model can generate a reasonable image of evolved Pokémon based on the input image.

During testing, the trained models will try to generate evolved samples from test inputs, because those generated samples will have no target to compute loss, they will then be visually checked to see if they are valid Pokémon and if they follow the general evolution pattern.

4 Methodology

The main model used in this work will be a combination of Conditional GANs (Mirza & Osindero, 2014) and Deep Convolutional GANs (Radford et al., 2015). These two techniques are both developed on the concept of the original Generative Adversarial Networks (Goodfellow et al., 2014).

GANs GAN is a deep learning model that train two networks competitively together, one as generator G to generate fake samples from random noise, another as discriminator D to tell if the fake samples are real (e.g. how similar they are to the target samples). This concept works well for generative tasks as the generator map inputs to the probability distribution of all real samples rather than a single target.

The objective of GAN training is a min-max game between the network D and G as presented by equation 1, in which p_z denotes the noise distribution where the random noise input x is sampled from; p_{data} denotes the target distribution where the real data y come from. During training, D net tries to distinguish fake samples from real ones resulting maximising V(D, G), while G net tries to generate realistic samples to fool the D net resulting minimising V(D, G).

$$\begin{aligned} Objective &= arg \min_G \max_D V(D,G) \\ &= E_{y \sim p_{data}(y)}[log(D(y))] \\ &+ E_{x \sim p_x(x)}[log(1 - D(G(x)))] \end{aligned} \tag{1}$$

C-GANs On top of GANs, Conditional GANs provide additional labels z to both networks of the original GANs. During training, the D net will calculate loss according to the conditional labels, so that the G net will be forced to generate specific samples according to the given conditions as well. The new objective function can be represented as equation 2, where z is the conditional label.

$$\begin{aligned} Objective &= arg \min_G \max_D V(D,G) \\ &= E_{y \sim p_{data}(y)}[log(D(y,z))] \\ &+ E_{x \sim p_x(x)}[log(1 - D(G(x,z),z))] \end{aligned} \tag{2}$$

DC-GANs Deep Convolutional GANs is another variation of GANs, it adopted the idea of Convolutional neural networks to both generator and discriminator networks of original GANs so that it will have better performance and efficiency on image generative tasks. The first baseline experiment mentioned in Task1: Improve Baseline Models with Conditional-GAN will use this concept to model the visual pattern of Pokémon images directly.

By combining the concept of Conditional GANs and Deep Convolutional GANs, we aim to adopt a model that can take the image of pre-evolved Pokémon as conditional labels and evolved ones as target data to generate realistic images of Pokémon that can be considered as an evolution of input labels.

Pix2pix: an image to image translation model The work published by Isola et al. (2017) with a released PyTorch implementation called pix2pix (Isola, 2017) provides us with an ideal solution to our desired application. The pix2pix model is a GAN with convolutional layers in both networks and takes images as input labels to translate them to target outputs. What differs from the conditional GANs mentioned in the previous paragraph is that pix2pix only use an image label as input to the G net, the noise is given in the form of dropout (Srivastava et al., 2014). This is because the noise inputs in conditional GANs are usually ignored and given zero weight by the model (Mathieu et al., 2015).

They aim to apply this model to a variety of image to image translation problems such as Map to aerial photo, edge to photo and grey-scale image colouring. But most of these are modification on surface appearance of images while keeping the overall structure, this is different from our application whose main point is structure transformation, but the network architecture can still be applied to our task with a little modification, which will be discussed later.

Generator of Pix2pix model The Generator networks used in most of image translation works (Pathak et al., 2016; Wang & Gupta, 2016; Johnson et al., 2016) use a encoder-decoder architecture (Hinton & Salakhutdinov, 2006) with a stack of down-sampling layers and another stack of up-sampling layers connected by a bottleneck layer in the middle. Based on this, pix2pix further adopted the concept of U-net (Ronneberger et al., 2015) which adds skip connections between down-sampling and up-sampling layers, the overall structure can be seen as right part of figure 5. These connections will concatenate channels at down-sampling layers to corresponding up-sampling layers, through which the output data can be more related with input data. They choose this structure in order to transfer more low-level information from input (such as locations of pixels of major edges in the case of image colouring) to output.

However, the skip connections may offer different functionality in this work. We aim to model the transformation of image structure, the relationship of low-level features between inputs and outputs are much weaker than surface appearance translation, these skip connections could bring too much constrains on the shape of outputs. But on the other hand, we will want the high-level features like the overall colour and type features (like fire or leafs) of Pokémon to be transferred to the output, the skip connections could help to do this. Thus, we should test both the original encoder-decoder architecture without skip connections and the U-net architecture on the Pokémon evolution task.

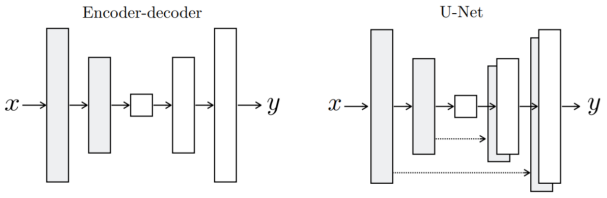

Figure 5. Generator architectures, Left: Encoder-decoder; Right: U-net (Isola et al., 2017)

Discriminator of Pix2pix model Because pix2pix model aims to solve image surface appearance translation, they do not need to model losses on overall structures. In order to make the model focusing on style and texture loss, the Discriminator only calculate losses in a patch domain of the whole image by conducting real or fake classification on each N*N patch. In this case, the image is treated as a Markov random field, pixels in different patches are assumed to be independent (Li & Wand, 2016). Similar assumption can also be found in other texture and style modelling works (Gatys et al., 2015; 2016). This design can also reduce the number of parameters required in the Discriminator by giving a small N while keeping results at a high quality.

For our task, this Discriminator focuses on a complete different direction. In order to model the loss on image structure, the dependence between pixels must not be ignored, even they are distant on the image scale. In order to keep the dependence between pixels spanning the whole image, losses should be computed on the full image domain, even this requires more parameters than their design.

Optimiser of Pix2pix model The pix2pix optimise its networks in the standard way proposed in the original GAN (Goodfellow et al., 2014), the optimisation keeps switching each step between Generator network and Discriminator network, while the step size performed on Discriminator is divided by 2 so that the training rate of Discriminator is slowed down compared with Generator. Minibatch SGD and the Adam learning algorithm (Kingma & Ba, 2014) was applied in the optimiser, with a learning rate of 0.0002, and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$ for the Adam algorithm.

During testing, dropout is applied to the generator and batch normalization (Ioffe & Szegedy, 2015) is applied using the statistics of the test batch. When the batch size is set to 1, this way of applying batch normalization is proposed as "instance normalization" by Ulyanov et al. (2016).

The optimiser used in pix2pix model has not been specifically designed for surface appearance translation, and most of it follows the standard training process from original GAN, it should works fine for our task requiring no more modification.

5 Experiments

The baseline of our model is based on the project PokeGAN (PokeData, 2018) which inspired our work. That is a original GAN developed on TensorFlow, trained on the whole Pokémon dataset including in-game display and full-body images, to generate new Pokémon from random noise. By giving specific subset of the whole dataset, the model can be re-trained to generate Pokémon of specific type or colour.

5.1 Deep Convolutional GANs

Deep Convolutional GANs(DC-GANs) works better than the vanilla GANs on the image generation tasks(Radford et al., 2015). We carry out this baseline experiment to train a DC-GAN model to generate new Pokémon. We aim to see how well a DC-GANs works on the Pokémon data set.

Model Structure Instead of using fully connected layers for both discriminator and generator networks, a DC-GANs will apply convolutional layers for the discriminator and

transposed convolutional layers for the generator. In addition, each of these layers is followed by a batch normalization except for the input layer of disciminator and the output layer of the generator (Radford et al., 2015). For the activation, the discriminator will use leakyReLU and the output layer will use sigmoid. And ReLU will be used for the generator except for the output layer which uses tanh as activation (Radford et al., 2015). The generator structure that we have described above is shown in Figure 6. The structure of discriminator is similar except for using convolution and different activations for each layer. Our baseline DC-GAN model uses the same structure as the model we describe above.

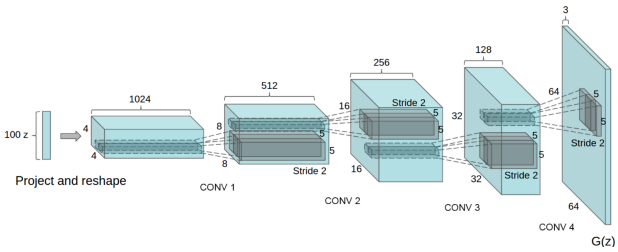

Figure 6. The generator structure of a general DC-GAN (Radford et al., 2015)

Hyperparameter Settings Basically, we set the hyperparameters to be the values as suggested by the paper (Radford et al., 2015), except for the batch size. We use a smaller batch size of 64 since our data set is relatively small. And by conducting the experiments with those settings, we get better results when a batch size of 64 is used. All the hyperparameter settings are shown in Table 1.

Image size	64
Number of channels	3
Size of generator input	100
Size of feature maps in generator	64
Size of feature maps in discriminator	64
Learning rate for optimizers	0.0002
Beta1 hyperparamter for Adam optimizers	0.5

Table 1. Hyperparameter settings for our DC-GANs model.

Results We conducted experiment on both data sets including In-game display images and painting style Full-body images. The best results are shown in Figure 7 and Figure 8. The generated Pokémon are not acceptable. The details of the generated Pokémon are vague. However, the model successfully stimulates the shape and colour scheme of Pokémon and the fourth Pokémon in the first row of Figure 7 has a huge resemblance to the Pokémon Pikachu. Comparing the results between the Figure 7 and Figure 8, the former has better shapes and details. This indicates that this model works better on the lower resolution images since the higher resolution images contain more details which is hard for the model to learn.

Figure 7. Result for DC-GANs model on the in-game display images data set.

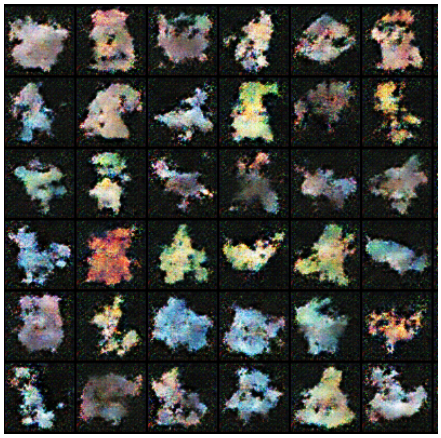

Figure 8. Result for DC-GANs model on the painting style full-body images data set.

5.2 Conditional DC-GANs

The Pokémon are generated randomly by the DC-GAN model. The model does not know what the types of generated Pokémon are. However, the type of a Pokémon is very important in modelling the Pokémon evolution. Based on our DC-GAN Model, we modify the model to be a Conditional DC-GAN. The new model will generate new Pokémon based on the given types of the Pokémon. The Conditional DC-GAN has the same model structure and hyperparameter settings as the previous DC-GAN model. The only difference is that both the discriminator and generator require an additional input, which is the conditions.

Results The results are shown in Figure 9 and Figure 10, and each row belongs to different types. Comparing with the results of the DC-GAN model, the Conditional DC-GAN does not improve the quality of generated Pokémon, but the results have a slightly higher resolution. In addition, the result of using the in-game display images (Figure 9) is

still slightly better than those of the Full-body images. This indicates that the In-game display images data set is more suitable for our tasks. As shown in Figure 9 and Figure 10, each row of Pokémon have similar shapes as compared with the results of the previous DC-GAN model. In addition, some of them have common colour scheme. By adding the types of Pokémon as the conditions to the model, we can direct the model to generate certain types of Pokémon. In the future experiments, we can apply conditions to model to learn how a Pokémon evolves since different types of Pokémon have different evolution patterns.

Figure 9. Result for Conditional DC-GAN model on the in-game display images data set.

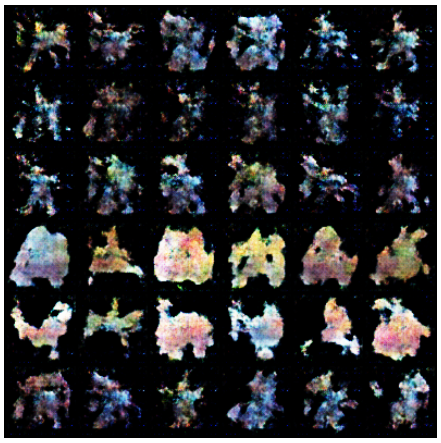

Figure 10. Result for Conditional DC-GAN model on the painting style full-body images data set.

5.3 Modelling Pokémon evolution with Pix2pix model

We adopt pix2pix model to conduct Task2: Model the Pattern of Pokémon Evolution to model the Pokémon evolution pattern. Most experiments carried out in this work only used the original pix2pix model without any modifications mentioned in sections Generator of Pix2pix model and Discriminator of Pix2pix model. Due to time constraints, we are unable to implement all modifications and the heavy traffic experienced on the GPU cluster provided did not supply the computational resources required to test the new model thoroughly.

The training process of pix2pix model in the experiments follows the default setting in the GitHub project (Isola, 2017) and the datasets are loaded with a batch size of 1. The model is trained on the original datasets with 200 epochs and the augmented datasets (which is 5 times larger) with 100 epochs. The first half of the total epochs will use a initial learning rate of 0.0002 and the rest will have a learning rate that decays linearly to zero.

The first two experiments trained pix2pix models on the original Full-body images dataset and In-game image dataset respectively. A subset of the test output samples can be seen in Figure 11 and 12.

Pix2pix on Full-body dataset As results shown in Figure 11, samples generated by the model trained on Full-body images dataset are rather poor, with some failing to construct a continuous body shape. The inner details are blurry and some features can only be recognised from there.

The reason can be concluded from the properties of the Full-body dataset. As the images in this dataset are used for anime, Pokémon in this dataset are mostly more expressive, with complex posture and portrayed in different angles. Some of the Pokémon also have a back view in the images. These complex properties make it harder for the model to find corresponding features within a pair of evolution. Another problem of this dataset is that the size of a Pokémon is not taken into consideration. Most of the Pokémon have the same size on the image, and this prevents the model from learning the size changes from evolution.

Pix2pix on In-game display dataset The model trained on In-game display images dataset gives relatively more reasonable output samples. It can be seen from Figure 12 that these samples have more concrete body shapes and sharper details inside. Most of the samples are getting bigger through evolution, and some of them even have recognisable type features like fire, leaf and wings.

In contrast of the Full-body image dataset, images in this dataset are more formatted as they are used in the game to illustrate each Pokémon. Thus, all Pokémon have similar standing posture and are facing to the left. Furthermore, these images have large blank area which makes the size difference of Pokémon more clearly. However, these output samples still can not be considered as valid Pokémon as most of them do not follow any animal shape and the inner details do not form any edges of body parts.

One observation in these output samples is that a large number have the same structure in the centre of the image. Regardless of the inputs, this structure exists with different colour. This problem is termed as mode collapse by Good-

fellow et al. (2014). Mode collapse refers to a commonly encountered failure case for GANs, where the generator learns to produce samples with extremely low variety. The main reason for this problem is that the Discriminator of GANs in most cases is easier and faster to train than the Generator. This makes it capable of telling fake samples from real ones in an early stage when the Generator can only produce a few possible samples. When this occurs, other attempts of the Generator will all be rejected. Due to the structure of Discriminator in pix2pix (described in Discriminator of Pix2pix model) that only classify in small patch domain, mode collapse is more likely to occur in small patches in this case. A possible solution to this is to make the training of Discriminator slower but this will require us to conduct more experiments in a fine tuning way which we are currently unable to do.

Pix2pix on augmented In-game display dataset In order to improve the quality of the output samples, a new model is trained on an augmented In-game display image dataset in this experiment. The augmentation include horizontal flip, rotation of 5 and 10 degrees in both clockwise and anti-clockwise direction. By doing augmentation, we increase the size of the dataset and provide the model with more samples to learn the evolution pattern. Basically this is still the same dataset with 5 time the size. To train the model on this dataset, only half of the epochs will be enough. Not only the fully trained model but also the checkpoints during the training will be stored and tested after the training.

Part of the test samples from the model trained with 50, 55, 100 epochs are listed in figure 13. Most samples are still far from valid Pokémon and no significant improvements can be observed from the augmentation. An observation from the change between different epochs is that the shape of generated Pokémon is getting more concrete and the colour in each part is getting solid. But mode collapse persists in this experiment as first two columns in figure 13 still have similar structure in the centre.

Another observation worth mentioning is about the output samples which uses Eevee (up-left corner in figure 11,12 and left column of figure 13) as an input. In the settings of the game, Eevee has multiple evolution to different types, and these evolution are all included in the datasets. However, each model can only generate one output sample with low variety in the test set. This verifies that the pix2pix model is not capable of generating stochastic output, as stated by Isola et al. (2017). This is still an open question to current works on conditional GANs.

Comparing with the results of previous baseline experiments, the output images of Pix2pix model have clearer shapes and colour. The baseline models only capture the low-level features of Pokémon, such as rough shapes and colour schemes. However, as shown in Figure 11 and Figure 12, the Pix2pix model captures the high-level features of Pokémon, such as the head, feet and wings of Pokémon. For example, you can observe that the second sample in the first row of Figure 11 is a flower. The toes can also be seen in the first sample of the third row in Figure 11.

Figure 11. Output samples of pix2pix model trained on **full-body** image dataset, left of each pair is pre-evolved input and right is generated evolved sample.

Figure 12. Output samples of pix2pix model trained on **in-game display** image dataset, left of each pair is pre-evolved input and right is generated evolved sample.

6 Conclusions

This work mainly focuses on modelling Pokémon evolution using pix2pix model. Three experiments were carried out to train pix2pix model on different datasets of paired Pokémon evolution images, and two baseline experiments were conducted using DC-GAN and Conditional DC-GAN respectively to model visual pattern of Pokémon images directly.

The overall results indicate that samples from models trained on evolution datasets have more concrete shapes, solid colours and sharper detailed structures, although they still can not be considered as valid Pokémon. The last two experiments training models on in-game display datasets encountered mode collapse, training of Discriminator should be further slowed down to cooperate with generator to deal with this problem.

Several modifications were designed to enable pix2pix

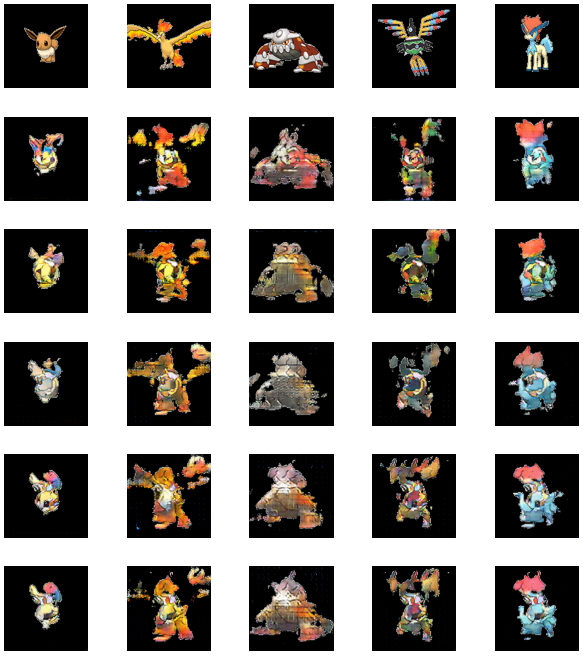

Figure 13. Output samples of pix2pix model trained on **augmented in-game display** image dataset for different number of epochs, left of each pair is pre-evolved input and right is generated evolved sample. **From top to bottom each row presents samples from inputs, and generated samples from models trained for 30, 50, 70, 90, 100 epochs**

model to capture structure transformation pattern in images, but due to the limitation on time and heavy traffic on provided GPU cluster, they can not be implemented in this work. Further works should investigate the influence from those modifications and try to find a better model design based on pix2pix.

References

Gatys, Leon, Ecker, Alexander S, and Bethge, Matthias. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pp. 262–270, 2015.

Gatys, Leon A, Ecker, Alexander S, and Bethge, Matthias. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Han, Steven. Pokemon generation. https://medium.com/pokemongan/pokemon-generation-77bd5c2ca6fa, 2018.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Isola, Phillip. pix2pix. https://github.com/phillipi/pix2pix.git, 2017.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, Chuan and Wand, Michael. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pp. 702–716. Springer, 2016.

Mathieu, Michael, Couprie, Camille, and LeCun, Yann. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Mirza, Mehdi and Osindero, Simon. Conditional Generative Adversarial Nets. *arXiv e-prints*, art. arXiv:1411.1784, Nov 2014.

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Pathak, Deepak, Krahenbuhl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

PokeData. Pokegan. https://github.com/PokeData/PokeGAN.git, 2018.

Radford, A, Metz, L, and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. arxiv preprint arxiv: 151106434. 2015.

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Sagar, Ram. Why gans are the biggest breakthrough in the history of ai. https://www.analyticsindiamag.com/gans-biggest-breakthrough-in-ai/, 2018.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Ulyanov, Dmitry, Vedaldi, Andrea, and Lempitsky, Victor. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Wang, Xiaolong and Gupta, Abhinav. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pp. 318–335. Springer, 2016.