Gotta Class Em' All: Building a Pokédex

COS 429: Final Project

Colton Bishop & Pulkit Singh

1 Introduction

Pokémon is a Japanese anime show that has become popular all over the world. The Pokémon universe features "creatures of all shapes and sizes who live in the wild or alongside humans", that are "raised and commanded by their owners", called trainers.[4] A Pokédex is a device in the series that is able to identify the Pokémon from an image, and in this project, we wanted to build a similar classifier for the original 150 Pokémon in first generation of the series. These Pokémon we aim to classify and a Pokédex is visualized below.

Figure 1: First generation Pokémon

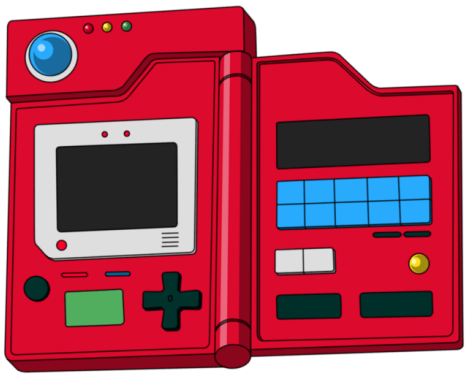

Figure 2: Pokédex

To begin with, we found a dataset on Kaggle, trained a CNN classifier on it and evaluated results. Our original dataset turned out to be extremely imbalanced and unsatisfactory, so we decided to collect additional data and make a new, more balanced dataset. After scraping, cleaning and manually inspecting all the new images, we were able to put together the most extensive dataset for Pokémon classification that we could find. Following this, we retrained a classifier and ran addditional experiments in order to build a Pokédex.

All code for the project is submitted with the final report - organized into two main folders for building the dataset and training the classifiers respectively. To demonstrate the structure of the collected dataset, we have included one Pokémon class (both training and

validation). A list of all libraries and resources used is attached to the end of this report. The report follows the structure of the project as it was undertaken.

2 Finding a Dataset

There are not many options for Pokémon data, so we were only able to find one dataset on Kaggle that had 10000+ images for each of the 150 Pokémon in the first Generation of the show. There is one folder for each of the Pokémon in the dataset, and each folder contains the relevant image files in jpg format. Some examples of training images and labels are visualized below.

Figure 3: Label=Gloom

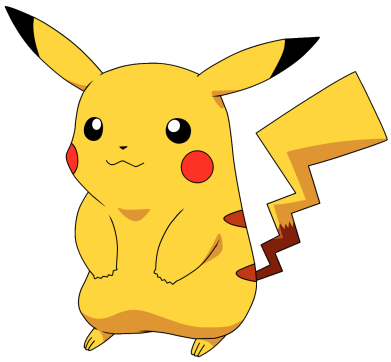

Figure 4: Label = Pikachu

Figure 5: Label = Mewtwo

This seemed promising, however when we started exploring the dataset we found that there was a huge discrepancy between the number of images between classes. Some Pokémon had 300 images while some only had 30, with the majority having around 50 images each. A visualization of the number of images per Pokémon class is below.

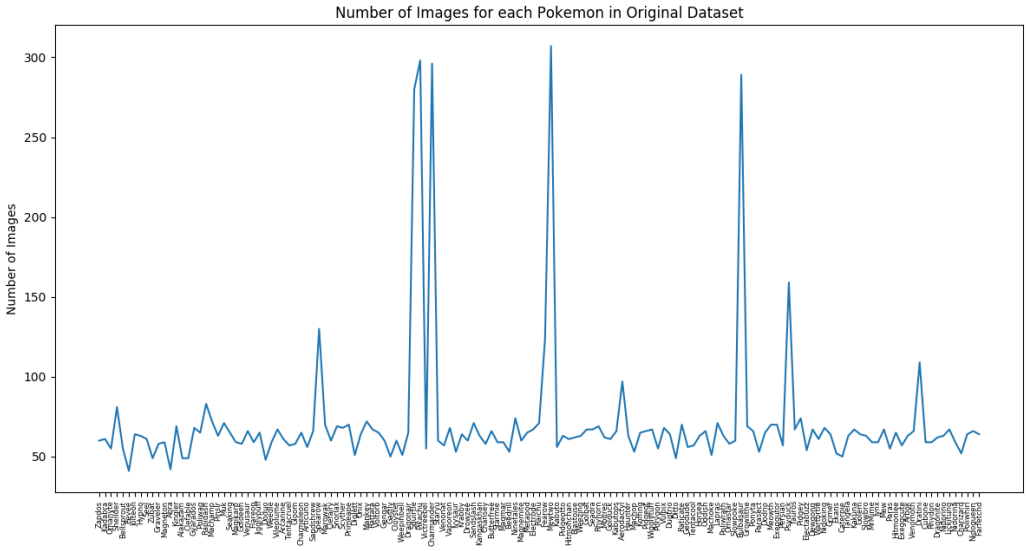

Figure 6: Number of Images across Pokémon classes

This exploration demonstrated that we would probably need to balance our dataset if we wanted to train a classifier on the full 150 Pokémon and build a functional Pokédex. However, in order to establish a baseline, we decided to train a classifier on this data

In order to make the data usable, we had to perform some cleaning. We ran a script that verified the readability of each file using the Python Imaging Library, removed unreadable files, standardized filenames, and finally shuffled and divided the dataset into training and test sets. We decided on a 80% training to 20% test set ratio because the dataset was already small to begin with, and we wanted to maximize the number of training images the model got. In addition, we had no hyper-parameters we wanted to tune, since our classifier was quite simple, so we decided against having a dev set.

3 Building a Classifier

3.1 Specifying an Architecture

Since the task at hand is a relatively standard object classification exercise, we decided that using a Convolutional Neural Network would be the best approach. The All Convolutional Net (all-CNN) architecture was proposed by Springenberg, Dosovitskiy, Brox and Riedmiller in order to create a simple but powerful architecture using the principles that the vast majority of modern CNNs are built on. They came up with a "a homogeneous network solely consisting of convolutional layers, with occasional dimensionality reduction by using a stride of 2", and were able to achieve state-of-the-art performance without the need for "complicated activation functions" or "max-pooling".[2] The simple and parameter-efficient

approach of this network makes it the ideal classifier for the project, as we had limited time and compute resources at our disposal.

3.2 Training the Convolutional Neural Network

We decided to work in Keras since it's a convenient high-level interface that works well for image processing. For our data pipeline, we uploaded the data to Google Drive, and trained our models on Google Colab. This workflow allowed us to directly access our data by using Google's native integration between Colab and Drive, while leveraging Colab's free GPU accelerated compute resources. We train the model for 100 epochs, using categorical-crossentropy for our loss metric, and Adam [1] as our optimizer. The training and validation accuracy and loss across epochs is illustrated below.

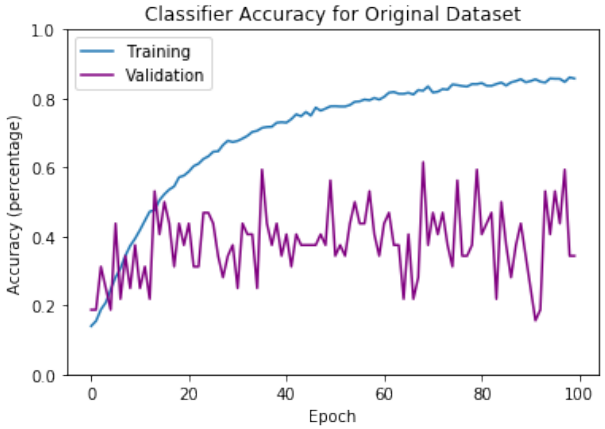

Figure 7: Accuracy across Epochs

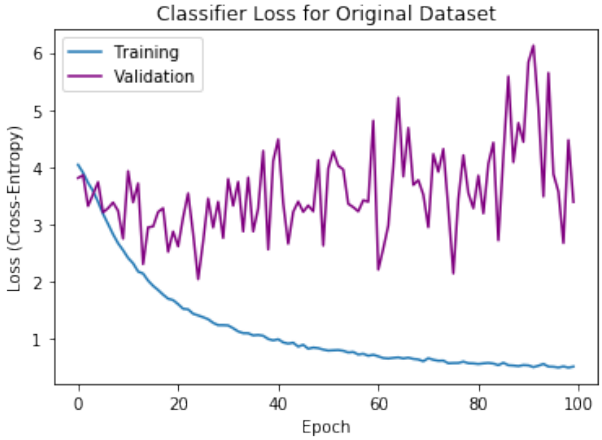

Figure 8: Loss across Epochs

As we can see from the graphs, the training accuracy increases smoothly, while the training loss decreases smoothly as usual. However, the validation performance fluctuates a lot, both in terms of accuracy and loss. We're uncertain why this is the case, but one reason could be that there is great intra-class variability within the dataset, because it is difficult to collect large amounts of Pokémon data in general. This variability could be contributing to the mismatch between training and validation performance as the weights change. The maximal training accuracy obtained is about 86% while the maximal validation accuracy obtained is about 60%.

3.3 Evaluating the Results

As mentioned above, our classifier obtained an average accuracy of about 60%, but we wanted to dive deeper and look at class-specific performance, since we were concerned about the imbalance of our dataset. We computed accuracy for each individual class, and aggregated our results to obtain a distribution of number of Pokémon classes across accuracy ranges. These results are visualized below.

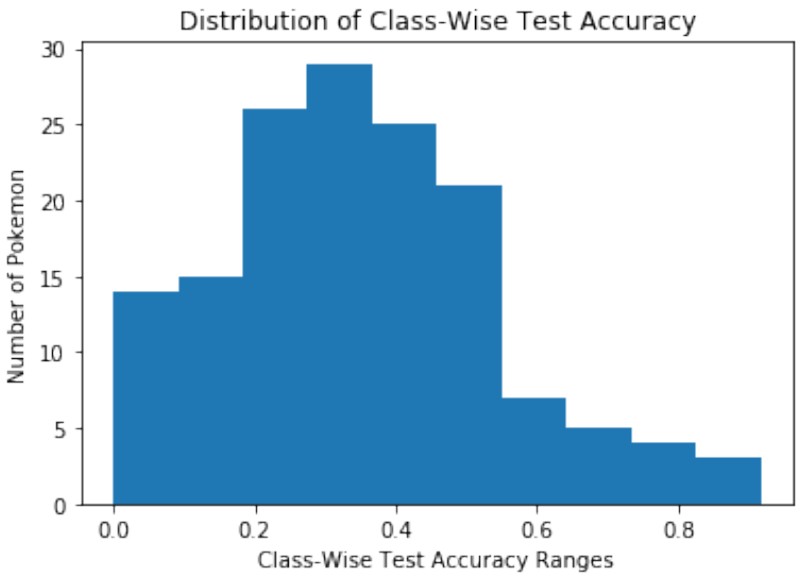

Figure 9: Distribution of Test Accuracy of Pokémon

The figure illustrates how the vast majority, about 80%, of Pokémon classes have a test-accuracy of under 50%. Very few Pokémon classes have greater than 50% accuracy. However, our average performance is much higher at about 60%. Our intuition is that the small number of classes with large amounts of images are being classified well, while the majority of classes with small amounts of images are being classified poorly, leading to a higher non-representative average test accuracy. In order to test this, we plot the number of training examples against the test performance for each class, as shown below.

Figure 10: Visualizing relationship between Number of Training Images and Test Accuracy across Pokémon Classes

The figure above illustrates how the few classes with the most training images end up having substantially better test performance while those with fewer training examples tend to have lower performance. We also ran a Pearson correlation between the number of training images and test accuracy across classes and got a correlation value of 0.49 and p-value of $1.93 \cdot e^{-10}$, which indicates a definite positive correlation between the two variables. This lines up with our intuition about the imbalanced dataset leading to a classifier that has performance skewed to only a couple of classes. This motivates the need to build a better dataset, which we work on in this project.

4 Building a New Dataset

It is challenging to build a a high-quality dataset of Pokémon images as there are a number of variations between the look of Pokémon across generations, fan art styles different from the original, numerous shots that contain multiple Pokémon and even Pokécards that contain large amounts of text in addition to the images of the Pokémon. This section outlines the step-by-step construction of the building of the new dataset.

4.1 Scraping New Images

We scraped images on Google Image Search using a python library appropriately titled google_images_download [3]. We aimed to have around 100 images for each Pokémon in the new dataset, so we scraped proportional to the number of images we had in the original dataset, with a margin of 50 images so that we could discard any low-quality, incorrect or inappropriate images. Thus we downloaded about 8000 images across Pokémon classes to our local machines, which took about 8 hours. We made sure that the directory structure was the same as the original dataset, with one sub-directory for each Pokémon, so that merging them in the future would not be a problem.

4.2 Manual Inspection

Each of the 8000 images required manual inspection due to the quality issues described above. The criteria used to determine whether we would keep an image is as follows:

- Image contains one instance of the correct Pokémon
- The style of the Pokémon contained is reasonably close to the original - we could not afford exact copies as they simply were not enough images, so we allowed similar looking fan-art and pictures of plush toys and action figures.
- Other elements of the image were minimal - there were no human characters, no large amounts of text etc.

The manual inspection took a fair amount of time - upwards of 12 hours due to the large amounts of images and the involved evaluation of each image. We found that even with popular Pokémon such as Pikachu, as we got to the middle of the scraped images, we would begin to get images of the wrong Pokémon or images that were unsuitable for the criteria

listed above. This resulted in us discarding upwards of 2000 images that we had scraped, resulting in the remainder of 6000 images across the different classes.

4.3 Combining Old and New

As mentioned above, the identical structures of the original and new images made them easy to combine simply by dragging and dropping. We copied over each of the sets of images to create a new balanced dataset with over 16000 images, with the range of images for each Pokémon varying between approximately 90 and 150 images per Pokémon. The number of images per class is visualized below, on the same scale as the visualization of the original dataset seen earlier in this report

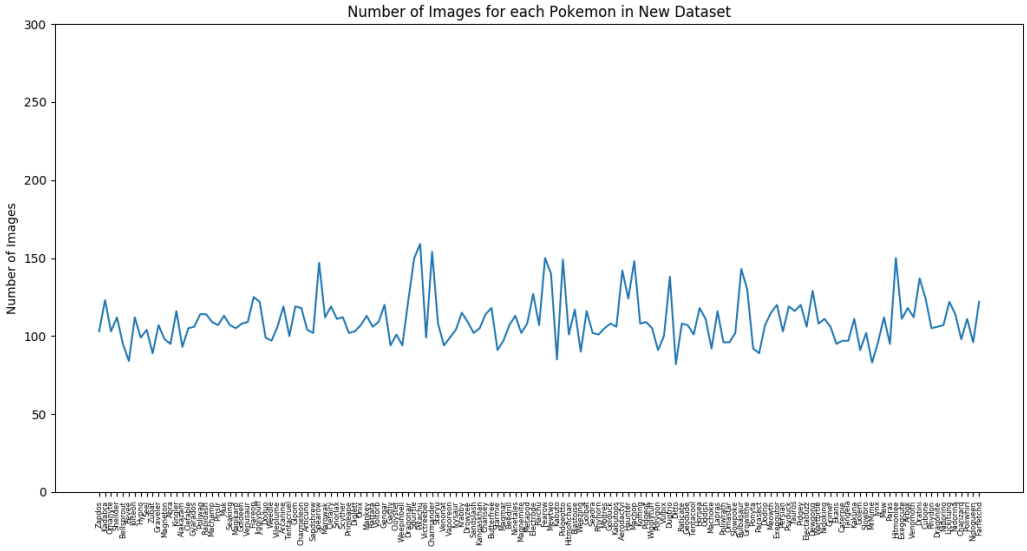

Figure 11: Number of Images across Pokémon classes (new balanced dataset)

4.4 Data Cleaning and Image Verification

We followed the following steps to make our data ready to use:

- Standardized all file names within Pokémon subdirectories
- Converted all images to jpg format
- Using the Python Image Library, ensure the readability of each file
- Divided all Pokémon classes into training and validation directories with an 80/20 split as justified earlier.

5 Building a Classifier (Take 2)

5.1 Training the Convolutional Neural Network

Now that we had a significantly more balanced dataset, we trained the same architecture of classifier to examine the results. We used the same workflow with Google Drive and Colab, and identical loss metrics and optimizers. The results of training and testing the resulting classifier are below:

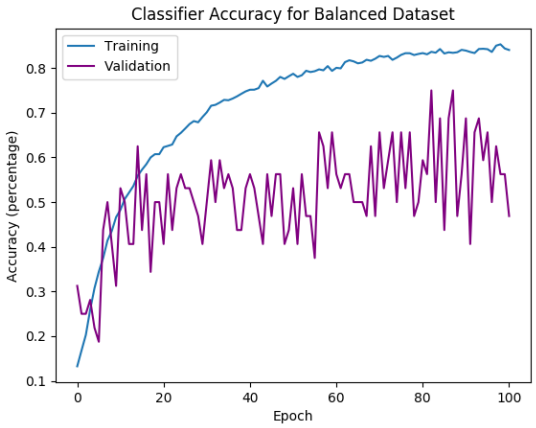

Figure 12: Accuracy across Epochs

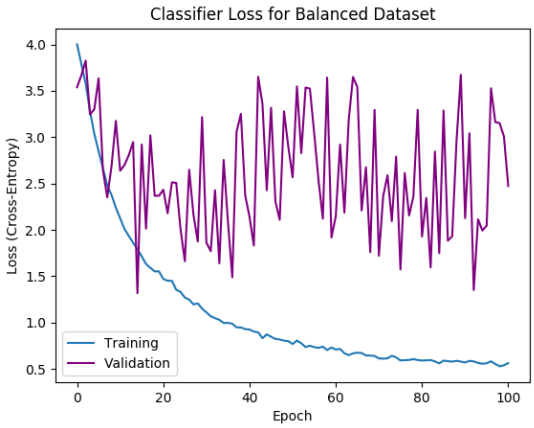

Figure 13: Loss across Epochs

As demonstrated in the figure, the classifier trained on the new balanced dataset obtains a maximal training accuracy of 86% and a maximal validation accuracy of 71%. This means about a 10% increase in average accuracy as compared to the previous dataset and classifier. However, we see the same trend as before where the validation accuracy and loss seem to fluctuate wildly across epochs. We hoped this variability would reduce with a high-quality balanced dataset, but it seems that the paradigm of training images is still too variable for a smooth validation curve.

5.2 Evaluating the Results

In order to delve deeper into the performance of the newly trained classifier, we performed an analogous class wise analysis, visualized below.

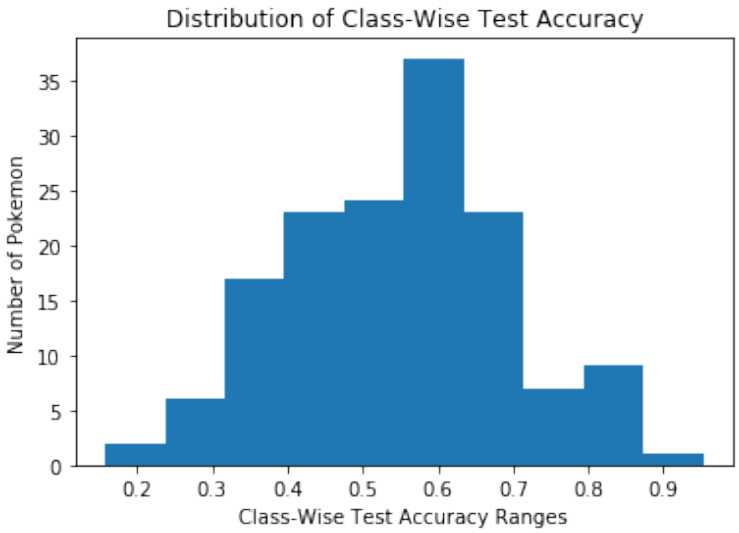

Figure 14: Distribution of Test Accuracy across Pokémon classes

The distribution of class-specific accuracies has now shifted to concentrate between 40% and 70%, which is an improvement from the majority of classes being classified at an under 50% accuracy earlier. It seems that we have now trained a better classifier as we have not only increased the average validation accuracy, but made the distribution of class-wise accuracy ranges much more reasonable. As a sanity check, we also plot the relationship between the number of training examples and test accuracy for each class, to make sure we have solved the imbalance issue we experienced earlier. This is visualized below.

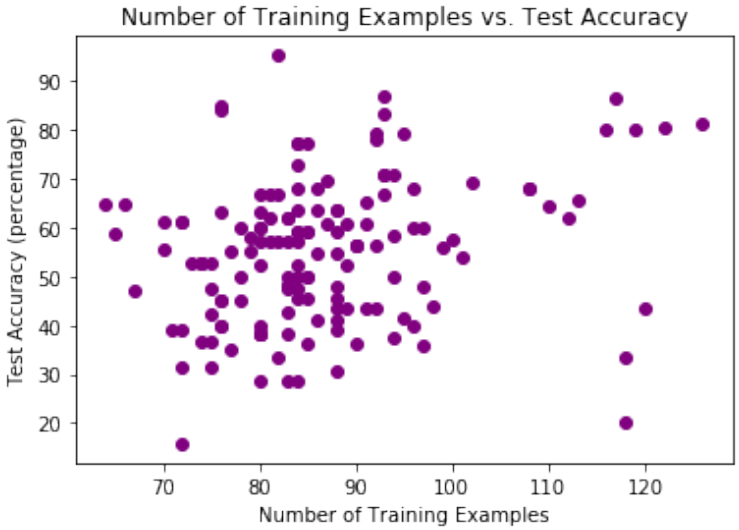

Figure 15: Visualizing relationship between Number of Training Images and Test Accuracy across Pokémon Classes

We see that we have eliminated the two clusters that were prominent in the previous plot, and the distribution of well-classified Pokémon is now much more even. This also

means that we are no longer averaging well-classified frequent classes with badly-classified infrequent classes, so the average validation accuracy reported is more representative of the true classification power of the system.

Now that we have performed some basic evaluation, we can analyze the performance at a more high-level. We are still unable to obtain optimal validation performance, as 71% accuracy is reasonable but not stellar. One reason for this lower performance could be that Pokémon have the concept of evolution, where one Pokémon levels up to a similar looking but differently named Pokémon. Until now, we have not taken into account that some Pokémon classes may look remarkably similar to each other becuase they are part of the same evolution chain. In order to test this hypothesis, we run an experiment an experiment on a subset of the data that only contain Pokémon from distinct Pokémon evolutionary chain. This experiment and the corresponding results are described in the next section.

6 Experimenting with a Subset of Data

As described above, the rationale for training a smaller classifier on a subset of the data is to test whether similar looking Pokémon from the same evolutionary chains are inhibiting the overall performance of the classifier. We select 20 classes from the balanced dataset, making sure to satisfy the constraint of evolutionary chains, and use the training and validation for these Pokémon to make a smaller dataset. This results in a training set of about 1850 images and a test set of 450 images. We train a classifier with the same architecture as the previous two cases, to make a fair comparison, using the same metrics and optimizer. However, we train the network only for 50 epochs since there are a significantly lower number of Pokémon classes, and subsequently, lower amount of complexity that needs to be captured. The accuracy and loss across epochs is visualized below.

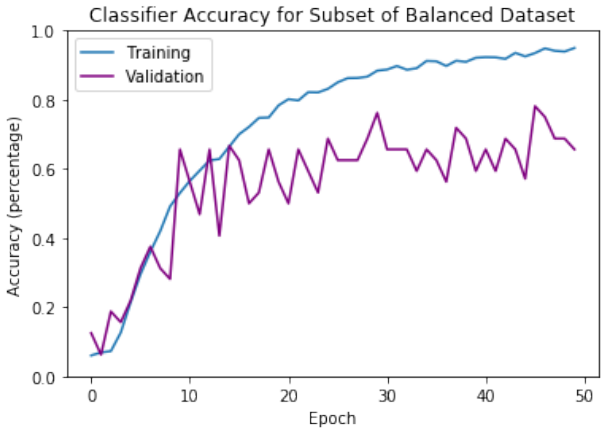

Figure 16: Accuracy across Epochs

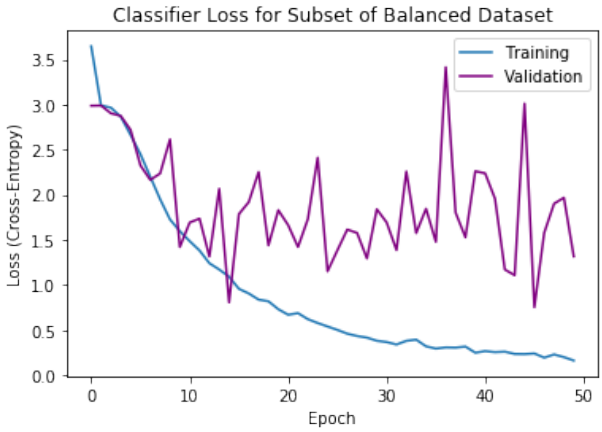

Figure 17: Loss across Epochs

We find that the maximal training accuracy obtained is about 94% whereas the maximal validation accuracy achieved is 78%. This is a 7% increase in validation accuracy as compared to the full-dataset. The distribution of class-wise test accuracies is illustrated below.

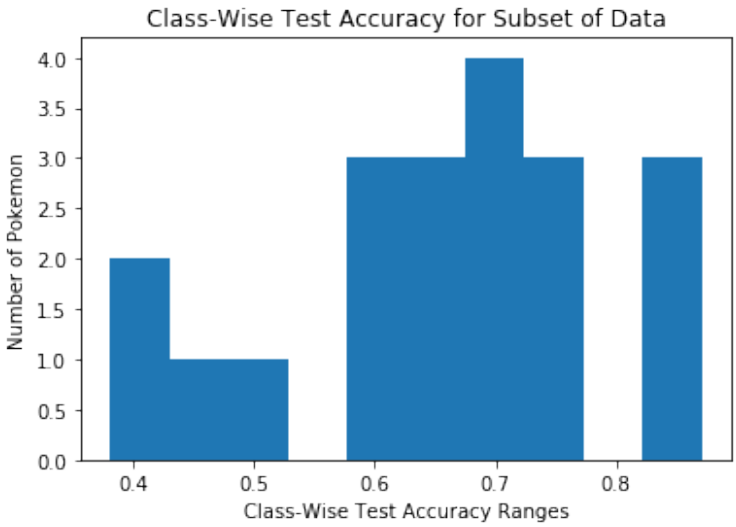

Figure 18: Distribution of Test Accuracy across Pokémon classes

The figure above illustrate that a large proportion of the Pokémon are classified with greater than 55% test accuracy. The classification is obviously still not perfect, but there is definitely an improvement. These results seem to provide support for our hypothesis that similar looking Pokémon from the evolutionary chain might be inhibiting the performance of the classifier. An idea for a future experiment is grouping together all the Pokémon in an evolutionary chain into the same class and train a new classifier with sets of Pokémon instead of individual ones.

7 Evaluating the Project

We set out to build a fully functional Pokédex and we were able to build a reasonable, though not optimal classifier. We think that our general approach of collecting relatively high-quality data and training a Convolutional Neural Network makes sense, and given more resources could further fine-tune performance. We feel proud to have collected the most extensive Pokémon dataset currently available, and are in the process of publishing it on Kaggle. We feel that collecting and working with an original dataset taught us just how integral the quality of the data is to the success of the project. We had mostly worked with pre-collected data in the past, so learning to clean, organize, manage and efficiently access our own data was a great learning experience.

In terms of improvement, we think that exploring intelligent groupings of Pokémon might be a way to make a better classifier in the future, because despite our best efforts, the size of the dataset is still quite small and pooling resources may make it easier to train an efficient classifier. Another idea would be to group large numbers of Pokémon by their type (eg. Fire, Water, etc) as there are much fewer types than individual Pokémon. Since type and color is often closely correlated, this might also yield a satisfactory classifier using the same data.

Another experiment we think could be valuable is training a classifier with a fully-convolutional approach, so that it is agnostic to the size of the input images. We think this might be a valuable investment of energy as the images are of different aspect ratios and we are currently resizing all of them to a standard format, which might be distorting the Pokémon and lessening performance. Overall, we really enjoyed working with a subject matter close to our hearts, and enjoyed building a functional computer vision system end-to-end.

8 List of Resources Used

- Keras - https://keras.io/
- OS - https://docs.python.org/3/library/os.html
- Pillow - https://pillow.readthedocs.io/en/stable/
- Numpy - https://numpy.org/
- Google Image Download - https://pypi.org/project/google_images_download/
- Original Dataset - https://www.kaggle.com/thedagger/pokemon-generation-one/data

References

[1] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[2] Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. 2015. URL: http://arxiv.org/abs/1412.6806.

[3] *google_images_download*. May 2019. URL: https://pypi.org/project/google_images_download/.

[4] *Parents' Guide to Pokémon*. URL: https://www.pokemon.com/us/parents-guide/.