APS360
Classifying Pokémon Generations from Sprites

Team 10, Project Proposal

Word Count: 1399

Professor S. Colic

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Shashwat Panwar **Connor Lee** **Prerak Chaudhari** **Siddarth Narasimhan**

Introduction

The goal of this project is to design a machine learning model that identifies the generation of hardware that a Pokémon sprite corresponds to. Nintendo has released seven generations of games across five handheld devices. The Nintendo Switch is the platform for generation eight Pokémon but it is still a work in progress as Pokémon are still being released and current collections of generation eight sprites are incomplete or inconsistent. Due to this, we are excluding this generation from our project for the time being.

The potential implementation of this project is on the twitch.tv streaming platform. Twitch requires streamers to display the game they're playing in the stream title so that it can be categorized correctly for viewers and statistics. This project would be the first step in making an auto-recognizer for streaming Pokémon games since sprites are a common in-game feature.

The games released on one device have little to no difference when it comes to the art style and many sprites are copied over exactly. For example, **Fig.1** shows the lack of a varying art style (size, resolution, colour variations, etc.) across two games on one device. The poses have been altered but we feel that does not warrant classifying these two sets of sprites differently.

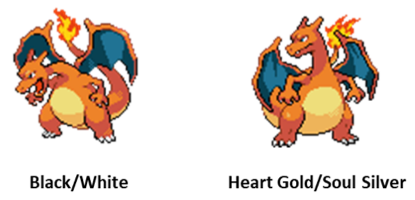

Figure 1: Charizard (#006) sprite from two different games released on the Nintendo DS.

The problem inherently involves using image classification to identify stylistic differences between several Pokemon sprites. Stylistic differences can range from the pixel resolution to colour and contrast, as seen in **Fig.2**. Due to the complex nature of this image classification task, a machine learning approach is most suitable.

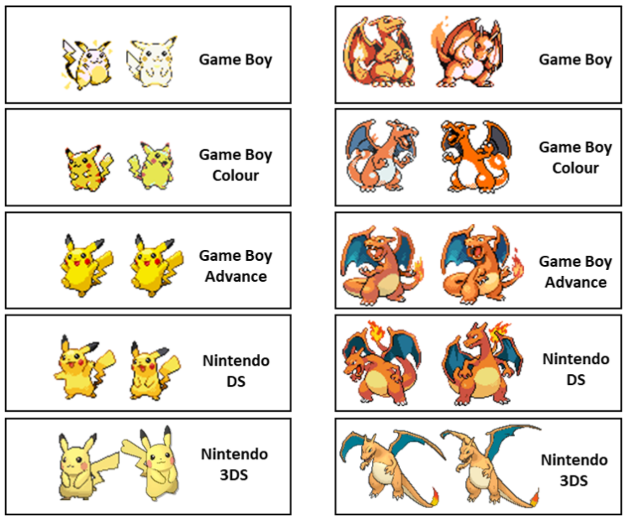

Figure 2: Sprites shown from five different generations of Nintendo devices. Notice how the Pokémon in any one generation share similar stylistic features, such as colour, resolution and shape.

Background and Related Work

There is evidence of little work (in fact, almost none) in the classification of Pokemon generations. The furthest people have gone is classifying the Pokemon themselves, their types, identifying legendary Pokemon, determining battle viability, or creating new Pokemon with GANs [1][2][3]. Seeing existing Pokemon datasets allowed us to streamline our data collection process.

Data Processing

The primary dataset will be obtained from an already scraped zip file available on GitHub [4], which already contains labels for the different generations of Pokemon. This data is split simply into the seven generations, however, as noted above, we will be combining the generations in the following way to classify sprites by the device generation they were made for:

- Game Boy: Generation 1
- Game Boy Colour: Generation 2
- Game Boy Advance: Generation 3
- Nintendo DS: Generation 4 & Generation 5
- Nintendo 3DS: Generation 6 & Generation 7

Part of the data processing will involve reorganizing the dataset to sort generations by their native hardware. In addition to this, we will scrape Pokemon databases such as **pokemon-trainer.com** using **this Python script,** which provides additional details about what games the sprites are from as well as the associated URLs.

There are various other databases to scrape from as fail safes [5]. The number of images per generations is limited and insufficient to train a good enough model. We will use Keras to perform data augmentation on our existing dataset [6]. Some generations have more images than others so we will need to ensure that the number of data in each generation is the same to prevent bias. There are some animated images, which will be removed, otherwise, static images will have to be created in the form of splitting a GIF into its frames and using those are individual images to grow our dataset.

Architecture and Illustration

An ANN is not ideal due to computational inefficiency, therefore, a CNN architecture will be used due to its superiority in image processing. As a starting point, we will use a CNN consisting of convolutional layers and max pooling, followed by fully-connected layers. We plan to follow an architecture similar to the one used in classifying cats and dogs in Lab 2, as depicted in **Fig.3**.

If we notice our model over-fitting too quickly, we will begin by adding dropouts with a probability of 20%, and starting with a weight decay of 0.001. Under-performing on the validation set, will be solved by gradually increasing the number of convolutional and fully connected layers.

An alternate architecture would involve training four neural nets that would evaluate input data through a series of binary classifications. This would tremendously increase training time but would likely decrease overall error.

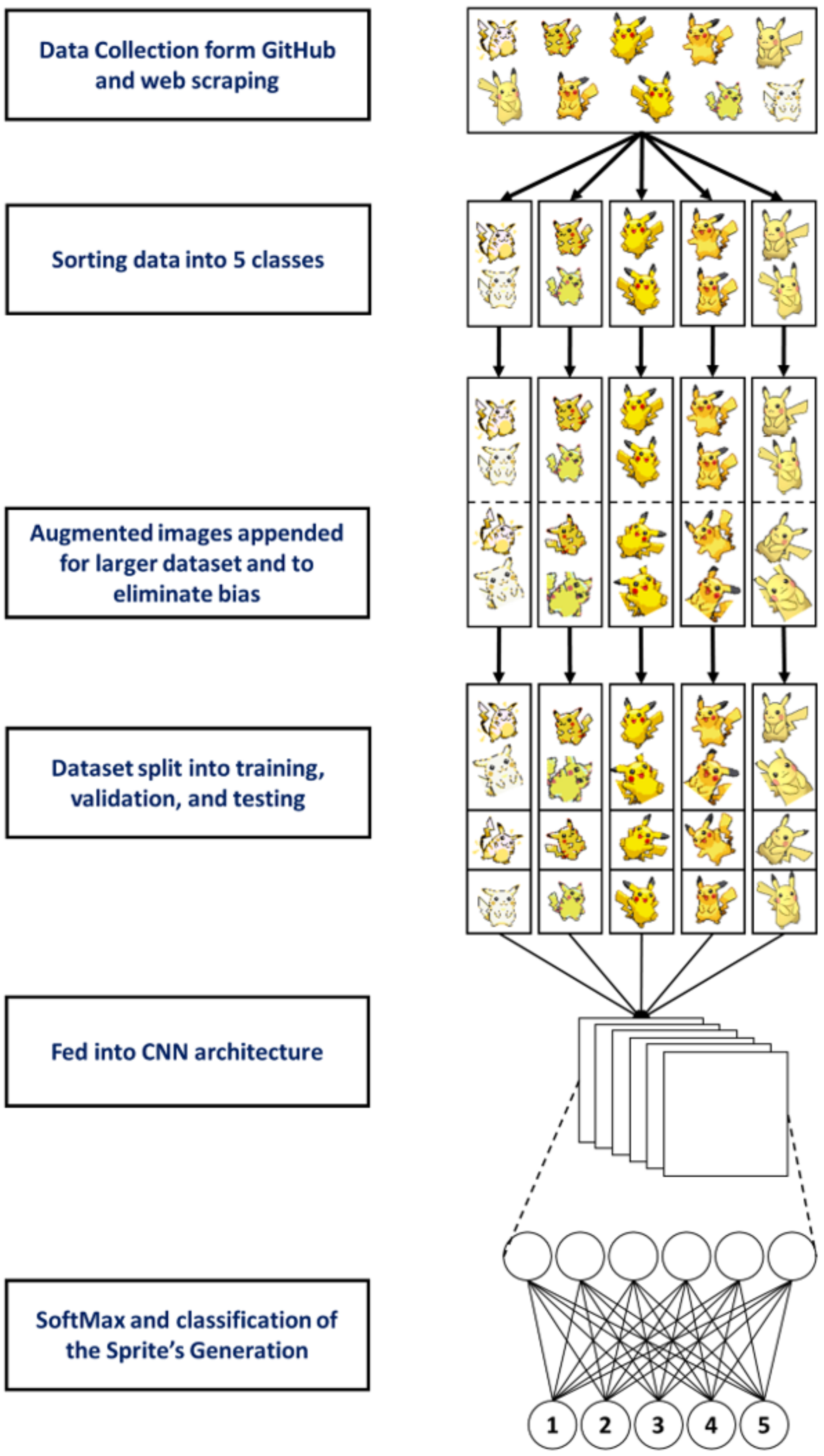
Data Collection form GitHub and web scraping
Sorting data into 5 classes
Augmented images appended for larger dataset and to eliminate bias
Dataset split into training, validation, and testing
Fed into CNN architecture
SoftMax and classification of the Sprite's Generation
1
2
3
4
5

Figure 3: Simplified outline of our proposed architecture with the required initial data collection and processing.

Baseline Model

According to the Scikit documentation, a SVM model would be appropriate for small-scale classification [7]. The SVM will transform the data using its kernel so that it is linearly separable and thus classifiable. However, SVMs have some drawbacks. Firstly, the model parameters are manually tuned. Secondly, model complexity cannot be increased. Finally, since SVMs use binary classification, multi-class implementations use several "one versus rest" models, increasing computation costs. CNNs are self-tuning models designed for image classification, offloading the responsibility of feature engineering optimizations [8].

Ethical Considerations

The art style for each generation is limited by the technology that powered their respective game [9]:

- Generation 1: Nintendo Gameboy (2-bit colour palette, 160 × 144 pixels screen) [10]
- Generation 2: Nintendo Gameboy Colour (15-bit colour palette, 160 × 144 pixels screen) [11]
- Generation 3: Nintendo Gameboy Advance (15-bit colour palette, 240 × 160 pixels screen) [12]
- Generations 4-5: Nintendo DS (18-bit colour palette, 256 × 192 pixels screens) [13]
- Generations 6-7: Nintendo 3DS (top screen is 800 × 240, bottom screen is 320 × 240 pixels) [14]

Note that Generation 1 games were playable on the Gameboy Colour, resulting in coloured sprites; however, there is still a distinct difference in art styles between generations 1 and 2 as shown previously.

The advancement of mobile device screens over the years has improved the quality of images present in video games. Our CNN will pick up on these discrepancies and use it to distinguish between consoles. This limitation in the training data must be acknowledged as the network will not solely look at the artwork from a purely stylistic perspective.

Regarding data, we are limited to official sprites released by Nintendo to ensure consistency in the art design. By Generation 7, there are only 808 Pokémon [15] in the Pokédex; many of whom were introduced in previous generations (a potential issue; see Risk Register). As such, our datasets are relatively small compared to other projects.

Project Plan

Link to detailed project plan: **Click Here**

We plan to meet up once twice every week to provide update and ensure tasks are completed on schedule. Our primary form of communication will be through Messenger. To prevent overwriting each other's code, we will upload our program to GitHub periodically to handle version control, and monitor each other's progress. An alternate option is to use Google Colab's version history to view changes that were made to code.

Links

Link to Google Colab Document: **Click Here**
Link to GitHub: **Click Here**

Risk Register

Risk	**Solution**	**Concern Level**
The neural network may not be able to differentiate sprites between consoles. For example, the games released on the Gameboy Advance and DS have similar art styles.	Increase our network's complexity or combine generations together.	**Large risk:** We ourselves have trouble distinguishing between certain sprites.
Certain Pokémon have more sprite samples across platforms whereas newer Pokémon only have sprites drawn in one particular way. This imbalances the dataset and may result in overfitting onto Pokémon with more image samples.	We can perform image augmentation using Keras to produce new images for our dataset- reinforcing the traits of certain classes that may have low accuracy.	**Moderate risk:** We are new Image augmentation but it will allow us to grow our data set while maintaining trait consistency.
Training a CNN network can be quite time consuming if one is to implement good programming practices such as state saving. This project utilises a comparable dataset size and will likely be more resource intensive due to added complexities and optimizations.	This risk can be mitigated through proper time management.	**Moderate risk:** Our project plan is designed to give us ample time to produce a working model.
In the event that this particular project does not work at all, we need to shift to a new perspective regarding how to utilise our collected data to solve a problem.	Another interesting problem is determining the Pokémon type of a particular Pokémon based on sprites. This is a simpler problem as Pokémon types usually correlate to distinct features in the art design (i.e. fire types have flames; flying types have wings).	**Large Risk:** Since this project has not been done before, there is a good chance that the idea will fail, in which case, We will effectively have to start from scratch.

Table 1: Risk Register

References

[1] J. Feldman, “What Makes a Pokémon Legendary?,” DataCamp, 2020. [Online]. Available: https://www.datacamp.com/projects/712. [Accessed: 05-Jun-2020].

[2] Y. alouini, “Pokemons Machine Learning 101,” Kaggle, 10-Sep-2018. [Online]. Available: https://www.kaggle.com/yassinealouini/pokemons-machine-learning-101. [Accessed: 05-Jun-2020].

[3] A. Dennanni, “Creating Pokémon with Deep Learning,” Medium, 29-Oct-2018. [Online]. Available: https://medium.com/neuronio/creating-pokemon-with-artificial-intelligence-d080fa89835b. [Accessed: 05-Jun-2020].

[4] Hemagso, “neuralmon,” GitHub, 28-Feb-2017. [Online]. Available: https://github.com/hemagso/neuralmon/blob/master/sprites/pokemon/centered-sprites.zip. [Accessed: 05-Jun-2020].

[5] BulbaBot, “Bulbagarden Archives,” Moltres - Bulbagarden Archives, 13-Nov-2005. [Online]. Available: https://m.archives.bulbagarden.net/wiki/Category:Moltres. [Accessed: 05-Jun-2020].

[6] J. Brownlee, “Image Augmentation for Deep Learning With Keras,” Machine Learning Mastery, 12-Sep-2019. [Online]. Available: https://machinelearningmastery.com/image-augmentation-deep-learning-keras/. [Accessed: 05-Jun-2020].

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Choosing the right estimator,” scikit learn, 2011. [Online]. Available: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html. [Accessed: 10-Jun-2020].

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Support Vector Machines,” scikit learn, 2011. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#classification. [Accessed: 10-Jun-2020].

[9] “Generation,” Bulbapedia, the community-driven Pokémon encyclopedia, 29-May-2020. [Online]. Available: https://bulbapedia.bulbagarden.net/wiki/Generation. [Accessed: 10-Jun-2020].

[10] “Game Boy,” Wikipedia, 13-Jun-2020. [Online]. Available: https://en.wikipedia.org/wiki/Game_Boy#Technical_specifications. [Accessed: 13-Jun-2020].

[11] “Game Boy Colour,” Wikipedia, 13-Jun-2020. [Online]. Available: https://en.wikipedia.org/wiki/Game_Boy_Color#Specifications. [Accessed: 13-Jun-2020].

[12] “Game Boy Advance,” Wikipedia, 06-Jun-2020. [Online]. Available: https://en.wikipedia.org/wiki/Game_Boy_Advance. [Accessed: 13-Jun-2020].

[13] “Nintendo DS,” Wikipedia, 13-Jun-2020. [Online].
Available: https://en.wikipedia.org/wiki/Nintendo_DS. [Accessed: 13-Jun-2020].

[14] “Nintendo 3DS,” Wikipedia, 10-Jun-2020. [Online].
Available: https://en.wikipedia.org/wiki/Nintendo_3DS. [Accessed: 13-Jun-2020].

[15] “Regional Pokédex,” Bulbapedia, the community-driven Pokémon encyclopedia, 03-Mar-2020. [Online]. Available: https://bulbapedia.bulbagarden.net/wiki/Regional_Pokédex. [Accessed: 10-Jun-2020].