APS360

Classifying Pokémon Generations from Sprites

Team 10, Progress Report

Due on July 12, 2020

Professor S. Colic

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

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

Brief Project Description

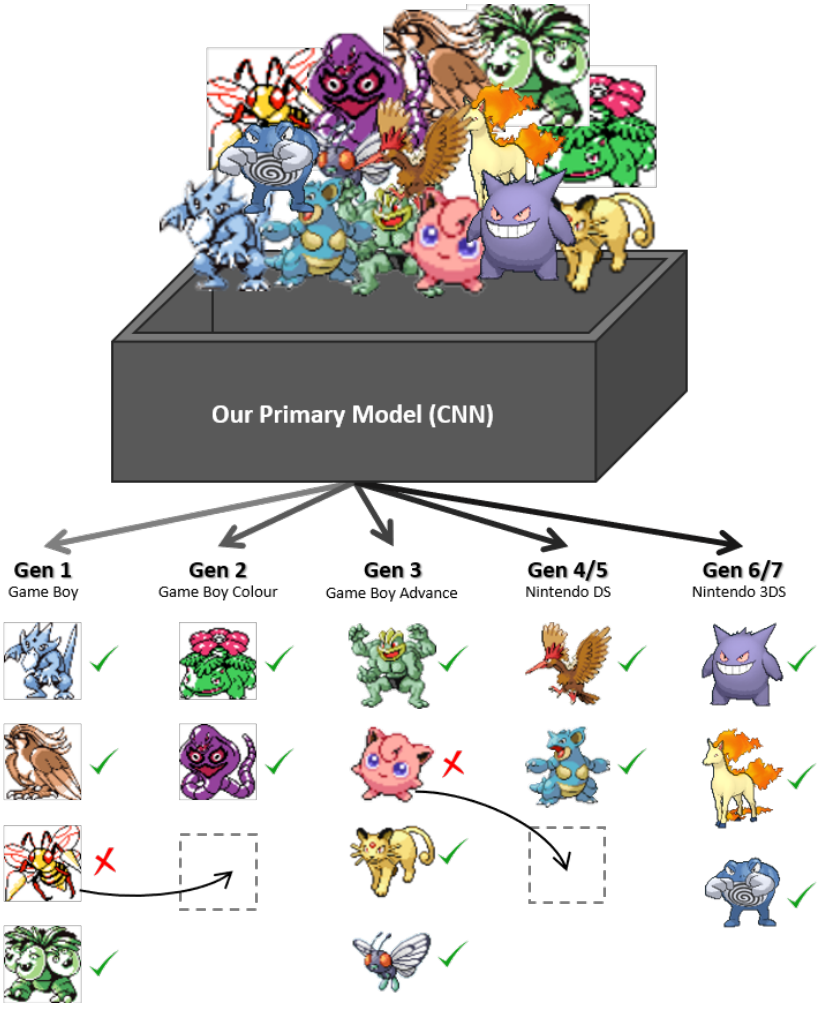

Figure 1: High-Level Visual Representation of our project and goal. Sprites of Pokémon across the years scraped/collected from databases are fed through our CNN which will be able to classify them on the basis of varying art-styles!

Project Goal	This project aims to be able to identify Pokémon sprites from various generations of games. The classes we have chosen are based on the hardware that each game was released for. Additionally, we hope to create a baseline to compare and show that machine learning was a reasonable solution to this classification problem.
Motivation	There has been little (almost none, in fact) work done in classifying Pokemon in this way. As many of us have grown up with Pokemon in our childhoods, we find this to be an interesting topic to apply machine learning. We may learn something new about the nature of each generation's artwork and design. This problem could also be extended to other forms of game hardware identification in the future.
Machine Learning Appropriateness	Currently, there are no classification projects such as this related to machine learning. This project is a perfect application for machine learning due to the nature of image processing and complex classification problems. A machine learning algorithm will likely be able to learn features and trends that a human or other types of algorithms would not be able to distinguish.

Data Processing

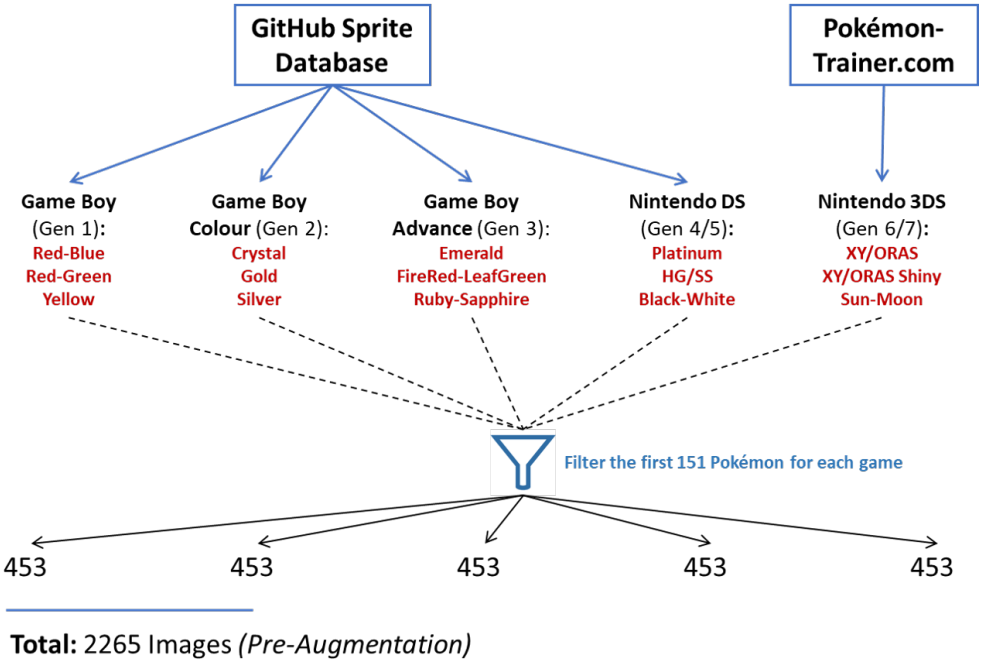

Figure 2: To create a preliminary balanced data set for initial training, the first 151 sprites from all 3 games (in red) in each were extracted and for a total of 2265 images (prior to augmentation).

Figure 3: Sample Charizard (#006) and Pikachu (#025) sprites Prior to any augmentation performed.

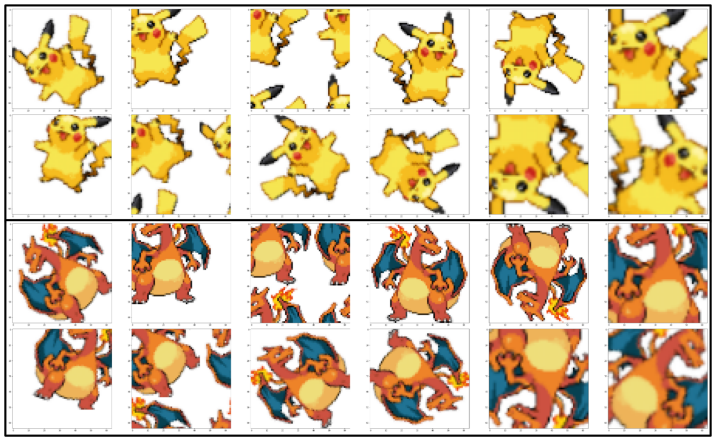

Figure 4: Sample Charizard (#006) and Pikachu (#025) sprites with 12 sets of augmentations performed. The transformations shown include: rotations, translations, wrapped translations, vertical flips, horizontal flips, cropping, and several combinations of these. **The augmentation script can be accessed here.**

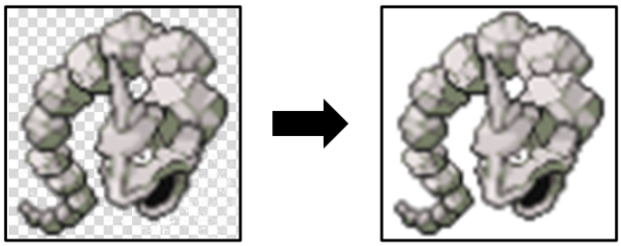

Figure 5: Within the dataset we are working with, some sprites have a solid white background and other sprites like Onix (#095) have a transparent background. To maintain consistency, images were run through a script that removed their alpha transparency layer, leaving all images with a solid white background.

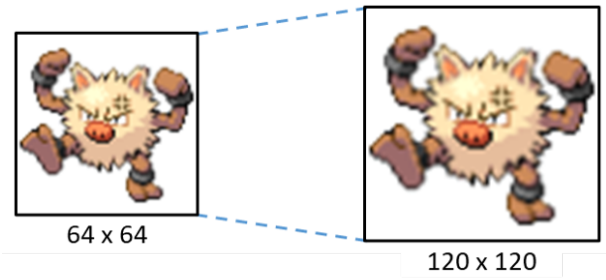

Figure 6: Within the dataset we are working with, Sprites range from a size of 64 x 64 to 120 x 120. Instead of down-scaling the higher quality images, we are up-scaling the smaller sprites like Primeape (#057) a size of 120 x 120 pixels to maintain consistency across the data.

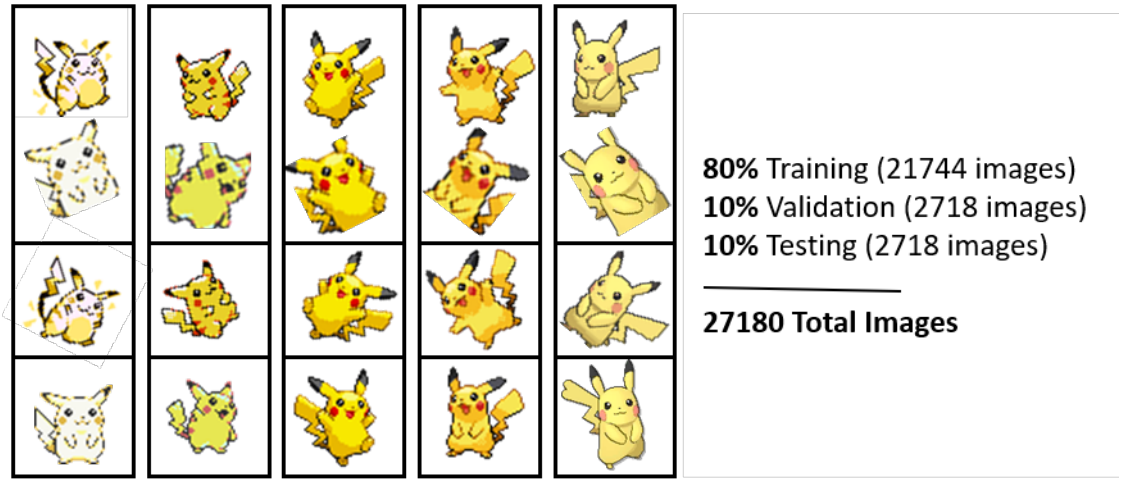

Figure 7: Once the 12 augmentation combinations are performed on the 453 images from each class, we have a total of 453 * 12 * 5 = 27180 images. The initial data split is 80% training (21744 images), 10% validation (2718 images), and 10% testing (2718 images). There are instances of both original and augmented images from each class placed in each of these 3 datasets. We are still experimenting with more variation within the augmentation but are refraining from altering the original artstyle through adding noise and editing hue/saturation.

Baseline Model

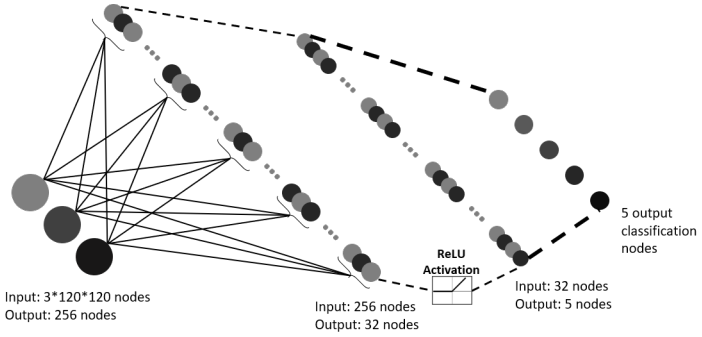

Figure 8: Visual Representation of Baseline ANN Architecture;
Total Parameters $= (3 \times 120 \times 120 \times 256 + 256) + (256 \times 32 + 32) + (32 \times 5 + 5) = 11067845$.
Layers: 3 fully-connected layers

Primary Model

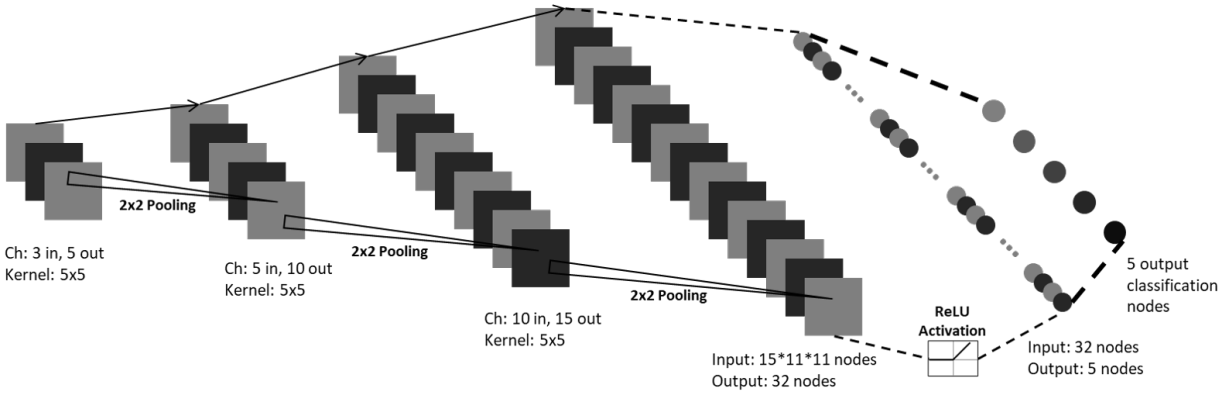

Figure 9: Visual Representation of CNN Architecture;
Total Parameters $= (3 \times 5 \times 5 \times 5 + 5) + (5 \times 10 \times 5 \times 5 + 10) + (10 \times 15 \times 5 \times 5 + 15) + (15 \times 11 \times 11 \times 32 + 32) + (32 \times 5 + 5) = 63682$.
Layers: 4 convolutional layers, 3 (2×2) max-pooling layers, 1 fully-connected layer

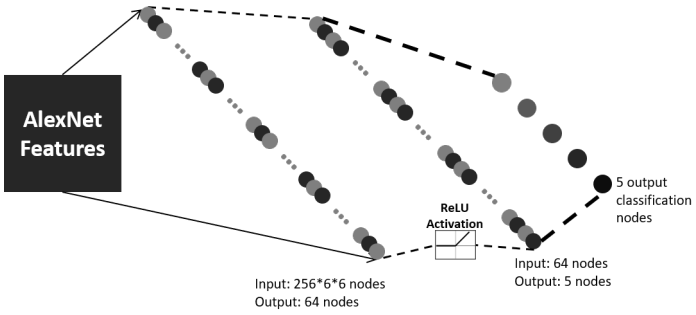

Figure 10: Visual Representation of Transfer Learning Architecture from AlexNet (alternate primary model);
Total Parameters $= (256 \times 6 \times 6 \times 64 + 64) + (64 \times 5 + 5) = 590213$.
Layers: AlexNet Feature Maps, 2 fully-connected layers

Results

Baseline Model

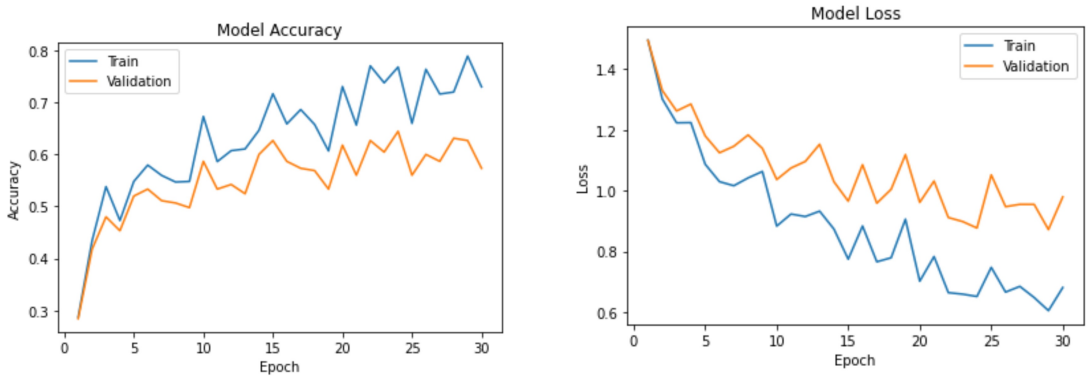

Figure 11: **Final Training Accuracy:** 0.7298, **Final Validation Accuracy:** 0.5733, **Final Training Loss: 0.6807**, **Final Validation Loss:** 0.9791. The naive ANN baseline model does rather poorly on the validation set.

True \ Prediction	**1**	**2**	**3**	**4/5**	**6/7**
1	126	202	7	22	5
2	3	350	3	5	1
3	0	2	219	131	10
4/5	0	0	27	321	14
6/7	0	0	8	49	305

Table 1: ANN Training Confusion Matrix

True \ Prediction	**1**	**2**	**3**	**4/5**	**6/7**
1	11	22	4	6	2
2	1	37	4	3	0
3	0	1	19	23	2
4/5	0	1	12	30	2
6/7	0	0	1	12	32

Table 2: ANN Validation Confusion Matrix

Primary Model

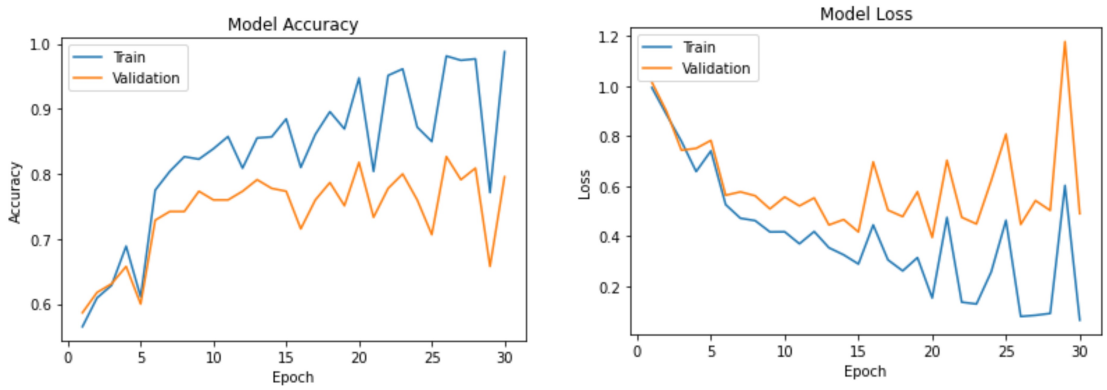

Figure 12: **Final Training Accuracy:** 0.9878, **Final Validation Accuracy:** 0.7956, **Final Training Loss:** 0.0636, **Final Validation Loss:** 0.4899. Upgrading to a CNN model allows our network to analyze spatial and geometric similarities in the pictures; this boosts the validation accuracy to 80%.

True	Prediction					
		1	**2**	**3**	**4/5**	**6/7**
	1	362	0	0	0	0
	2	1	361	0	0	0
	3	1	0	358	3	0
	4/5	3	0	14	345	0
	6/7	0	0	0	0	362

Table 3: CNN Training Confusion Matrix

True	Prediction					
		1	**2**	**3**	**4/5**	**6/7**
	1	42	0	3	0	0
	2	4	39	2	0	0
	3	6	0	27	12	0
	4/5	1	0	18	26	0
	6/7	0	0	0	0	45

Table 4: CNN Validation Confusion Matrix

Transfer Learning Model

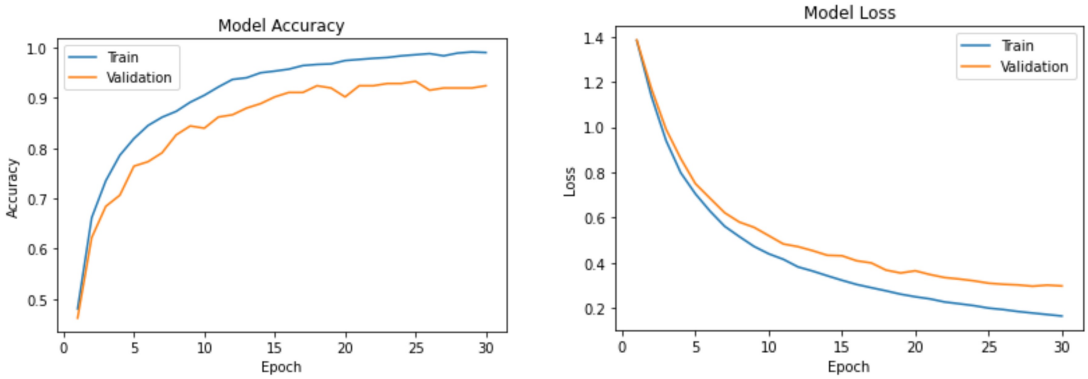

Figure 13: **Final Training Accuracy:** 0.9906, **Final Validation Accuracy:** 0.9244, **Final Training Loss:** 0.1644, **Final Validation Loss:** 0.2975. Using pre-trained AlexNet features to classify our images increases the validation accuracy to 92%, which is a major improvement. There are also significantly less fluctuations in the curve when compared to the CNN.

True \ Prediction	**1**	**2**	**3**	**4/5**	**6/7**
1	359	3	0	0	0
2	1	361	0	0	0
3	0	0	351	11	0
4/5	0	0	2	360	0
6/7	0	0	0	0	362

Table 5: AlexNet Training Confusion Matrix

True \ Prediction	**1**	**2**	**3**	**4/5**	**6/7**
1	43	2	0	0	0
2	1	44	0	0	0
3	1	0	36	8	0
4/5	0	0	5	40	0
6/7	0	0	0	0	45

Table 6: AlexNet Validation Confusion Matrix

Discussion

These models were only trained using the first 151 Pokémon in each class. With a 80:10:10 dataset split, there were 1810 sprites in the training set and 225 in the validation set (we chose to experiment with pre-augmented data first). After augmentation, we expect to resolve the over-fitting issues that our models present while trying to achieve higher validation accuracies in our CNN and transfer learning model.

Looking at the confusion matrices, all of our models consistently misclassify Gen 1 with Gen 2, and Gen 3 with Gen 4/5 (and vice-versa), which is an expected result due to the stylistic similarities in the sprites. Gen 6/7 (the Nintendo 3DS) is classified almost perfectly in the CNN and the AlexNet models, likely due to higher-quality and vastly-different sprites in the Nintendo 3DS.

Project Progress

- Link to updated Project Plan: **Click Here**
- The workload was fairly distributed amongst group members and internal deadlines were met
- We continue to use and update **GitHub** and **Google Colab** to help us with version control and track edits to code
- Our form of communication has been and will continue to be through a Messenger group chat
- Link to Project Proposal for reference: **Click Here**