Project Topic

For this project, we chose to analyze Pokemon to see if we could accurately classify a Pokemon's type based on their appearance. For instance, if a Pokemon is blue and has fins, could we reasonably state that it's an indication they would be a water type? This was a supervised categorical classification problem. We hope to learn more about image classification, and get a bit of insight into how well these models can run; pokemon in general tend to be very different looking and so we hope to learn a lot about how well it can pick up on specific features, especially with the implementation of GradCam.

Data

We are using the Pokemon Image Dataset here. There are two primary documents here, the Pokemon.csv and the Images folder. The images folder contains a picture of each Pokemon taken from their 3D rendered model rather than their sprites so that there is consistency across old and new Pokemon. The Pokemon.csv just contains the Pokemon's name, its primary type, and its secondary type if it has one. Not all Pokemon have two types.

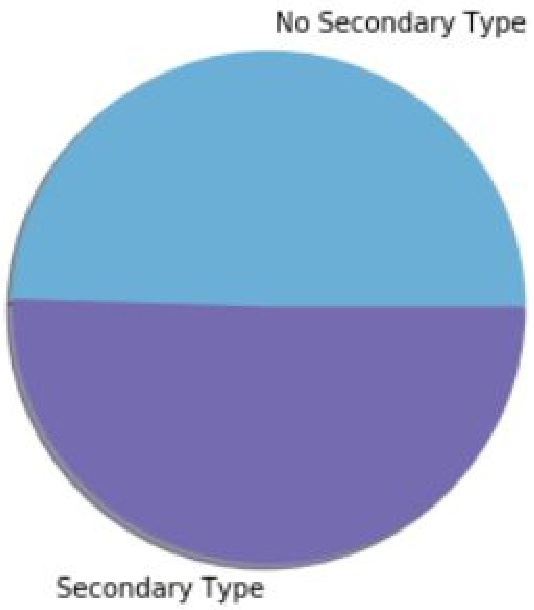
There are 809 unique pokemon. For the pictures, it was initially a mix of .pngs and .jpgs. They both are 120 x 120, but the .pngs have 4 channels, whereas the .jpgs have 3.

Data Cleaning

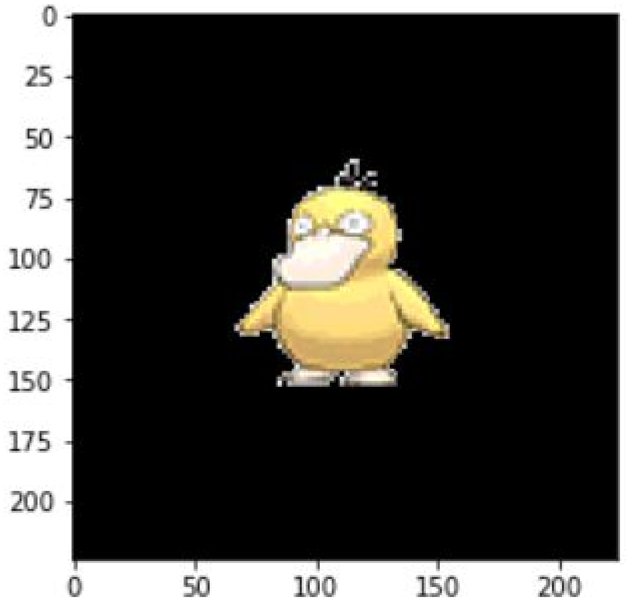
For our data cleaning, after converting our list of images to a nparray, there was an additional column in our data which needed to be dropped because it was not relevant for the analysis. We also One Hot Encoded all of the primary types. This was to make the categorical data work in our selected model. We decided to drop the secondary type in our testing because given the nature of our dataset, it wouldn't have been meaningful to impute. Additionally, 50% of our pokemon have a secondary type, so imputing that could have led to false conclusions. We made the assumption that a Pokemon's features would likely correlate more to their primary type anyways. Additionally, we converted all of the images to RGB and to .jpgs so they would be consistent in type and number of channels. We also expanded the dimensions so that it fit with the model and normalized all of the images in case there was a discrepancy in image size.

Exploratory Data Analysis

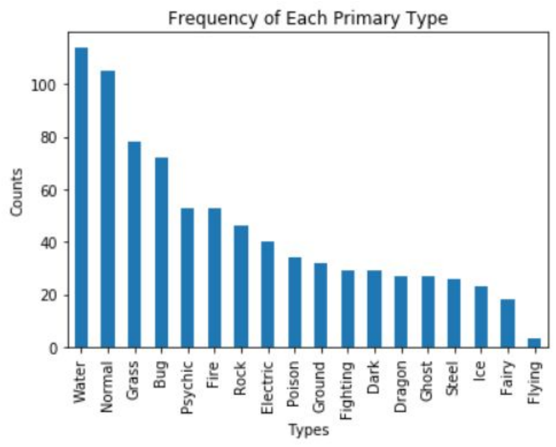
We first wanted to see the breakdown of how many singly typed pokemon there were versus how many dual types there were to see how we should approach whether or not to account for dual types in our model.

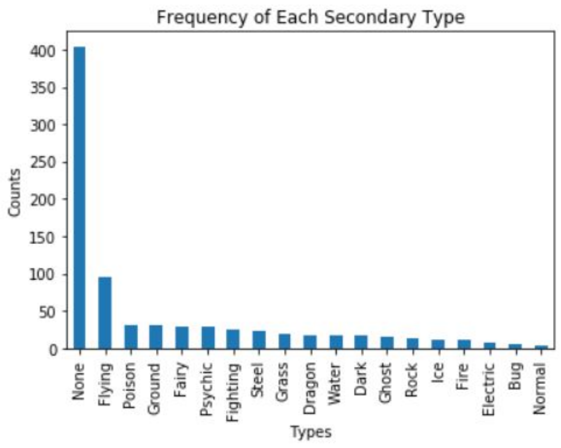


We found that it was a roughly 50/50 split for Pokemon with and without a secondary type. We also visualized the image itself to get an understanding of what it looked like.



Then, we took a look at the distribution of Pokemon by primary type as well as by secondary type to see if we should take additional preprocessing steps. The figures below just show a histogram of Pokemon by type and by secondary type.

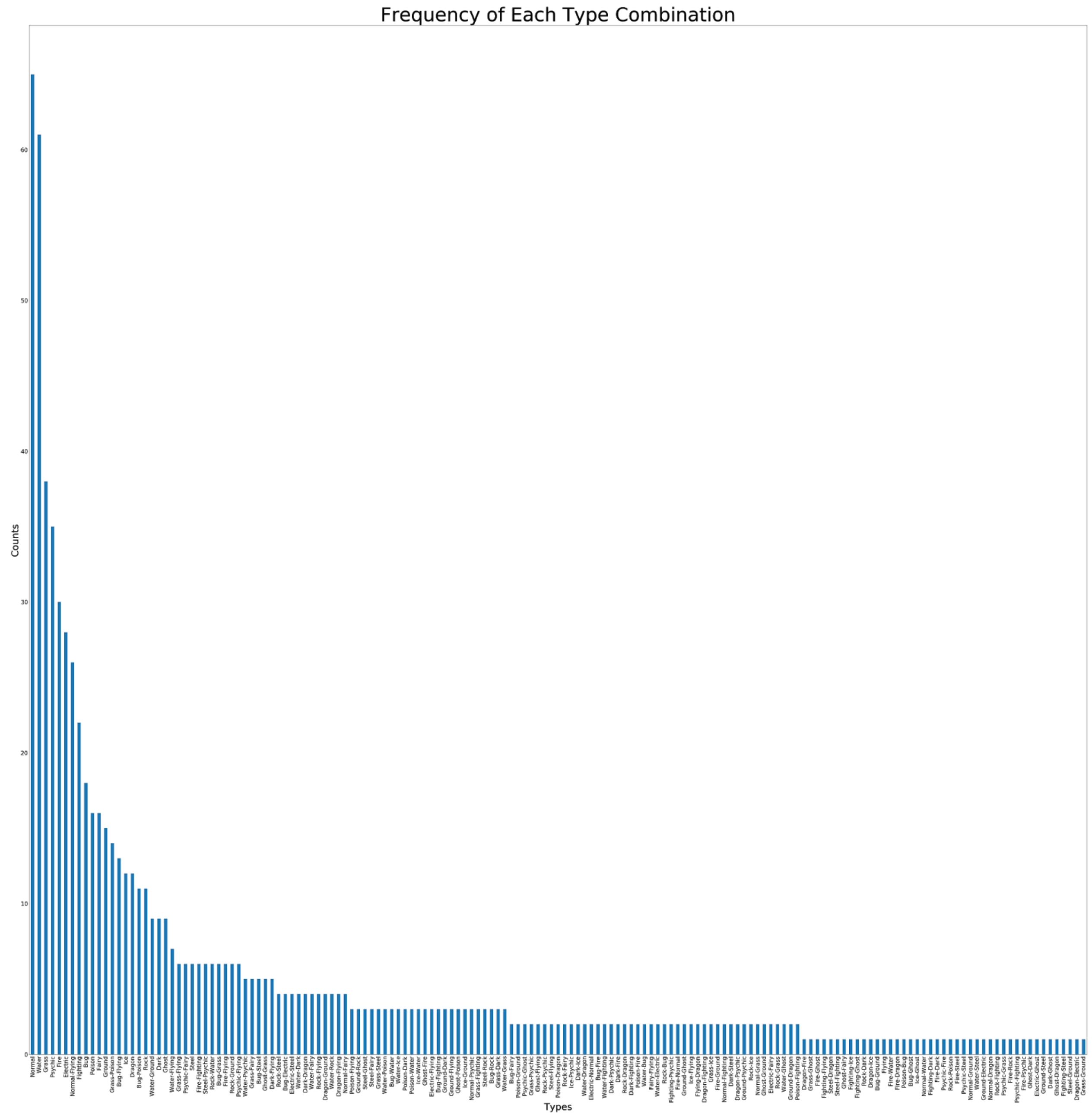




Interestingly enough, Flying is very rarely a primary type and almost exclusively a secondary type where it makes up the most secondary types. Unfortunately, because we are only categorizing the primary type, this will end up getting overlooked by our model despite probably having very strong common traits such as beaks and wings.

We also had the option to classify the pokemon based on their type combination which would mean the combination of their primary and secondary types. This had the potential of allowing us to capture some of the interesting features in Flying types for example. In order to do this, we made a new column in our dataframe called “Type” which combined the Type1 and Type2 columns into one. Unfortunately, when we visualized the data, we realized that there was only one pokemon with many of the combinations. As a result of this, we decided to focus instead on the primary types rather than the combination, because the model would be better able to classify the pokemon and because it isn’t especially helpful to develop a model that can correctly identify the one pokemon with a type combination.

Also as can be seen in the figure, the distribution is incredibly uneven which would likely lead to the model just assigning each pokemon the Normal type as it is the most common. Thus we decided to focus on just the Type1 information in the dataset.



Furthermore, we decided to run a chi^2 test and a t-test on the primary and secondary types to see how they relate to one another. The chi^2 test should allow us to see if the primary and secondary types are independent or dependent, while the t-test will help us to understand if the types follow similar distributions or not. In order to accomplish this, we mapped the type1 and type2 columns to categorical variables in order to make them into integers rather than strings. In the chi^2 test we got a p-value of 0.000, which signifies a very high chance that the columns are dependent. This makes sense given the fact that certain type combinations are more frequent than others as seen in the visualization from earlier in this section. From the t-test we also had a p-value of 0.000 which signifies that the variation in the distribution of the types is unlikely to be coincidental. This also makes sense given the fact that primary types and secondary types do not appear to be distributed in the same manner as seen earlier in this section, and also because there are certain types (like Flying) that are significantly more common as secondary types.

The fact that the types are dependent on each other provides further support for the idea that we should only use the first types in training and testing the model in order to avoid any collinearity issues.

Models

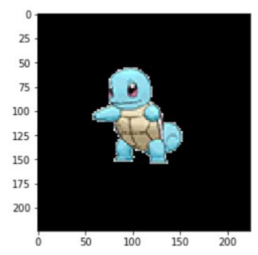
We used a Convolutional Neural Network for our image classification. We selected a CNN because they are the best suited model for image classification. Our model has 9 layers, with a softmax output layer. We also added two max pooling layers which should help to reduce overfitting. We have three convolutional layers in our model because in our early testing of comparing the results from a CNN with 1,2, and 3 convolutional layers, we found that the third produced the best accuracy. Our model also utilizes Early Stopping to optimize our gains from the epochs in place of tuning them by hand. This allows us to stop the model when it sees that the accuracy starts to plateau. Because we chose to only focus on the primary type for our analysis, collinearity is not a problem. We saw from our chi^2 test that the primary and secondary types were dependent on each other which is why we selected only Type1for our analysis, thereby preventing any collinearity/interaction issues we may have otherwise encountered. We spent some time processing the images so that they would be best suited to go into our model, and while it did not make much sense to run PCA given that this is image classification we did engineer our features so that they would run in our model.

We started by running our model with some default hyperparameters. For our cross validation and hyperparameter tuning we used GridSearchCV to optimize our batch size and optimizer by testing a series of different options for hyperparameters opertimizers. We used 5-fold cross validation. Then we printed out the best performing model and determined that we should use a batch size of 32, and the relu activation function, as well as the adam optimizer. Thus we reran our model with these newly tuned hyper parameters. After our revised model ran, we analyzed its decisions using GradCam to see what parts of each image were being used to classify the type. GradCam is a really interesting technique that we did not really cover in class; it produces a map to highlight the most important regions in the image for predicting a specific class which ends up looking somewhat like a heatmap. By using this technique we could see what features the model was focusing on for each type. This also gives us an understanding of which features are the most important within the image classification.

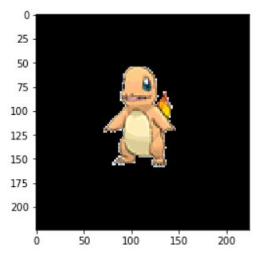
Results and Analysis

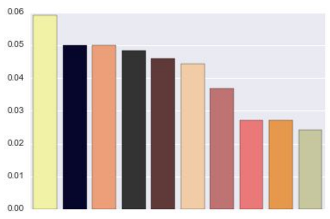
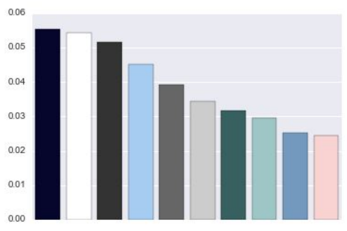
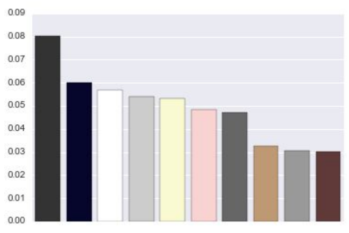
Our model did not classify the pokemon's types with a high level of accuracy. We had a final categorical accuracy of 0.1034, after tuning our hyperparameters and running cross validation. Given that there are 18 categories in the dataset we are performing better than just a randomized model which would have an accuracy of around 0.05. This does make some sense though, as we learned during this analysis pokemon of similar types do not necessarily look very similar. For example, there are around 34 different dog-like pokemon and they're almost all different types, so if the model was picking up on dog features to classify the pokemon based on type it would be unsuccessful. Additionally interestingly, we decided to pull out some of the classifications to see what the model predicted, and found that Squirtle and Charmander, which are two of the first pokemon ever created and are water and fire types respectively, were both classified as Normal type pokemon. To us this might seem silly, they are different colors, their colors match the color most commonly associated with their type and they have different body designs, so they should obviously be classified as different types of pokemon. However, this assumption would not take into account the other 3d models associated with those types and thus we see these two classified as normal.

```
array(['Normal'], dtype=object)
```

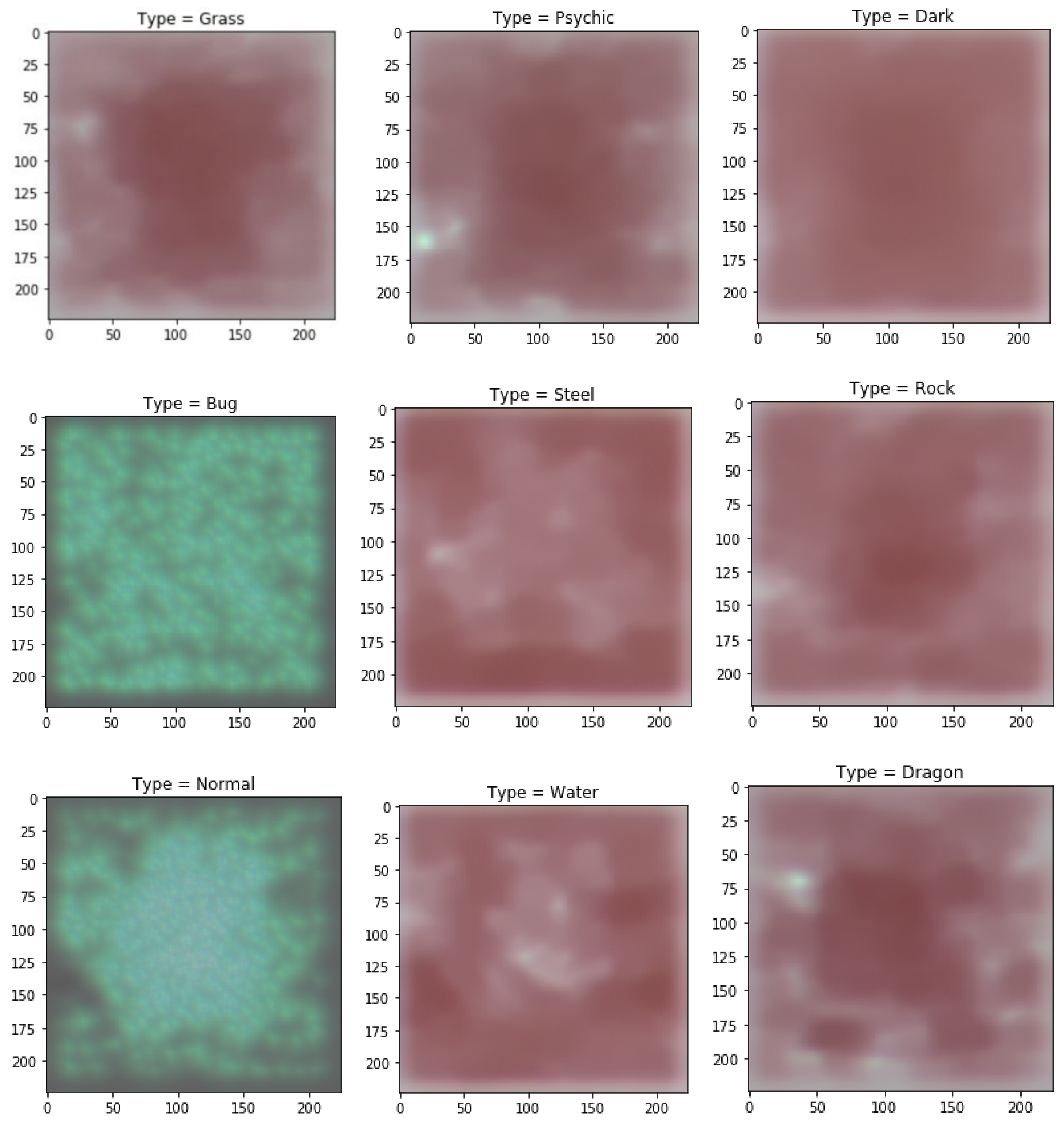


```
array(['Normal'], dtype=object)
```



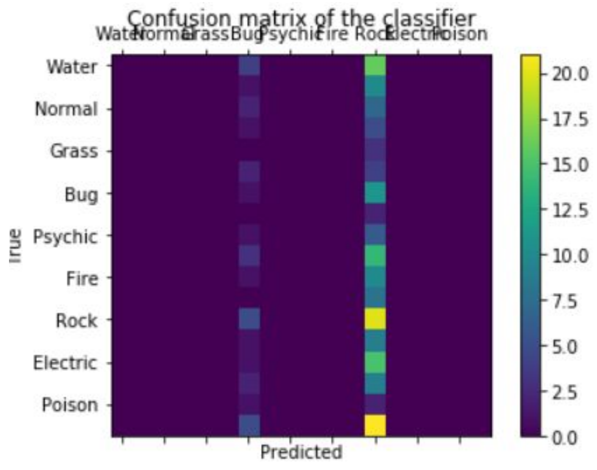




Below the character models we have included the graphs for the color breakdowns by type of (from left to right) fire, water, and normal found in this analysis. From these graphs we can see that black is very prominent in all three, and the primary colors of these two models (light blue and orange) are not even the most common for their types, so their classification as normal is not as surprising. Below are the GradCam results for 9 of our classes.



These are averaged across each type, rather than selecting an individual pokemon because in this way we should be able to understand what the model is looking for to identify a type. From these heat maps we can get a sense for why our model was not as successful as we had hoped, the heat maps are not clear for any of the types. Clearly the model is not picking up on any specific features that predict the average instance of any of the types.

As well as using accuracy we decided to make a confusion matrix for the data which is pictured below. Interestingly, it seems that we are predicting bug and rock much more than we should actually be.



Discussion and Conclusion

From this project we saw that our model was somewhat successful in classifying the pokemon by type, achieving results that are definitely better than random, but not as high as we hoped. Despite this, the project was super interesting. Our team learned a lot about GradCam and how it can be used to peer under the hood of an image classification model such as this one in order to determine what features the model is picking up on. Also, we learned a lot about our dataset in particular and how we can apply this understanding in the future. We gained a better understanding for the extreme variation in models for pokemon within a type, and now have a deeper understanding of how to create a model for image classification. This didn't work as well as expected, but that appears to be mostly as a result of the variation in each class rather than as a fault of the model itself. When we used image classification in the last Kaggle to classify street signs there was still variation but within each sign there were specific features that were held constant, which is not true of our dataset. When a Dragon type pokemon can look like both Goomy and Druddigon (pictured below) it does seem fair that the model would not have the highest accuracy level.

In the future it could be really interesting to analyze some other games that have types and see if they are more consistent than pokemon. For example, we could look at Digimon and see how their character models predict type, or how Yu-Gi-Oh's models match up with monster types.

Alternatively, we could apply the model we developed in this project to different datasets to see how it would fare with dog breed classification or the classification of celebrities based on their images. Furthermore, it could be interesting to break down the pokemon by the generation they are in and analyze them separately to see if individual generations are more consistent than the dataset in general.

GitHub: https://github.com/pickettmap/Pokemon-Type-Classification.git
Presentation:
https://docs.google.com/presentation/d/1ilKGQHhFs8JDMSDTiTBsQEC0kNLpbf63znyQRn7H5Xc/edit?usp=sharing