Gotta Generate ’em All! Pokemon (With Deep Learning)

Devi Acharya[*]
University of California, Santa Cruz
dacharya@ucsc.edu

Rehaf Aljammaz[*]
University of California, Santa Cruz
raljamma@ucsc.edu

Beth Oliver[*]
University of California, Santa Cruz
elaolive@ucsc.edu

Mirek Stolee[**]
University of California, Santa Cruz
mstolee@ucsc.edu

ABSTRACT

When using multiple generators to construct content, it is important to ensure that each generated part forms a cohesive whole. We focus on generating trading cards based on an existing set of Pokemon cards, generating text and images by training different machine learning algorithms on the original card data. We also use the original card data to train classifiers that sort images and text by each Pokemon’s elemental type. We can use these classifiers to sort generated images, names, and attacks for cards into the existing elemental types. We theorize that by using a pipeline of image and text generation and then classification, we can create trading cards with a consistent design akin to existing cards in the game.

CCS CONCEPTS

• **Computing methodologies → Neural networks**.

KEYWORDS

datasets, neural networks, image generation, text generation, games

ACM Reference Format:
Devi Acharya, Rehaf Aljammaz, Beth Oliver, and Mirek Stolee. 2020. Gotta Generate ’em All! Pokemon (With Deep Learning). In ,. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine learning has been used to create novel text and images based on sets of data. Where machine learning is currently less successful is in creating generated objects with different, consistent parts. While some work has been done creating different generated elements that work well together, such as generated images to match captions or vice versa, or some forms of card text and images, we examine how we can use classifiers on generated content in order to create cards with an overall consistent design. In other words, to use classifiers to find images and text that complement and make sense together.

[*]all authors contributed equally to this research.

For this project, we focus on the generation of trading cards using data from existing cards in the Pokemon trading card game. The Pokemon trading card game has thousands of cards with unique names, card art, and sets of attacks, each featuring an elemental "type" that is distinct from other types in abilities and style. Being able to generate these cards, rather than hand-author art, attacks, and the overall design of the card, would significantly reduce the labor required to make such cards. That said, the cards generated through this project do not yet achieve the same level of quality shown in human-authored cards. The goal of this project is to explore how we can use existing machine learning techniques in combination with image and text classification techniques in order to generate Pokemon trading cards in which the generated parts make sense together and the card overall is something emblematic of an original card from the trading card game.

Being able to generate such cards has interesting implications. Firstly, it will help us better understand features that are common to each Pokemon type. By examining how well Pokemon card images and text can be classified, we can assess the accuracy of such classification to see how distinct each type is, and possibly identify which features are used to distinguish card types from one another. Secondly, we can examine how the method of classifying generated content creates cards that are more consistent overall, since individual elements such as name, image, and attacks are all classified into the same type. Finally, we can compare generated trading cards to existing trading cards in game to see in what ways the different methods of generation failed, and what can be improved on in later work.

2 PROBLEM STATEMENT

Our intent in developing this project is twofold. The first is to explore various machine learning methodologies for creating novel text and images based on existing data, focusing on neural networks and classification for generation and categorization of new artifacts. The second purpose is blending generated artifacts together to create a new, cohesive product. We chose playing cards as a platform, specifically the Pokemon playing cards, as we believe that cards hold structures fit for text generation, rule constraints, and [21] pose a challenging problem. In the following sections, we present similar work, our data collection methodology, model, implementation approaches taken, and finally, the paper ends by discussing our concluding thoughts and the limitations posed in this project.

3 RELATED WORK

Previous researchers [15, 21] have used card games as platforms within the machine learning domain. The authors [21] analyzed textual and visual information on playing card games, specifically Magic the Gathering through neural networks. They introduced a methodology for linking generated text to a given input image through classification means. Ling et al. [15] introduced the latent neural network; one of the main contributions of this paper is the network's ability to generate minimized code. The authors used digital game cards, specifically from the trading card games Magic the Gathering and Hearthstone, to generate text and the accompanying code as a way to validate their model [15]. Others have even designed card generation tools, ex. Mystical Tutor, by taking in user input and generating designed cards [17].

This paper presents both text and image generation methods. We mainly used recurrent neural networks, specifically char-rnn, for text generation, and tested out different generative adversarial networks (GANs) for image generation. Authors have improved recurrent neural networks (RNNs) in generating text. Graves showed the potential of generating sequential text using Long Short-term Memory (LSTM) as a basis and predicting upcoming text in complex textual structures [11]. Furthermore, authors have coupled an RNN with their Hessian-Free optimizer (HF) in creating character-based models [18], further contributing to the text generation domain.

Goodfellow et al. introduced the concept of GANs [10]. Other researchers have furthered the cause in developing different variations of GANs, including layering networks used in generating high-resolution images and captioned image results with stackgan [10], improving performance and functionality with the Wasserstein Gan (WGAN) [9] and finally, image translation by taking in an outline or a silhouette and producing an image in pix to pix [13].

4 DATA COLLECTION

4.1 Dataset Extraction for card images and text assets

We collected a set of 6,534 trading cards using the Pokemon TCG Developers API [1]. The Pokemon Trading Card Game [8] has several "supertypes" of cards, including Pokemon cards, Trainer cards, and Energy cards. The central mechanic of the game is centered around each player using "Pokemon" cards to fight, with "Trainer" and "Energy" cards supporting them. Cards in the "Pokemon" supertype also have a fairly homogeneous structure, sharing common elements such as attacks. Due to these features, we decided to focus on collecting and generating cards of this supertype. We used the API to access cards of supertype "Pokemon," downloading the relevant card images and text for names and attacks.

Both images and text pulled from the database were then sorted using the elemental "type" of the card. Most cards have a single type, and we only used the first type of those that have two. In general, card art, names, and attacks are common to cards of certain elemental types. For example, the fire type Pokemon card "Magmar" borrows its name from the word "magma" and has volcanic attacks like "eruption" and "combustion". The card art for "Magmar" primarily features the color red, a common element across many fire type cards. While not all card features are this closely-linked to their type, there is enough of a correlation that our classifier has some success sorting them by type.

As some of the types have many more cards than others, we condensed the eleven types down to seven to achieve more similarity in size between categories for better training and classifying. We added the "Darkness" type to the "Psychic" dataset, the "Dragon" type to the "Water" dataset, the "Fairy" type to the "Colorless" dataset, and the "Metal" type to the "Fighting" dataset.

We then performed some processing on the data in order to aid in the classifier training process and the generation process. For images, we sorted each image into a folder by its type, so that the classifier could then be trained to distinguish images of certain types from one another. The original card images featured the entirety of the trading card, including images of the text on the card and other extraneous information. Because we wanted our GAN and classifier to only use the image of the Pokemon itself (see Figure 3, "card art"), we cropped each card image down to only show the Pokemon's illustration.

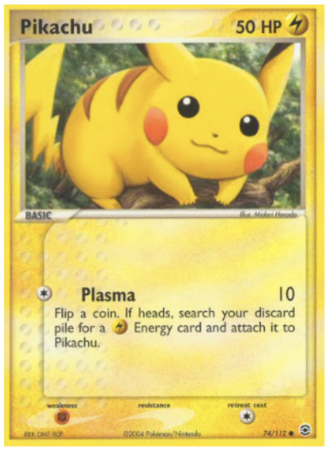
Pikachu
50 HP
BASIC
Plasma
10
Flip a coin. If heads, search your discard pile for a Energy card and attach it to Pikachu.
weakness
resistance
retreat cost

Figure 1: The entire trading card

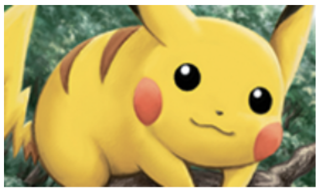

Figure 2: Our cropped image of the original card, showing only card art

For the text, we used the API to extract each Pokemon name and attack, putting them into comma-separated values (CSV) text files labeled with each type for later processing. For the names, this is simply a CSV of the Pokemon's type, then a comma separating the type from the Pokemon's name, in the format "type, name". For the attack, we accessed a dictionary through the API containing all of the information about each attack (cost, name, text, and damage) and wrote those to a CSV with double quotes surrounding the

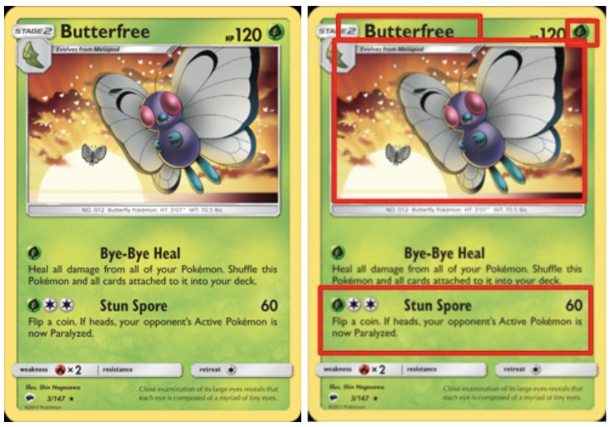

Figure 3: (Left) A typical Pokemon trading card; (Right) Different sections of the trading card outlined: Top left, name; top right, type; center, card art; bottom, attack

attack text so that commas in the body of the card would not be interpreted as a new item in the CSV.

Grass,"['Grass'],Absorb,'Remove 1 damage counter from Turtwig.',10"

Above is an example of the attack text of a card in CSV format, first listing the type, then the body of the attack text–attack cost, name, description, and damage

Finally, we created a JavaScript Object Notation (JSON) structure for names and attacks to use in text classifier validation, using the same data as contained in the CSV.

"type": "Psychic", "description": "['Psychic', 'Psychic', 'Psychic'],Dark Flash,'This attack's damage isn't affected by Resistance.',120"

4.2 Other data sets

As an alternative approach for image generation, we used a Pokemon data set provided by a Github user kvpratama [14]. This data set includes images of 800 Pokemon sprites with transparent backgrounds. We are using this set as an alternative way of generating Pokemon and reducing noise that may be provided by background art located in cards. This sample is used with a wgan. The base Github code was provided by siraj's pokegan generator [16].

5 IMPLEMENTATION

5.1 System Model

By examining the trading cards extracted through the Pokemon TCG API, we were able to find certain features common across Pokemon cards that we could then use to generate new cards. For instance, most cards of supertype "Pokemon" have a type, name, list of attacks, and an image of that Pokemon, all of which are important for determining who the Pokemon is, and their abilities (Figure 4). We used this to decide what parts of the trading cards to generate, using a recurrent neural network (RNN) to generate text for Pokemon names and attacks, and a generative adversarial network (GAN) to generate images.

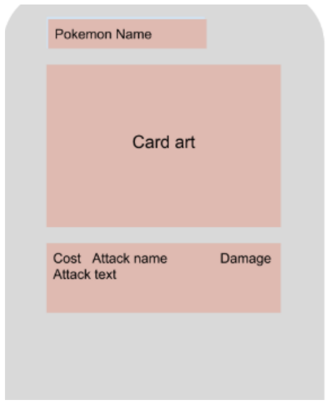

Figure 4: Our model for generated cards, where red sections will be generated.

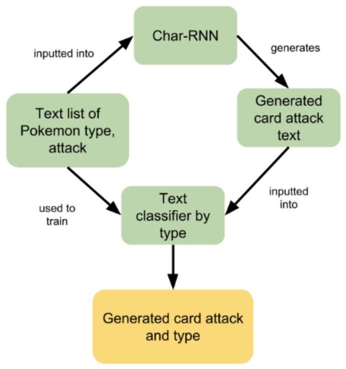
Char-RNN
inputted into
generates
Text list of Pokemon type, attack
Generated card attack text
used to train
inputted into
Text classifier by type
Generated card attack and type

Figure 5: Our model for generating and classifying Pokemon attacks by type. A similar model was used for names.

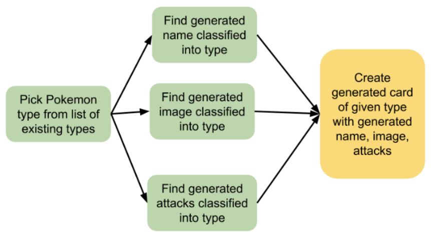
Pick Pokemon type from list of existing types
Find generated name classified into type
Find generated image classified into type
Find generated attacks classified into type
Create generated card of given type with generated name, image, attacks

Figure 6: Complete pipeline for generating a Pokemon card from various generated components using classifiers.

6 GENERATION

6.1 Image Generation

We investigated three different models for generating images.

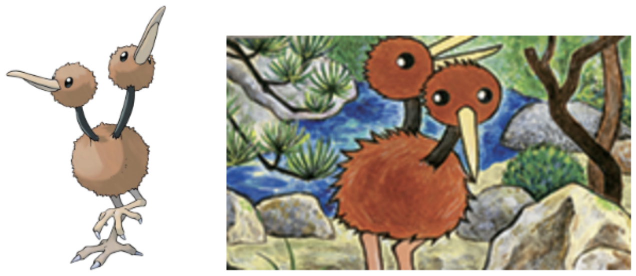

Figure 7: Left: Pokemon sprite without a background. Right: Card art of the same Pokemon.

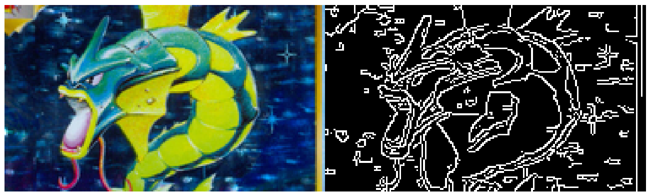

Figure 8: Left: Target image. Right: Input image (obtained by running target through CV2 Canny edge detection).

The first was a Generative Adversarial Network (GAN) written by Manu Mathew Thomas [19]. The training data set included the Pokemon card image set of 6,534 images. The generator was made up of three hidden layers while the discriminator was made up of four. This GAN was trained on a batch size of 5 over 150k iterations.

Our second attempt at image generation was a Wasserstein GAN [9] written by Siraj Raval [16]. According to Siraj[2] the WGan provides better results as it implements a better alternative to the fitness function by decreasing the Wasserstein distance. We trained the two models with Siraj's WGan implementation each on a different data set. The first model was trained on 800 images of Pokemon sprites without any backgrounds over 2800 epochs. We trained this model using a remote GPU provided by Floydhub[E]. The training process entailed 30 hours on a High Performance 12-16 GB Memory GPU with a Tesla V100 chip. The second was trained on 6800 images of Pokemon cards over 800 epochs on our professor's lab's computer over 48 hours. Both models trained with a batch size of 64. Figure 7 illustrates the image samples used in the pokeGAN, left to right shows the Pokemon sprite and a Pokemon card used in the process.

Finally, pix to pix [12] was our final attempt in image generation. This approach focused on an image's edges and silhouette providing a better distinction of the object from the background. We used the Pokemon card image set as a training data by first running it through a CV2 Canny edge detection script and using the produced output as an input into the GAN. We used a model provided by [3] and trained the model on 5200 images. Figure 8 illustrates a sample input and target image.

6.2 Text Generation

We generated text using Sherjil Ozair's TensorFlow implementation [4] of Andrej Karpathy's Char-RNN model [5]. This model, a recurrent neural network, generates text character by character. Two Char-RNN models were trained: one for card names and one for attacks. The training data for the card name model consists of the name of every card in the dataset, and the attack model training set consists of the attack cost, the attack name, the description of the attack, and a number representing the damage the attack deals to the opposing Pokemon.

7 CLASSIFICATION

7.1 Image Classification

Our generated images were classified using Manu Mathew Thomas' SimpleImageClassifier [20]. This convolutional neural network sorts images. Our model was trained on the card art for each card in our dataset, sorted by type.

7.2 Text Classification

We performed the text classification using Jie Zhang's Multi-Class Text Classification CNN [6]. This convolutional neural network is used to sort text into a number of predetermined categories. We again trained two models, one for names and one for attacks. The dataset used for each model was the same as the dataset used for the text generation for each field, with the addition that each name and attack was associated with the elemental type of its card. For example, the name "Bulbasaur" was associated with the grass type because it appears on grass type cards, and the attack "Vine Whip" from the same card is also associated with the grass type. The models sort text into our seven types: "Colorless", "Fighting", "Fire", "Grass", "Lightning", "Psychic", "Water". After training the classifier on the existing set of names and attacks, we used it to classify generated names and attacks into one of the given types.

8 CARD CREATION

To create cards from our generated content, we picked an elemental type and found a generated image, Pokemon name, and set of attacks that were classified into the given type. We then used Pokemon Card Maker [7] to assemble each of these parts into a whole card.

9 RESULTS

9.1 Text Generation

The names produced by the name generation model are generally believable. Examples such as "Draquarina, "Liracion", and "Scentar" fit in with human-authored names like "Chandelure" and "Vanillite". The model also demonstrates understanding of specific name structures. Cards in the original dataset such as "Sabrina's Gengar" refer to Pokemon that belong to a specific Pokemon trainer. The model produces card names like "Turnha's Gublir" that use this structure. Additionally, while the model sometimes produces names that already exist in the original dataset like "Slowking" and "Growlithe", some of the Pokemon cards in the dataset already share the same name. Producing additional cards with the same name is not necessarily an issue.

The attack generator produces attacks that imitate the structural format of the attacks in the dataset. The attacks generated have a cost in energy, an attack name, a description, and damage. They are listed in that order within the attack text.

It also recognizes that not all attacks have all of these components. For instance, simple attacks like "Tackle" in the training dataset do not have a description and simply inflict the amount of damage listed:

> [’Colorless’, ’Colorless’],Hang Wind,”,20

This generated attack, "Hang Wind", has no description text. Other attacks might do no damage at all:

> [’Colorless’, ’Colorless’, ’Colorless’],Surfen,’Flip a coin. If heads, your opponent discards the top card of his or her deck. Shuffle your deck afterward.’,

This generated attack, "Surfen", has a written effect in the description but does not deal damage. The original dataset contains instances of similar effects.

The attack generator could be improved in the future. Not all of the generated attack descriptions make sense within the rules of the game:

> [’Water’, ’Colorless’, ’Colorless’],Echaring Magma Hy,’Discard a Psychic Energy attached to this Pok\u00e9mon. This attack does 10 damage times the number of heads.’,20\u00d7

This attack, "Echaring Magma Hy", does not instruct the player to flip any coins but does damage depending on the "number of heads". This attack also illustrates an issue the model has handling special characters. The "\u00e9" is a stand-in for the accented "e" character in the game’s name, and the "\u00d7" stands in for the small "x" character used in place of the word "multiply" in attack descriptions. This can be fixed easily by replacing these stand-ins with the proper characters.

9.2 Text Classification

The accuracy of the text classifiers can be assessed by running them on a test sets of names and attacks set aside for this purpose. The name classifier tests at an accuracy around 85%, and the attack classifier performs at around 80%. Although the name classifier ostensibly performs at a higher percentage, in reality the trained model seems to converge on a singular type and then assign that type to all testing data. This may be due to the fact that names are much shorter than attacks, and usually singular words. This may make it difficult for the classifier to differentiate between names. For the generated cards in the next section, names were not classified.

9.3 Image Classification

As with the text classifier, the image classifier can be assessed by running it on a set of images set aside for this. On this test set, our classifier only preforms at 32%. This is initially fairly underwhelming results. However, upon a manual inspection of its guesses to the card art the reason for this poor performance was, to some extent, explained.

All Pokemon cards are based off of characters from the animation and/or the video game series. In both other formats, typing is more complicated, and many have two types. Consider Figure 9, this particular Pokemon in the video game is dual typed "Water" and

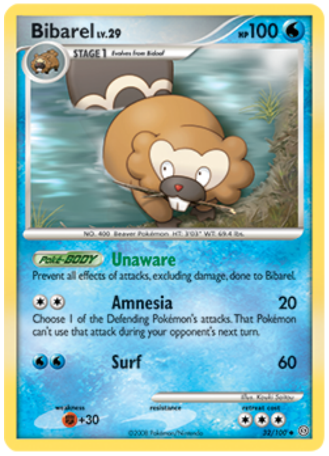
Bibarel LV.29
HP100
STAGE 1 Evolves from Bidoof
Poké-BODY Unaware
Prevent all effects of attacks, excluding damage, done to Bibarel.
Amnesia 20
Choose 1 of the Defending Pokémon's attacks. That Pokémon can't use that attack during your opponent's next turn.
Surf 60
+30

Figure 9: An example of a Dual Typed Pokemon. This card is typed as "Water", but in the animation and in the video game, "Bibarel" is a "Normal/Water" type.

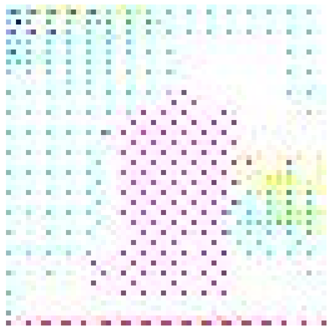

Figure 10: A sample output from our first attempt at generating images.

"Normal" ("Normal" is the video game equivalent to the card game’s "Colorless"). The card was classified as "Colorless" which would be accurate if looking at the Pokemon outside the very specific context of the card game, and is in fact more representative of its general design (most "Colorless" Pokemon are whiles, dull browns, or pale pinks). To complicate maters, there are many different cards featuring this Pokemon, some of which are "Colorless" and some of which are "Water".

9.4 GANs

9.4.1 Manu Mathew Thomas’s. The first GAN results were unfavorable. The resultant images arguably featured a central dotted figure but lacked depth, lines, and features. Figure 10 illustrates these initial results. We believe many factors contributed to these results, insufficient training time and insufficient complexity in the model’s structure being the most significant.

9.4.2 WGAN. As for Siraj’s WGAN model [16], the initial training was done on a set of 100 card images for 2500 epochs, the resultant image was colorful, arguably featuring a central figure with colors separate from the background (Figure 11). We then trained the data

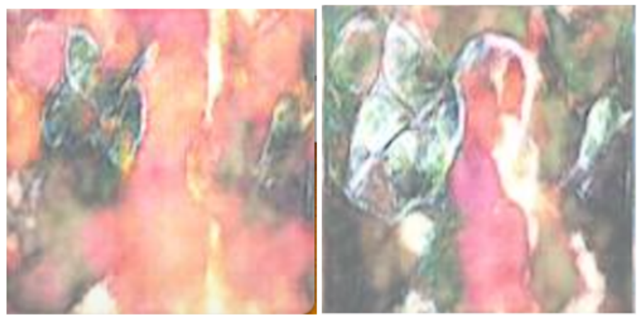

Figure 11: Two sample outputs from the WGAN trained on 100 images for 2500 epochs.

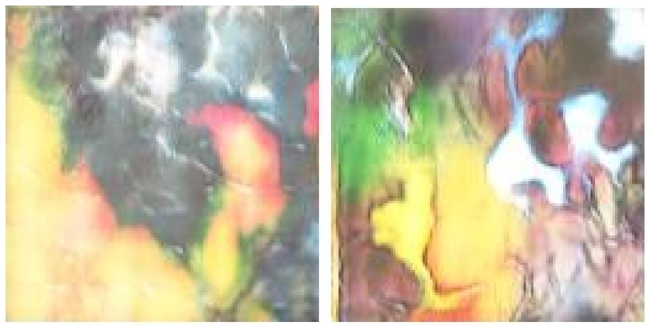

Figure 12: Two sample outputs from the WGAN trained on 6500 images for 800 epochs.

set with 6500 cards and for 800 epochs. We had hoped to train it over more epochs than this, however, due to technical issues such as training interruptions and corrupted checkpoints we had to end it there. The results are not significantly greater than the first, but that we got this far with more data in fewer epochs is promising.

As an alternative approach, we trained a version on a dataset consisting of Pokemon sprites instead of Pokemon card images (see Figure 7 for the difference). The results presented a clear outline for some of the generated Pokemon, giving it a clear shape and a black background color. We then classified the image based Manu Mathew Thomas's SimpleImageClassifer into the aforementioned types and modified the black background color to fit that type instead.

9.5 Pix2Pix

Upon seeing the results of our GANs thus far, it was suggested to us to try the pix2pix model [12]. As such, we trained a model based on an implementation by "affinelayer" [3]. This model was trained on 5200 card art images which had been run through an edge detection algorithm. Training ran for 26 epochs over 4 days on a CPU before time ran out on us again. Still, the results are not terrible, somewhat washed out, but beginning to take the colors one might expect.

Ideally, this could be used in conjunction with a model which generated edges to create better images. As things stands, we hooked this model up with our previous GANs, their output run through the same edge detection algorithm we used to create our training

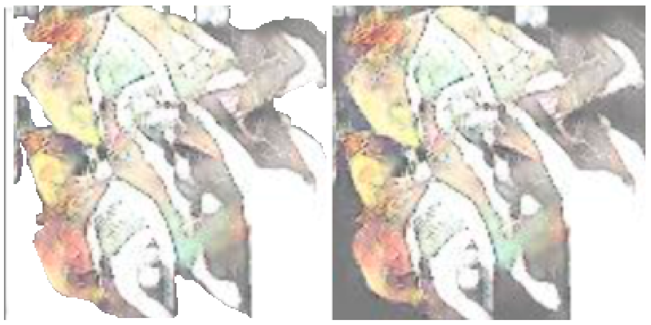

Figure 13: A sample output from the WGAN trained on 800 images on a dataset of only Pokemon sprites on both a white background (left) and a black background (right).

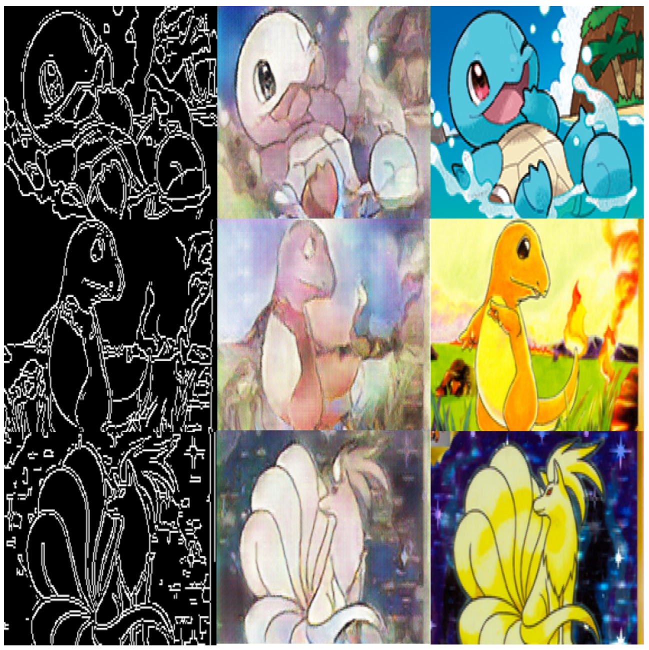

Figure 14: Left: Input to the pix2pix model. Right: Target. Center: Output from the pix2pix model.

data. The results are still underwhelming, but perhaps promises another direction to pursue in the future.

9.6 Completed Cards

Combining various results from text generation (RNNs) and image generation (GANs) and classifying them we had a somewhat cohesive overall look. While not perfect, there is a general shape, text and color structure that we think fits with the type of classified cards. As an example, the images in figure 15 were categorized as Fighting/Ground types based on the classifier post generation. The input image was from the Pokegan (card model version) both the name, text and HP were also generated using the text generation neural networks.

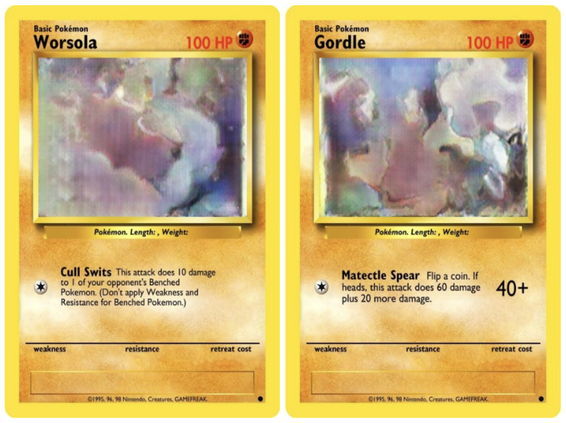
Basic Pokémon
Worsola
100 HP
Pokémon. Length: , Weight:
Cull Swits This attack does 10 damage to 1 of your opponent's Benched Pokemon. (Don't apply Weakness and Resistance for Benched Pokemon.)
weakness
resistance
retreat cost
Basic Pokémon
Gordle
100 HP
Pokémon. Length: , Weight:
Matectle Spear Flip a coin. If heads, this attack does 60 damage plus 20 more damage.
40+
weakness
resistance
retreat cost

Figure 15: Figure 15: From left to right: Worsola and Gordele generated Fighting-type cards

Another example shows the Pokegan sprite version figure 16 and the accompanying generated image. The generated Pokemon was classified as a psychic type. The color palette and shape, in our opinion, matches the nature of the card with the only difference being a more defined outline. All three samples have been passed through the pix-to-pix model to get a more defined shape and outline.

figure 16:

10 DISCUSSION AND LIMITATIONS

In the execution of this project we discovered a number of difficulties with our dataset and limitations in our approach, in particular in regard to card art and combining of all the elements in a harmonious way, especially in contrast to other, superficially similar projects.

In our attempts to classify and generate images for our cards we discovered that the results were generally very colorful blobs. We believe much of our difficulties in getting concrete images was in the nature of our dataset. Although the animation and the video game versions of Pokemon have distinct and consistent styles, the art on the cards are from a wide range of people with little attention placed toward creating a consistent style. Art ranges from an anime style to 3D modeled to highly simplified art. In addition, Pokemon are displayed at dramatic, and wildly varying angles lessening what similarities in silhouette there might have been. Furthermore, card art can have many variations, on varied kinds of backgrounds, sometimes featuring other extraneous characters (such as other Pokemon, or trainers). Because of this, it may be harder for the classifier to detect and the GAN to recreate various features common to different Pokemon.

Setting aside the challenges of generating and classifying images, there are several structural elements of the card that should all make sense when seen together. Our approach attempts to achieve such harmony between elements by organizing all generated parts by type. This approach assumes that all names, art, and attacks within a type are interchangeable, but this may not be an entirely safe assumption.

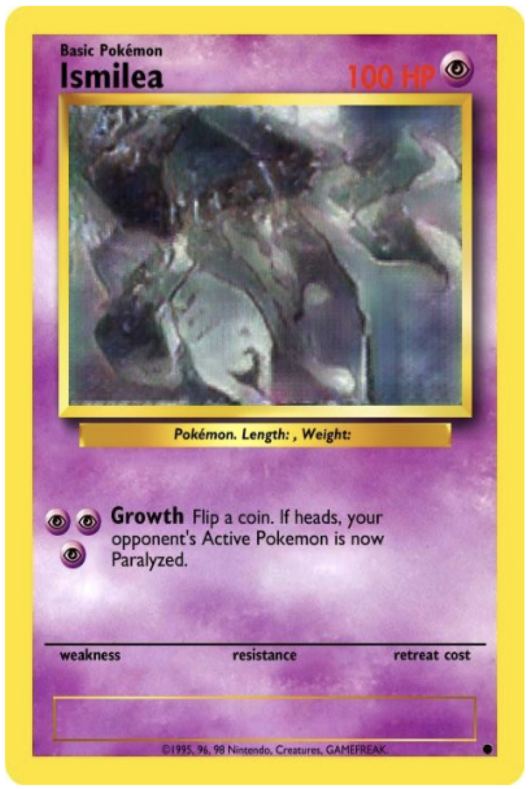
Basic Pokémon
Ismilea
100 HP
Pokémon. Length: , Weight:
Growth Flip a coin. If heads, your opponent's Active Pokemon is now Paralyzed.
weakness
resistance
retreat cost

Figure 16: Figure 16: Ismilea a psychic Pokemon card

Consider the two Pokemon in Figure 18. On the left is feeble Magikarp, on the right, mighty Gyarados. Both are "Water" types, so for our purposes all their attacks would be "Water" as well. Swapping their names would probably be fine in this case, odd to someone familiar to the game, but not outlandishly so. "Flail" on Gyarados probably would be fine too. "Dragon Spark" or "Pulverize" on Magikarp though might result in some raised eyebrows.

Another factor we that should be mentioned is the difference between Pokemon cards and other similar datasets, such as anime. At a glance, both are stylized art, generally with large eyes and unrealistic body proportions. However, anime characters have much more uniformity between different characters, generally having the same number limbs, similar–if exaggerated–body proportions, and wide facial features. When compared to the variety in Pokemon body structure, where even the number of heads the subject might have is not standard, even setting aside the differences in art style, it is little wonder we had such difficulties.

11 FUTURE WORK

We wonder whether more promising results could be obtained by training our image generating algorithms on a subset of Pokemon (for example, all Pokemon which look like dogs). Perhaps, if all members of the dataset had more in common with each other, their features could more efficiently be learned and Pokemon of that kind (continuing the example, new dog-like Pokemon) might be created.

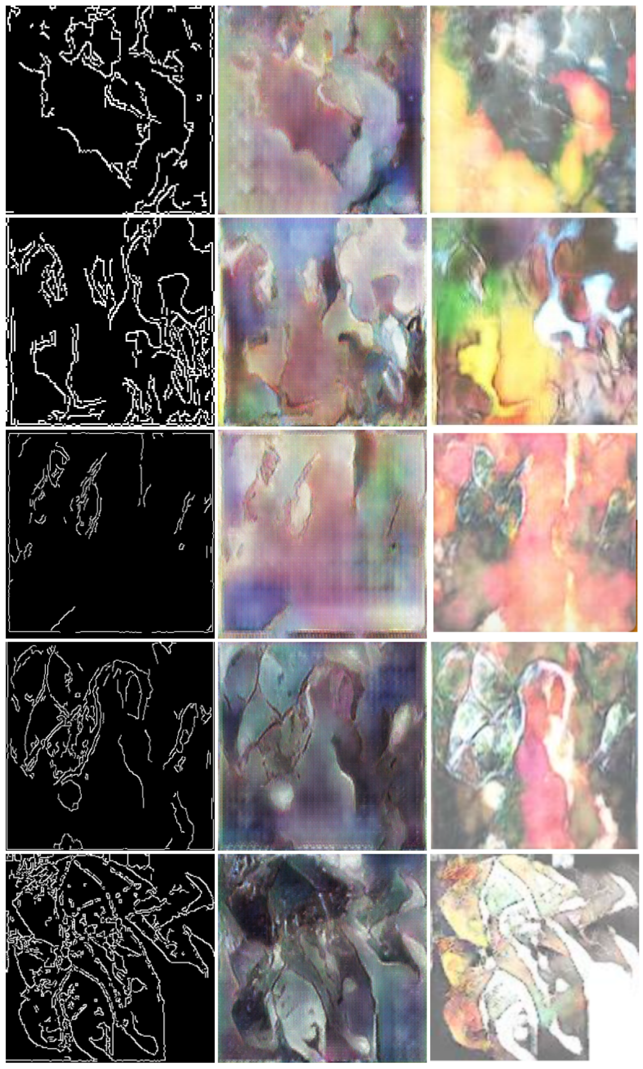

Figure 17: Right: "Target". An image generated by one of the previously mentioned GANs. Left: Input to the pix2pix model created by running the right most image through our edge detection algorithm. Center: Output from the pix2pix model. Top 2: Input generated by the model trained on 6500 card art images. Third and Fourth Rows: Input generated by the GAN trained on 100 card art images. Fifth Row: Input generated by the model trained on 800 Pokemon sprites.

A different approach might involve how other online users handled Pokemon generation. They drew and styled generated characters with the help of a human artist. Perhaps, this project can serve as a stepping stone for card designers designing creatures.

There are also additional fields that our model does not generate. Pokemon cards have a health point value and a height and weight for the represented Pokemon, for example. Future attempts could generate this data as well for a more complete card. Our model also glosses over one of the core features of Pokemon, "evolution". Future work could perhaps attempt to generate "evolutionary lines".

Basic Pokémon
Magikarp 30 HP
Fish Pokémon. Length: 2' 11", Weight: 22 lbs.
Tackle 10
Flail Does 10 damage times the number of damage counters on Magikarp. 10x
weakness resistance retreat cost
In the distant past, it was stronger than its horribly weak descendants that exist today. LV. 8 #129
Illus. Mitsuhiro Arita © 1995, 96, 98, 99 Nintendo, Creatures, GAMEFREAK. © 1999 Wizards. 35/102

Gyarados 90 HP
STAGE 1 Evolves from Magikarp Illus. Hisao Nakamura
Dragon Spark
This attack does 10 damage to each of your opponent's Pokémon. (Don't apply Weakness and Resistance for Benched Pokémon.)
Full Retaliation 20x
Does 20 damage times the number of damage counters on all of your Magikarp.
Pulverize 50+
If the Defending Pokémon already has at least 2 damage counters on it, this attack does 50 damage plus 50 more damage.
weakness resistance retreat cost
TH2-6LA-7LA ©2005 Pokémon/Nintendo 8/107

Figure 18: Left: Magikarp, comically weak. Right: Gyarados, famously strong. Now imagine their move sets swapped.

12 CONCLUSION

We theorize that by grouping content generated by a variety of different sources by classifying them into the same type, we can create cards with an overall more consistent look and design than cards with purely randomly generated content. We have only begun to scratch the surface with the ways we have combined different machine learning techniques, and we do not claim all of the combinations we tried were a success. However, we hope that this might provide a stepping stone for others to go further.

REFERENCES

[1] [n. d.]. https://pokemontcg.io/
[2] [n. d.]. https://www.youtube.com/watch?v=yz6dNf7X7SA
[3] [n. d.]. https://github.com/affinelayer/pix2pix-tensorflow
[4] [n. d.]. https://github.com/sherjilozair/char-rnn-tensorflow
[5] [n. d.]. https://github.com/karpathy/char-rnn
[6] [n. d.]. https://github.com/jiegzhan/multi-class-text-classification-cnn
[7] [n. d.]. https://www.pokecard.net/
[8] [n. d.]. PokÃÎmon Trading Card Game Rules. https://assets.pokemon.com//assets/cms2/pdf/trading-card-game/rulebook/sm9_rulebook_en.pdf
[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*. 214–223.
[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
[11] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *arxiv* (2016).
[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.
[14] kvpratama. 2018. Pokemon Images Dataset. https://www.kaggle.com/kvpratama/pokemon-images-dataset
[15] Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744* (2016).
[16] Siraj Raval. 2018. Pokemon GAN. https://github.com/llSourcell/Pokemon_GAN
[17] Adam James Summerville and Michael Mateas. 2016. Mystical tutor: A magic: The gathering design assistant via denoising sequence-to-sequence learning. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference.*
[18] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 1017–1024.
[19] Manu Mathew Thomas. 2019. CMPM202_GANs.

[20] Manu Mathew Thomas. 2019. CMPM202_SimpleImageClassifier.
[21] Felipe Zilio, Marcelo Prates, and Luis Lamb. 2018. Neural Networks Models for Analyzing Magic: the Gathering Cards. In *International Conference on Neural Information Processing*. Springer, 227–239.