A Smart Pokédex: Harnessing the Power of Machine Learning

Team: Christian Cipolletta, Tyler Walsh

1) Introduction

Pokémon is a world famous phenomenon. It has video games, tv shows, movies, plushies, trading cards, posters, figurines, and so much more. There are over 1,025 unique pokémon which makes it difficult to know them all. Some have similar names, similar looks, or similar abilities which makes telling them apart harder. We believe that machine learning can be applied here to help speed up the search process.

The dataset we are using has 1,215 pokémon where 1,025 are unique and the other 190 are the different forms of already included pokémon [1]. It is split into two parts, a folder filled with 112x120 color images of each pokémon and a comma-separated value file (csv) with a number of statistics about each pokémon. According to the information page for the dataset, the .csv contains:

- ***images:*** *Path to an image of the Pokemon, in the 'images' folder*
- ***index:*** *Each Pokemon number in the PokeDex.*
- ***Name:*** *Name of the Pokemon.*
- ***Type 1:*** *First type of the Pokémon.*
- ***Type 2:*** *Second type of the Pokémon. Not all have two types. This will be null in that case.*
- ***Total:*** *Sum of all stats.*
- ***HP:*** *Hit points, or health. How much damage a pokemon can withstand.*
- ***Attack:*** *The base for normal attacks.*
- ***Defense:*** *The base resistance against normal attacks.*
- ***SP. Atk.:*** *The base for special attacks.*
- ***SP. Def.:*** *The base resistance against special attacks.*
- ***Speed:*** *Determines which pokemon attacks first each round.*

Our final deliverable will be 2 machine learning models. The first model will take in an image of a pokémon and output the first type of it. The second model will take in the value of each of the six stats (HP, Attack, Defense, SP. Atk., SP. Def., and Speed) and output the first type of it. We may also make the input the six stats and the total or possibly just the total, depending on which gives the best results.

For the image classifier, we plan to implement two Convolutional Neural Network (CNN) models and compare their performance to a K-Nearest Neighbor (KNN) model. The first CNN will have you-only-look-once (YOLO) architecture through an Ultralytics package based on PyTorch and the second will be a basic LeNet-5 CNN created using PyTorch [2, 3]. We will implement our own KNN using python packages like PyTorch and Scikit. For the statistics classifier, we plan to implement a logistic regression model and another KNN. If we have time, we would like to explore other model types like transformers and reinforcement learning, but do not want to be too ambitious at this time.

We will compare the image CNN and KNN by recording their accuracy on a held-out test set. We will also run a determined hyperparameter sweep to judge the effect they have on

accuracy. For the CNN we will change the learning rate, weight decay, and number of epochs. For the KNN, we will be changing the value of K. This will allow us to not only understand which architecture is more effective, but the effect of hyperparameters as well. We plan to record the time taken for training and testing in addition, so that we can take that into account. For the logistic regression model and statistical KNN, we will again record the accuracy on a held-out test set. We will test an unregularized logistic regression, one will L1 regularization, and one with L2 regularization. The KNN again will have a changing value of K.

2) How We Have Addressed Feedback From the Proposal Evaluations

The feedback we received stated that our project proposal was that it was a “very nice report with clear technical details.” In response to this, we started working on our project immediately to make sure we had time to ask questions and fix any issues when they inevitably arise.

3) Prior Work We are Closely Building From

A. How to Use Machine Learning to Build a Pokémon Search Engine — pt. 1. https://itnext.io/how-to-use-machine-learning-to-build-a-Pokémon-search-engine-pt-1-2db1f8551426. This source is very similar to our project as it uses a dataset of Pokemon to make a search engine, using the data of each pokemon. This project uses KNN to find pokemon with stats similar to what the user inputted.
B. Pokemon Type Predictor. https://www.kaggle.com/code/mrkupo/pokemon-type-predictor. This code is an example of what we are trying to implement. It is an image classifier that guesses the first type of a pokémon using a random forest model as well as a basic CNN.

4) Contributions

4.1 Implementation Contributions

For the image classifier, we plan to implement two Convolutional Neural Network (CNN) models and compare their performance to a K-Nearest Neighbor (KNN) model. The first CNN will have YOLO architecture through an Ultralytics package based on PyTorch [2]. This package enables the easy use of different versions of YOLO like v5, v8, v9, v10, and v11 as well. YOLO is being chosen because of its status as an state-of-the-art model (SOTA) with incredible accuracy and speed. Each version of YOLO has a slightly different architecture, but the main difference between YOLO and other models is that it only looks at an image once. The second CNN will have LeNet-5 architecture and be created using PyTorch [3]. LeNet-5 is made up of 7 layers, convolutional, subsampling, and fully-connected. This is being chosen because of its basic nature and proven ability. We will implement our own KNN using python packages like PyTorch and Scikit. We not only will try these three different approaches and the number of different models, but will also vary hyperparameters. For the YOLO models, we will vary

learning rate, weight decay, number of epochs, and image size. For the LeNet-5 model, we will vary the same hyperparameters. The only hyperparameter for KNN models is the value of k.

For the statistics classifier, we plan to implement multiple logistic regression models and compare them to each other and to a KNN model. The first logistic regression model will be unregularized, while the other two will be L1 regularized and L2 regularized respectively. We will implement our own KNN using python packages like PyTorch and Scikit. The hyperparameter we will be varying for logistic regression is τ (tau) which is the prediction threshold. KNN has the same hyperparameter as before, k.

4.2 Evaluation Contribution

Our experiments aim to address several key questions regarding the performance and robustness of our models. For the image classifier, we aim to determine how YOLO-based CNN compares to a custom LeNet-5 CNN and a KNN model in predicting a Pokémon's primary type from its image. We will evaluate how hyperparameters such as learning rate, weight decay, number of epochs, image size, and the value of k impact both accuracy and runtime. For the statistics classifier, we will compare logistic regression models (unregularized, L1-regularized, and L2-regularized) to a KNN model to assess their accuracy in predicting a Pokémon's primary type based on its stats. We will also explore how τ affects logistic regression performance and k affects KNN performance.

To evaluate our models, we will use a simple KNN implementation as the baseline for both the image and statistics classifiers. Performance metrics will include accuracy, precision, recall, F1-score, and inference time, with an emphasis on understanding the trade-offs between model accuracy and runtime efficiency. For shifting the image dataset, we plan to convert images to grayscale and rotate the images. To shift the statistics dataset, we plan to add in more variables such as the “total” or even the second type of the pokémon. By conducting these experiments, we aim to validate the hypothesis that CNN models will outperform simpler approaches for image classification, while logistic regression will be more effective for stats-based classification. This evaluation will also provide insights into the importance of hyperparameter tuning and dataset quality in achieving optimal model performance.

For both the images and the statistics, we will have 80% of the instances reserved for training, 10% for validation, and 10% for testing. The dataset can be read and the data can be preprocessed. We have already set up the training and testing of the KNN image classifier. We have set up the training of the YOLO model, but not the testing just yet. The hyperparameter sweeps, data shifts, LeNet-5 image classifier, and statistics classifiers have not been implemented yet. The KNN image classifier with a k of 3 has a test accuracy of 10.85%.

5) Risk Mitigation Plan

We plan to start with image recognition due to it being the easiest since we have prior experience working with image recognition models. The dataset we are using already contains images of all the pokemon so inputting an image and having our model guess is a simple task. We also will get one example working for each of our planned models before starting any hyperparameter sweeping. This way we can still compare models against each other even if we don’t have the time to improve them with hyperparameter tuning. One issue is that 1,215

images is relatively small for deep learning models like CNN so if it does not work, we plan to combine other pokémon image datasets with our current one to improve the training and testing accuracy [4]. If we do not have enough compute for our ambitions, we plan to use models with fewer parameters, training the models for less epochs, and reducing the size of the datasets. For example we can just use the first couple generations instead of all of them. As a last resort, Ultralytics comes with a number of example datasets (COCO, etc.) that we can pivot over to if needed [2, 5]. Overall, we have many different risks identified and planned for. There are possibly more out there that we have not thought of, but our work should be adaptable enough to still produce results.

6) Other Prior Work / References

1. Christoffer MS, “Pokemon with stats and image”, https://www.kaggle.com/datasets/christofferms/pokemon-with-stats-and-image.
2. Glenne Jocher, Jinq Qiu, “Ultralytics”, https://docs.ultralytics.com/
3. Yann Lecun, Léon Bottou, Yoshua Bengio, Patrick Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE 1998.
4. Divyanshu Singh369, “Dataset of 32000 Pokemon Images & CSV, JSON”, https://www.kaggle.com/datasets/divyanshusingh369/complete-pokemon-library-32k-images-and-csv
5. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár, “Microsoft COCO: Common Objects in Context”, https://cocodataset.org/#home.

7) Original Project Proposal

Title: A Smart Pokedex: Harnessing the power of Machine Learning

Team: Tyler Walsh, Christian Cipolletta

Kaggle dataset:
Pokémon with stats and image
(https://www.kaggle.com/datasets/christofferms/pokemon-with-stats-and-image/data)

Part 1: Implementation contributions:
We plan to implement an array of different models in this project. We plan to implement a Convolutional Neural Network (CNN) which will guess the main type of the pokemon in an image and compare its performance to a K-Nearest Neighbor (KNN) model. For the CNN we plan to use YOLO architecture through an Ultralytics package based on PyTorch (https://github.com/ultralytics/ultralytics). For the KNN we will implement our own using python packages like PyTorch and Scikit. The other experiment we are going to run is testing different models on their ability to guess the "Total Score" of a pokemon based on 6 other parameters. For this we plan to compare the accuracy of linear regression models with and without regularization. Again we will most likely create our own using the homeworks from this class and python packages.

If we have time, we would like to explore other model types like transformers and reinforcement learning, but do not want to be too ambitious at this time.

Part 2: Evaluation contributions:
We will compare the CNN and the KNN by recording the Test Accuracy on a held-out test set. We plan to use k-fold to ensure there is not a lucky run by one of our models. We will also run a determined hyperparameter sweep to judge the effect they have on accuracy. For the CNN we will change the learning rate, weight decay, and number of epochs. For the KNN, we will be changing the value of K. This will allow us to not only understand which architecture is more effective, but the effect of hyperparameters as well. We plan to record the time taken for training and testing in addition, so that we can take that into account.

For the linear regression model we will also record the Test Accuracy on a held-out test set and use k-fold verification. We will determine the accuracy by using the Mean Squared Error (MSE) of the model. The hyperparameter we will be tuning for the linear regression models is lambda.

Prior work:
1. https://itnext.io/how-to-use-machine-learning-to-build-a-Pokémon-search-engine-pt-1-2db1f8551426
2. https://docs.ultralytics.com/
3. https://www.kaggle.com/code/devraai/analyzing-pokmon-stats-and-predictive-modeling

Which parts of the curriculum from this class do you expect to apply?:
We plan to apply KNN, CNN, and linear regression lessons we learned from the curriculum. We have a dataset with both image and quantitative data which will allow us to implement a plethora of different models. We expect that reference 1 will give us a good starting point for the KNN, reference 2 will help us set up the CNN, and reference 3 will help us set up the linear regression.

Expected challenges and risk mitigation:
One challenge will have to deal with is training and inference time. In this day and age, everyone expects lightning fast speeds. The KNN and linear regression models are expected to train and predict fast, but the CNN might be a bigger burden. We will make sure to record the time taken to train and predict to compare this for the future. Furthermore, we plan to test the use of fine tuning a pretrained model which, in theory, should take a shorter time to train and be more accurate.

Another one of our expected challengers is data manipulation. The dataset is impressive as it gives us the images, names, types, and statistics of the pokemon. However, if we wanted to do any more analysis, for example some Pokémon are legendary status, we would have to manipulate the data ourselves. However, this does give an interesting idea, maybe a reinforcement learning model could be used to guess which Pokémon are legendaries.

Ethical considerations and broader social impact:
The major ethical concern behind our project is that Pokémon is technically not in the public domain. While we are not selling anything nor profiting in any way, the publishers of Pokémon (Nintendo) are very hostile against people using their work. Therefore if the project becomes big, it could easily be taken down by the publishers, Nintendo or Game Freak. Another impact is that machine learning models oftentimes use much more processing power (and therefore more electricity) than conventional methods. There is a chance that there will be more electricity consumption globally because of implementing these methods.