Final Report For COMP4471 2023 Fall- Creating Medium Resolution Image Of Pokémon By Conditional Deep Convolutional Generative Adversarial Net With Multi-label Classifier

Wong Erastus
20522468
HKUST
ewongac@connect.ust.hk

Abstract

Generative adversarial net [4] (GAN) generates random samples based on random noise in the latent space. To create a specific type of sample, we propose a new variation of the conditional generative adversarial net [12] (cGAN), which uses a multi-label classifier as the discriminator to create conditioned samples. By feeding random labels to the generator and comparing the output of the discriminator to this input, we wish to train a conditioned generator. Pokémon images contain many attributes simultaneously for each species, which are challenging to capture. To show our work is practical, we compare generated images of Pokémon using GAN, CGAN, and our proposed method. We show that our method converges more easily in our settings and can capture the given label to the pixel space. We also perform tests on different settings of our proposed method and try to achieve the best result within the project span.

1. Introduction

GAN is a generative framework that tries to synthesize samples according to the distribution of the training data. One of its major application domains is image generation.

Pokémon is a Japanese franchise of a huge collection of imaginary creatures with supernatural abilities, classified by types, forms, and more attributes. First released in 1996 with 151 Pokémons, the franchise has grown to the ninth generation. According to the official Pokédex [1], there are now officially 1010 different species of Pokémon in total.

Numerous projects [17] [14] [16] [1] [8] [2] have tried to synthesize Pokémon images through GAN, but they used lower resolution $(32px - 128px)$ and the results are not satisfying. More importantly, these attempts all create Pokémon by purely random noise without any conditioning.

While cGAN demonstrated an extension to GAN to specify the condition of generated images, there are some limitations of the method. Since the condition being trained corresponds to the label of the real data set, the behavior can be unexpected when we want to synthesize new species of Pokémon with non-existing combinations of attributes. An extra layer is needed to transform the labels to fit the image as inputs, increasing the difficulty of training.

This project [2] proposes an alternative method of training feature-specific images through GAN. We compare the results of our method with GAN and cGAN to demonstrate the improvement of our method over traditional methods. In the experiments, we generate $512px$ Pokémon images and evaluate the models by comparing the generated images with the corresponding training data and comparing the training loss.

2. Related Work

2.1. Generative Adversarial Nets

The architecture of GAN consists of two neural networks, a generator G and a discriminator D. D is a binary classifier trained to distinguish whether the image is from the data set and G is trained to fool D by creating samples close to real data. In each batch of the training, D is fed with both real samples $\boldsymbol{x}_{image}$ and $G(\boldsymbol{z}_{noise})$ and G is fed with $\boldsymbol{z}_{noise}$. By careful calibration, this mini-max system converges to a stage where G approximately maps the noise distribution $p_{\boldsymbol{z}_{noise}}$ to the real data distribution p_{data}, and D is completely fooled such that the accuracy converges to

[1] https://www.pokemon.com/us/pokedex

[2] The source code of the project can be found on https://github.com/w-era/multilabel_pokegan

0.5. Since GAN uses binary cross-entropy loss, we optimize

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}_{image} \sim p_{data}}[\log D(\boldsymbol{x}_{image})] + \mathbb{E}_{\boldsymbol{z}_{noise} \sim p_{z_{noise}}}[\log(1 - D(G(\boldsymbol{z}_{noise})))]$$

2.2. Deep Convolutional Generative Adversarial Nets

Deep convolutional generative adversarial net (DCGAN) [15] is an extension of GAN utilizing convolutional layers. Both G and D are composed of purely convolutional layers, where G uses the transposed convolutional layers to increase the dimensions. Since convolutions focus on the spatial relation of data, DCGAN works better for visual tasks. This work also gives general suggestions on training DCGAN. In the project, we will use the DCGAN, mostly following the paper's suggestion, instead of vanilla GAN to boost performance.

2.3. Conditional Generative Adversarial Nets

cGAN extends GAN by feeding additional labels $\boldsymbol{x}_{attributes}$ as inputs for both G and D. The optimization problem then turns to

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}_{image} \sim p_{data}}[\log D(\boldsymbol{x}_{image}|\boldsymbol{x}_{attributes})] + \mathbb{E}_{\boldsymbol{z}_{noise} \sim p_{z_{noise}}}[\log(1 - D(G(\boldsymbol{z}_{noise}|\boldsymbol{x}_{attributes})))]$$

3. Data

3.1. Data Set

To evaluate the performance of our method, we perform image generation of Pokémon due to its nature in acquiring multiple labels simultaneously. We use the data set from https://www.kaggle.com/datasets/giovaniandrade/pokemon-images, which is generated by rendering in the Pokémon Home app 3.0.0 (Gen IX) [3]. It is a collection of 1527 512×512 transparent background images of every species and form of Pokémon. The data set also contains 11 categorical labels, which suit the purpose of verifying our method.

3.2. Data Pre-processing & Augmentation

All the missing values are first filled with zero. Then, we perform one-hot encoding to all data to facilitate learning. Since each Pokémon can have one or two types, we normalize the one-hot vector for the type. After encoding, there are a total of 54 dimensions in the label space. We also perform a random horizontal flip, a random rotation of $\pm 10°$, a random resized cropping with a scale between 0.85 to 1, and a normalization across each channel with $\mu, \sigma = 0.5$ as the augmentations to our data set. This is to address the issue of having only one image for each species of Pokémon.

3.3. Random Label Generation

Although we want to generate new combinations of attributes for Pokémon generation, we follow the following rules to avoid unreasonable combinations.

1. Either one or two types for each Pokémon
2. Either one or none of Mega and GMAX
3. Either one or none of Legend, Mythical, and Ultra Beast
4. Either one or none of the Future and Past Paradox
5. Exactly one shape and one color

4. Method

4.1. Our Method

We propose an alternative way to condition GAN in this project. Instead of using a binary classifier as the discriminator, we use a multi-label classifier to distinguish whether the image is real, as well as determine its label. Generator $G(\boldsymbol{z}_{noise}, \boldsymbol{z}_{attributes})$ generates $512px$ images where $\boldsymbol{z}_{noise}, \boldsymbol{z}_{attributes}$ are randomly generated. Our discriminator $D(\tilde{\boldsymbol{x}}_{image})$ is the multi-label image classifier producing $(t, \tilde{\boldsymbol{x}}_{attributes})$, where $t \in (0,1)$ is the probability of $\tilde{\boldsymbol{x}}_{image}$ being real, and $\tilde{\boldsymbol{x}}_{attributes}$ is the predicted label. Fig. 1 shows our proposed framework. Using loss $\mathcal{L}$, the optimizing objective becomes

$$\min_G \max_D (\mathcal{L}(D(\boldsymbol{x}_{image}), (0, \boldsymbol{x}_{attributes})) + \mathcal{L}(D(G(\boldsymbol{z}_{noise}, \boldsymbol{z}_{attributes})), (1, \boldsymbol{z}_{attributes})))$$

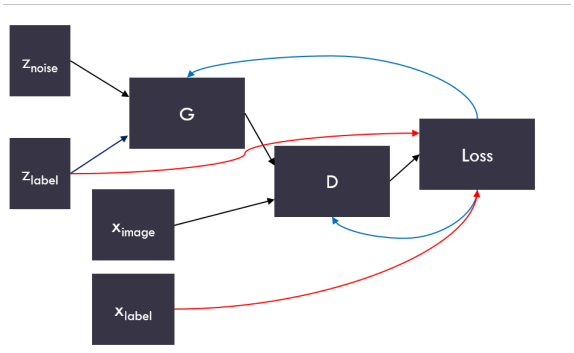


Figure 1. This is the proposed framework. Black lines indicate the input and output of networks, red lines indicate the target labels for the loss function, and blue lines indicate the gradients for updating networks.

In the initial setting, we use the weighted binary cross-entropy as the loss function with weight $\boldsymbol{w}$ to calculate the

loss for each batch N:

$$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N} -\boldsymbol{w}_i^T[\boldsymbol{y}_i \odot \log(\hat{\boldsymbol{y}}_i) - (\boldsymbol{1} - \boldsymbol{y}_i) \odot \log(\boldsymbol{1} - \hat{\boldsymbol{y}}_i)]$$

4.2. Architecture

The basic architecture Fig. 2 for the baseline models (GAN & cGAN) and our method is extended from DCGAN. The implementation is based on [6] [9]. The architecture of cGAN differs from GAN by adding an extra fully connected layer to the discriminator. Our method shares the same network architecture with GAN except for input and output dimensions.

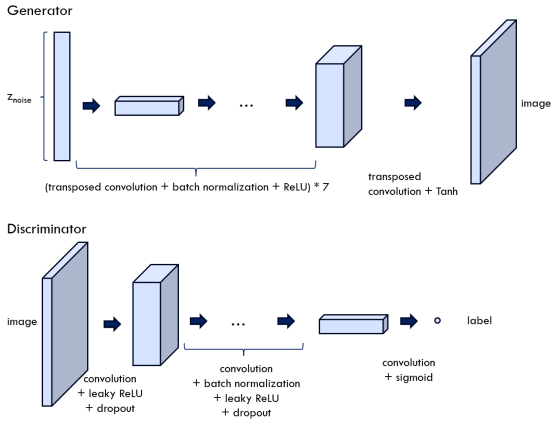


Figure 2. This is the basic architecture for our baseline models.

4.3. Initial Training & Hyper-parameters Settings

We initialize all the weights with Kaiming's method [5] and all the biases to be zero. In the initial settings, we take the following hyper-parameters and settings: A batch size of 32 with shuffling, a noise dimension of 64, a weighting of 5 for distinguishing real images and 1 for other labels, 0.0004 and 0.0002 learning rate with cosine annealing scheduling for G, D respectively, Adam [7] optimizer with $\beta_1 = 0.4, \beta_2 = 0.992$. The three models are trained with 100 epochs and 500 epochs each for evaluation.

5. Experiments

5.1. Basic Results - 100 epochs

As shown in Fig. 3, the training loss of our proposed model attains a smaller value and is more stable. Fig. 4 demonstrates the images generated by our proposed model are better in quality and are more diverse. They are also more similar to their corresponding real samples, illustrating the ability of our proposed model to capture labels. However, the rough quality suggests that more training is required.

5.2. Necessity On The Distinguishing Label

While the vanilla GAN analyzes features from the input image to determine whether the image is real or not, our model also captures features to assign attributes to the image. The information for distinguishing images may be encoded into the labeling process, such that only images close to the real ones can be correctly labeled. Is it true that we can use a multi-label classifier purely predicting the attributes of Pokémon to train a GAN? Unfortunately, this is not likely the case. We train our model with the slight adjustment, not distinguishing the image in the discriminator, for 100 epochs. Despite showing great convergence in Fig. 5, the generated image does not generate reasonable results. This result suggests the model learns irrelevant features that do not contribute to the generation of Pokémon without a distinguishing label.

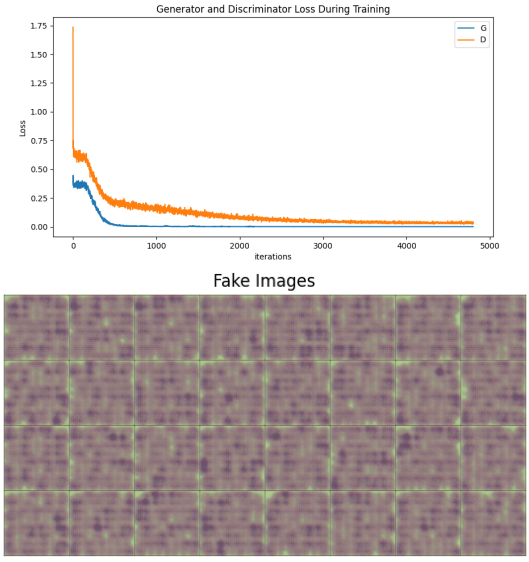
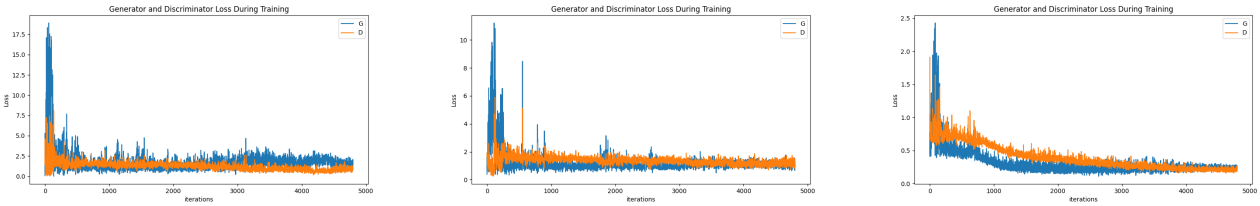


Figure 5. Results of the proposed model without the distinguishing label.

5.3. Resize Convolution

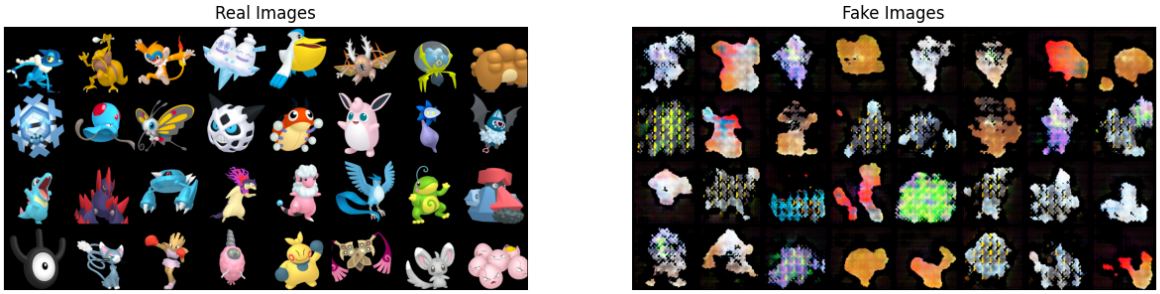
We can observe checkerboard patterns in our models, which is an issue due to the nature of DCGAN [13]. As suggested by the paper, we replace all the transposed convolutional layers with nearest-neighbour interpolations followed by a convolutional layer keeping the dimension. We train this modified version for 100 epochs. As shown in Fig. 6, while attaining good convergence, the attributes capturing ability is unsatisfying and the diversity is insufficient. Besides, although the checkerboard artifacts disappear, the model now gives bright or dark spots. Since convolutions capture nearby data to modify each element, it is mostly used as feature extractors to abstract raw representations.



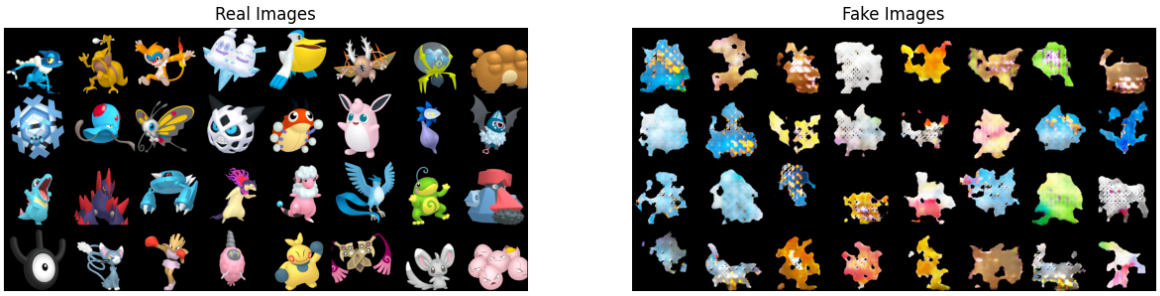
(a) GAN (b) cGAN (c) Proposed Model

Figure 3. Sec. 5.1: Training losses for the three models for 100 epochs.

(a) GAN



(b) cGAN



(c) Proposed Model

Figure 4. Sec. 5.1: Comparisons with real images for the three models for 100 epochs.

This suggests that resize convolution is more difficult to train for generative purposes.

5.4. LSGAN

As suggested by [10] [11], LSGAN is more stable in training and can generate images with better quality. We train a version using L2 loss instead of weighted binary cross-entropy loss for 100 epochs. The training result is shown in Fig. 7. Despite having stable loss, the result image quality is poor. It suffers from serious checkerboard artifacts and the image is much blurrier. It also does not capture the attributes well. We might need to spend extra effort to fine-tune the hyper-parameters since the gradients work differently from the cross-entropy loss.

Figure 6. Sec. 5.3: Results of the proposed model with resize convolution for 100 epochs.

Figure 7. Sec. 5.4: Results of the proposed model with L2 loss for 100 epochs.

5.5. Basic Results - 500 epochs

We train the three models again with the same settings with 500 epochs. Unfortunately, cGAN diverges and fails to generate sensible images and the losses of the other two models also do not converge well. As shown in Fig. 8, both models first go to low losses, then fluctuate in the middle of training, and finally converge to a higher loss than the 100 epoch version. Still, our proposed model has a significantly better result than GAN. Despite having higher quality images, Fig. 9 shows GAN is suffering from model collapsing [4] by having limited diversity. While our proposed model does not collapse, its label-capturing ability is significantly weakened in this trial. Therefore, fine-tuning is required to produce satisfying results.

5.6. Reducing Learning Rate

Since the proposed model first goes to a lower loss before fluctuating, we train a model with the learning rate of G, D being 0.0003 and 0.0001 for 500 epochs. Fig. 10 shows that this trial converges slightly better than the original one. It also captures the attributes better. However, it produces less variety of images and worse quality compared to the original one.

5.7. Larger Noise Dimension

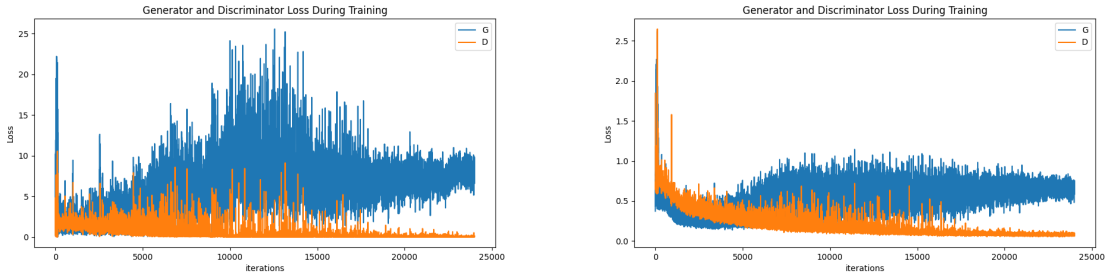
To provide more diversity to Pokémons with similar attributes, we train a version with a noise dimension of 512 for 500 epochs. Fig. 11 shows similar loss performance for this version, nevertheless, the generated images contain wavy or line patterns and the result is not favorable. This could be due to the lack of data samples given a higher dimension of input. While enlarging the capability of the model, it is more difficult to train on the other hand.

5.8. Without Noise

On the other side, we try to attain better quality with the sacrifice of diversity. We train a version with no noise vector feeding to G for 500 epochs. Shown in Fig. 12, this version performs similarly to the original one in terms of training loss. While we can see the strong similarity between generated Pokémons corresponding to similar real Pokémons, the different generated Pokémons look almost identical. Besides, the generated images contain large spot or hexagon patterns. This might be due to the insufficient capability of the model trying to generalize different Pokémons with the same features.

5.9. Balancing Training Of Generator And Discriminator

We see that the generator loss of our proposed model slowly increases from the middle of the training while the discriminator loss keeps decreasing. This might be an indication that the discriminator is dominating in the training. Therefore we modified the training routine so the discriminator is trained once for every n^{th} iterations in the n^{th}-$hundred$ epoch, while the generator is trained in every iteration. Moreover, we increase the weighting for the distinguishing label to 10, aiming to generate more realistic images by learning more from the distinguishing label instead of the attributes. We train this version for 500 epochs and obtain Fig. 13. Although the discriminator seems to be less dominating, the loss increases and is less stable. The generated images are more complicated but still contain some obvious patterns. Despite being diverse, it does not capture the attributes well.



(a) GAN (b) Proposed Model

Figure 8. Sec. 5.5: Training losses for the two models for 500 epochs.

(a) GAN

(b) Proposed Model

Figure 9. Sec. 5.5: Comparisons with real images for the two models for 500 epochs.

6. Conclusion

We have proposed an alternative way to generate conditioned images through GAN, have demonstrated its advantage over traditional methods, and experimented with some different settings integrating our method. We believe it can be useful in generating data containing multiple labels concurrently.

Training GAN models is a challenging task as it is tricky to balance the discriminator and the generator. In the future, we will continue to fine-tune the hyper-parameters in detail to boost the model's performance in Pokémon generation. We would also try to adapt different strategies to improve image quality and stabilize the training process. It is also important to prepare a better data set since the current one only contains one image for each species.

On the other hand, although our proposed method is successful on the selected task, we should further experiment with some more popular and sophisticated data sets to prove that it works in general. More quantitative measurements should be applied to analyze the behaviour of the model.

Figure 10. Sec. 5.6: Results of the proposed model with lower learning rate for 500 epochs.

Figure 11. Sec. 5.7: Results of the proposed model with a larger noise dimension for 500 epochs.

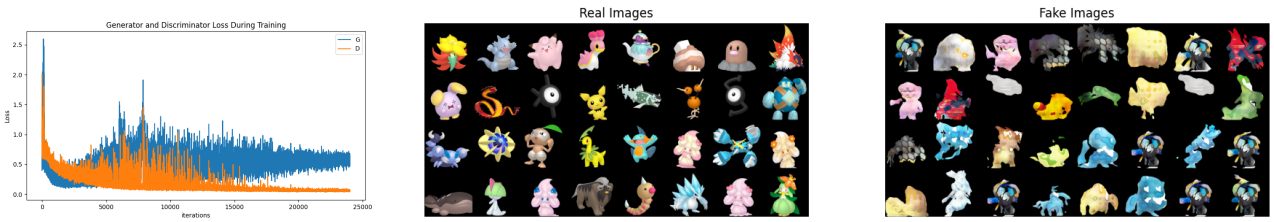


Figure 12. Sec. 5.8: Result of the proposed model without noise input for 500 epochs.

Figure 13. Sec. 5.9: Result of the proposed model with decaying training frequency on discriminator for 500 epochs.

References

[1] AlexMGitHub. Pokegan, 2021. https://github.com/AlexMGitHub/PokeGAN. 1

[2] Gonçalo Chambel. Generating realistic pokemons using a dcgan, 2022. https://medium.com/@goncalorrc/generating-realistic-pokemons-using-a-dcgan-331c7f75e211. 1

[3] The Pokémon Company. Pokémon home, 2020. 2

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 5

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 3

[6] Nathan Inkawhich. Dcgan tutorial, PyTorch. 2023. https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html. 3

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[8] Justin Kleiber. Pokegan: Generating fake pokemon with a generative adversarial network, 2020. https://blog.jovian.com/pokegan-generating-fake-pokemon-with-a-generative-adversarial-network-f540db81548d. 1

[9] ARTUR MACHADO LACERDA. Pytorch conditional gan, 2018. https://www.kaggle.com/code/arturlacerda/pytorch-conditional-gan/notebook. 3

[10] Xudong Mao, Qing Li, Haoran Xie, Raymond Lau, and Wang Zhen. Multi-class generative adversarial networks with the l2 loss function. 11 2016. 4

[11] Xudong Mao, Qing Li, Haoran Xie, Raymond Lau, Wang Zhen, and Stephen Smolley. Least squares generative adversarial networks. pages 2813–2821, 10 2017. 4

[12] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[13] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 3

[14] ENSUENO PABON. Gan demo: Pokemon generator, 2019. https://www.kaggle.com/code/enspabon/gan-demo-pokemon-generator/notebook. 1

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[16] Xiao Yang and Majid Komeili. 2d art generation with progressive gan. 10.13140/RG.2.2.19062.27207, 11 2022. 1

[17] Zhenye-Na. pokemon-gan, 2018. https://github.com/Zhenye-Na/pokemon-gan. 1