Pokemon Classification Analysis

Joshua Affleje and Jack Oebker

Arizona State University, Tempe, AZ 85281, USA

May 6, 2025

Abstract

This project explores the task of predicting Pokémon typings (e.g., Water, Fire, Grass) using computer vision and machine learning techniques. Our team built two distinct models: a binary MLP classifier for distinguishing Pokémon from real-world animals, and a multi-label CNN classifier for predicting Pokémon types. For the binary task, we achieved a peak accuracy of 78% at 9 training iterations using an MLP. For the typing task, we trained a ResNet18-based CNN that incorporates both image features and average RGB color; this model reached a accuracy of 58% after 10 epochs. We also explored K-Means clustering to interpret color-type relationships. The results highlight both the strengths of image-based models and the limitations of relying solely on color. Our work suggests that combining neural networks with interpretable visual cues can lead to promising strategies for future Pokémon classification models. In addition to exploring these models, we were also curious whether visual clues in Pokémon designs might reflect their real-world inspirations. By distinguishing Pokémon from animals and identifying their elemental types, we aimed to explore patterns that could potentially hint at how future Pokémon might be designed. As such, we trained two complementary models: a binary MLP classifier to separate Pokémon from animals, and a multi-label CNN to identify types.

Introduction

In the world of Pokémon, each species is defined in part by one or two elemental types (e.g., Fire, Water, Grass). These types impact in-game behavior and are often reflected in a Pokémon's appearance. This project seeks to leverage those visual cues to build a machine learning model capable of predicting a Pokémon's typing based on its image. We explore two classification problems: (1) a multi-label classifier for assigning elemental types to Pokémon using deep learning, and (2) a binary classifier distinguishing Pokémon from animal images using simpler MLP models. Our goal is to understand how well image-based models can learn typing information and how interpretable visual features like color contribute to prediction performance.

Methods

Data Collection and Preprocessing

The Pokémon images and typing metadata were sourced from public datasets on Kaggle and supplemented by image scraping. All images were resized to 224x224 and normalized. The typing labels were mapped to integer indices and encoded as multi-hot vectors to support multi-label classification.

Binary Classifier Dataset Creation

To create our Pokémon-vs-animal dataset, we downloaded clean PNG sprites for Pokémon and placed them over randomly chosen natural landscape backgrounds. This composite pipeline involved scaling, flipping, and overlaying sprites to ensure the model focused on the Pokémon and not white background pixels. All images (both Pokémon and animals) were resized to 224×224. We split the dataset into training (655 samples) and test sets (164 samples) per class, and stored them in separate folders. The MLP classifier was then trained on flattened RGB image arrays across multiple iteration settings to explore optimal training length.

Multi-typing Classifier Dataset Merging

To prepare the dataset for training the CNN classifier, we merged typing metadata with image data using the Pokédex number as a key. Typing information, including primary and secondary types, was sourced from the *Complete Pokédex v1.0.0* dataset by Betetta (2021), which provides structured metadata for all officially released Pokémon. This was joined with a local folder of Pokémon sprite images using the pokedex_number and pokemon_name fields, aligning each image with its corresponding type1 and type2 labels. To reduce ambiguity and improve label consistency, we filtered the dataset to retain only base-form Pokémon and removed entries with missing or undefined primary typings. This yielded a clean set of 692 labeled Pokémon.

After preprocessing, we split the dataset into training and validation sets using an 80/20 stratified split to preserve type distribution across both subsets. For model training, we implemented a custom PyTorch dataset class, PokemonTypeDataset, which loads and preprocesses each image to a 224×224 resolution, normalizes pixel values, and converts the image into a 3-channel tensor. The dataset also constructs a multi-hot target label encoding both primary and secondary types. Additionally, each sample returns the average RGB color of the image as a separate feature tensor to provide supplementary global color information. This structure enabled our model to learn from both detailed image features and overall color cues in a streamlined pipeline. Full implementation details are available in the accompanying notebook.

CNN Model

ResNet-based Model: For multi-label Pokémon type classification, we selected ResNet18 as the basis for our convolutional neural network. ResNet18 is a well-established architecture known for its balance between depth, performance, and computational efficiency, making it a strong candidate for tasks with moderate dataset sizes like ours. Unlike larger architectures, ResNet18 avoids overfitting while still capturing hierarchical spatial patterns such as shape and texture—critical features for distinguishing Pokémon based on appearance.

To tailor the model for our task, we replaced the final fully connected (FC) layer with a custom classifier head. This head took as input both the deep visual features extracted by ResNet18 (512-dimensional vector) and a 3-dimensional average RGB color vector computed from each image. By concatenating these two components, the model was encouraged to use both global color cues and fine-grained visual features during classification. This design also made it easier to interpret the influence of color on prediction outcomes.

Since many Pokémon have two elemental types, our model was trained as a multi-label classifier using the binary cross-entropy loss function with logits (BCEWithLogitsLoss). This setup allowed the model to predict one or more active labels from the full set of types, rather than being forced into a single-class output. Our final implementation achieved acceptable accuracy for 10 epochs and reflected meaningful structure in type predictions across different color and shape groupings.

Binary Classifier: For the Pokémon vs. non-Pokémon task, we used scikit-learn's MLPClassifier with image pixels flattened into input vectors. The model was tuned over different max_iter values and evaluated using test accuracy and visual confusion categories.

Results and Discussion

Binary Classification

The Pokémon vs. non-Pokémon classifier was implemented using a 3-layer MLP that takes in flattened 224×224 RGB images. We trained the model with varying iteration counts (1–10) to determine the optimal stopping point. As shown in Figure 1, the model quickly climbed to 75% accuracy by iteration 5, peaking at 78% by iteration 9. Beyond that, accuracy plateaued or declined slightly, suggesting diminishing returns with overfitting. The loss curve similarly dropped rapidly and leveled out around 0.4.

To analyze prediction behavior, we created a confusion matrix (Figure 3) and visualized examples from all four categories: true positives, true negatives, false positives, and false negatives (Figures 4–7). The model handled most Pokémon cases well, but confusion arose with animals exhibiting unusual textures or fantasy-like environments. Overall, the results suggest that an MLP can disentangle shape and color cues surprisingly well in this binary context.

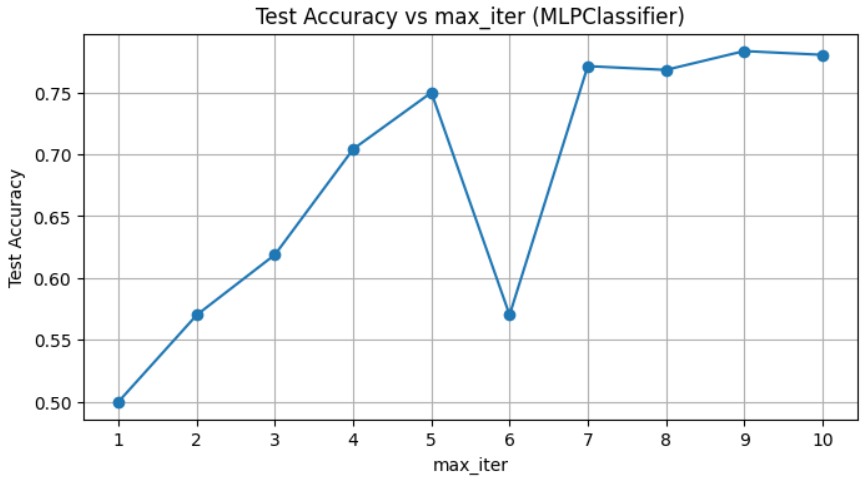


Figure 1: Test accuracy of the MLPClassifier across different training iterations (max_iter).

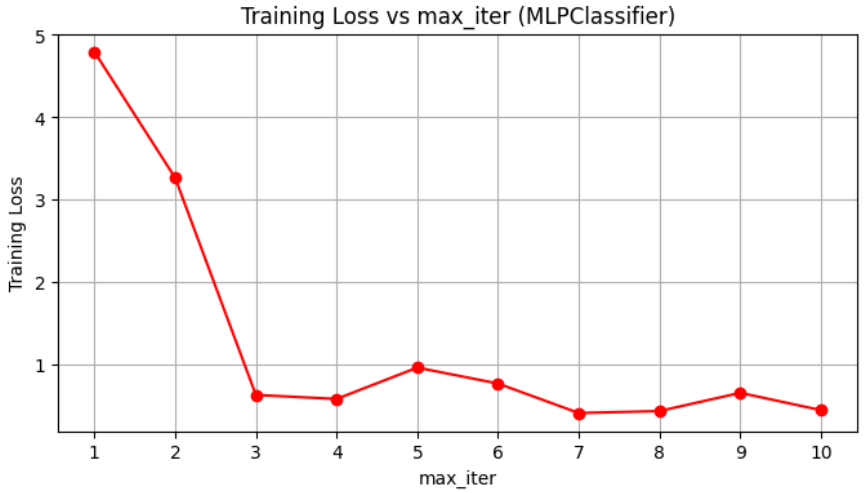


Figure 2: Test Loss of the MLPClassifier across different training iterations (max_iter).

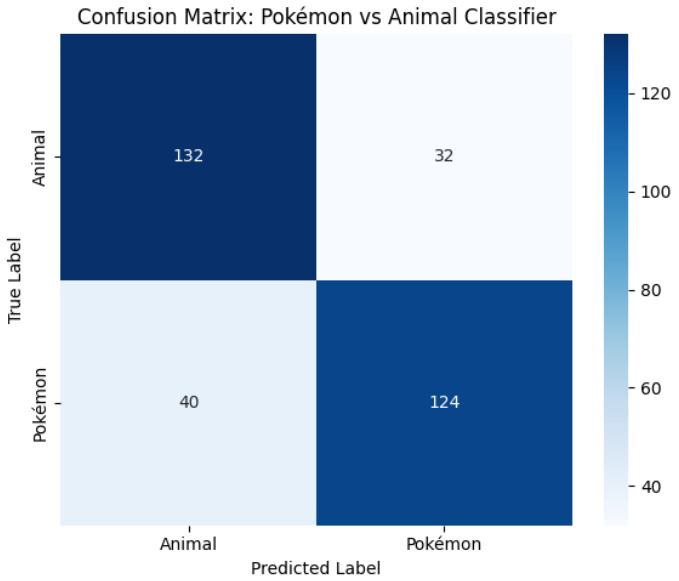


Figure 3: Confusion matrix for Pokémon vs. Animal classifier using the best MLP configuration.

Bindary Classification Predictions

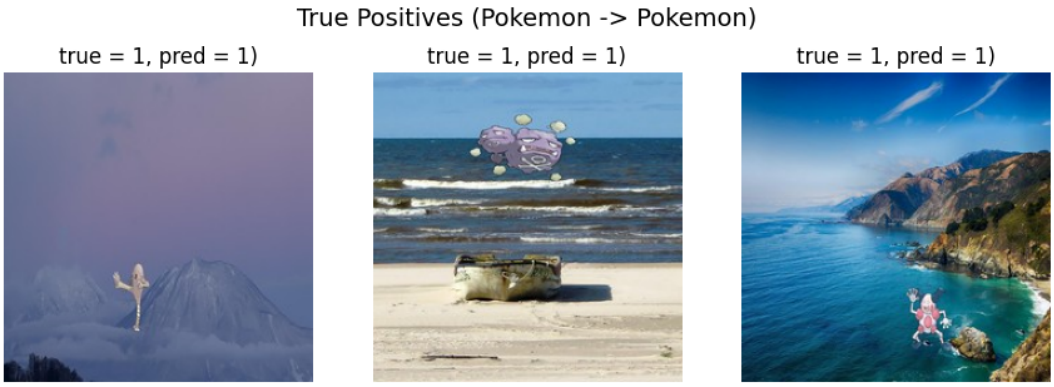


Figure 4: True Positives: Pokémon correctly identified as Pokémon.

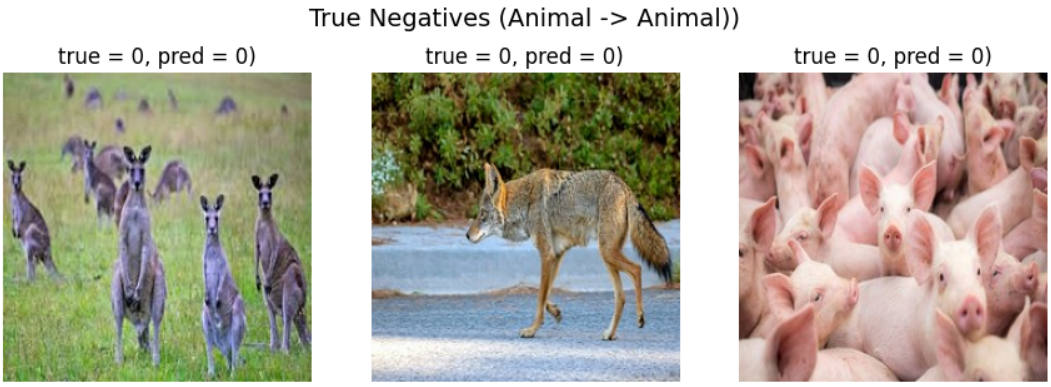


Figure 5: True Negatives: Animals correctly identified as animals.

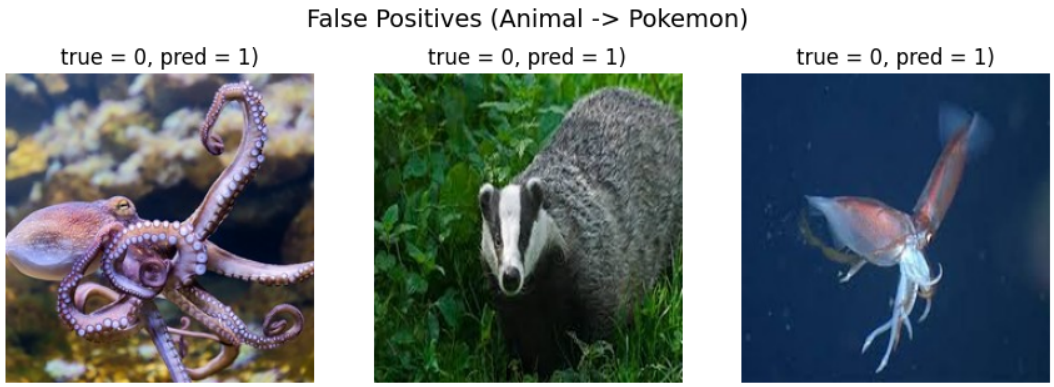


Figure 6: False Positives: Animals incorrectly classified as Pokémon.

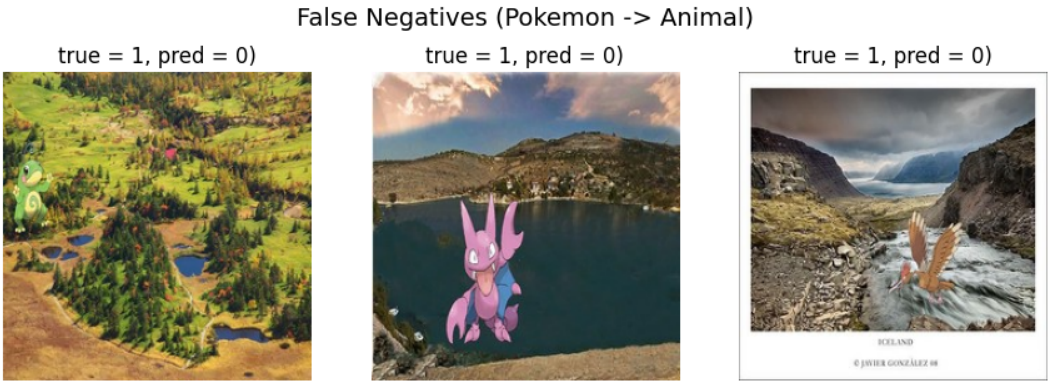


Figure 7: False Negatives: Pokémon incorrectly classified as animals.

Multi-label Typing Model

The ResNet-based CNN model aimed to classify Pokémon by their elemental types using both image and average color input. After 10 training epochs, the model reached an accuracy of 58% (Figure 8). Loss

dropped from 0.6 to approximately 0.1 before flattening, suggesting diminishing improvement beyond epoch 8. Figure 9 shows per-type accuracy, with Electric, Bug, and Water achieving the best performance. Sample predictions (Figure 10) reveal cases where the model predicted one of the correct types while missing the second. The results demonstrate that while image-based cues are useful, distinguishing fine-grained type details remains a challenge.

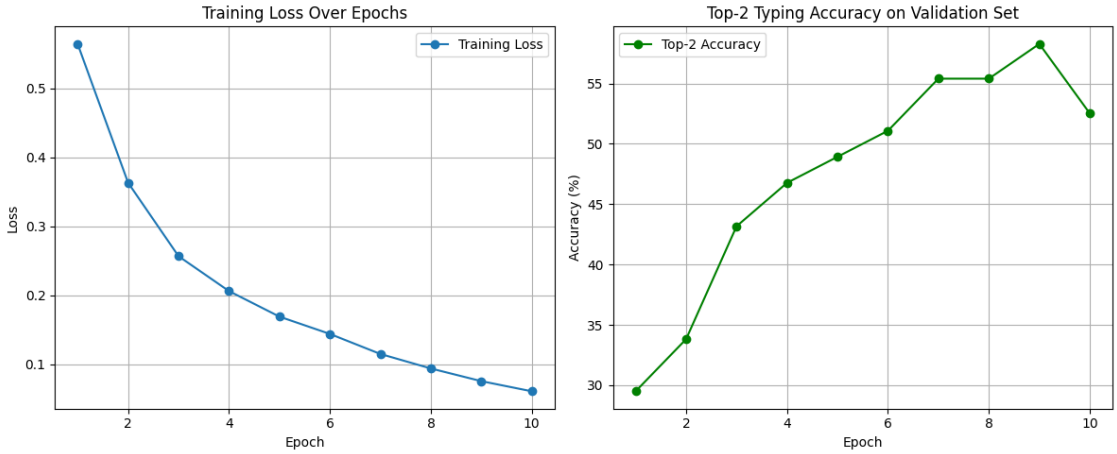


Figure 8: Training loss and validation accuracy over 10 epochs.

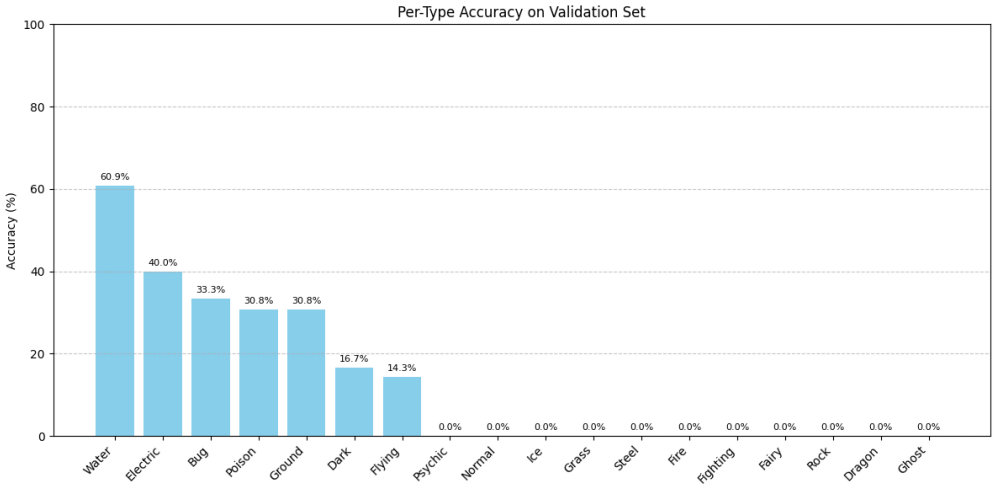


Figure 9: Accuracy per type on the validation set.

Figure 10: Example model prediction. Correct: Poison; Missed: Bug.

Exploratory Analysis

K-Means clustering revealed strong correlations between certain clusters and type labels. For instance, blue-dominant clusters aligned with Water types, and red/orange clusters with Fire types. However, many types (e.g., Poison, Psychic) overlapped in color space. A heatmap of cluster vs. type frequencies supported this observation.

Color Features and Clustering

We computed the average RGB color for each image to explore color-type correlations. We then applied K-Means clustering (k = number of Pokémon types) on these average color vectors. Each Pokémon was assigned a cluster ID, and clusters were analyzed based on the distribution of true types they contained. The results of the clustering revealed some interesting patterns in the distribution of typings. Water and Normal typings appeared the most as a primary typing while flying appearing significantly more as a secondary typing.

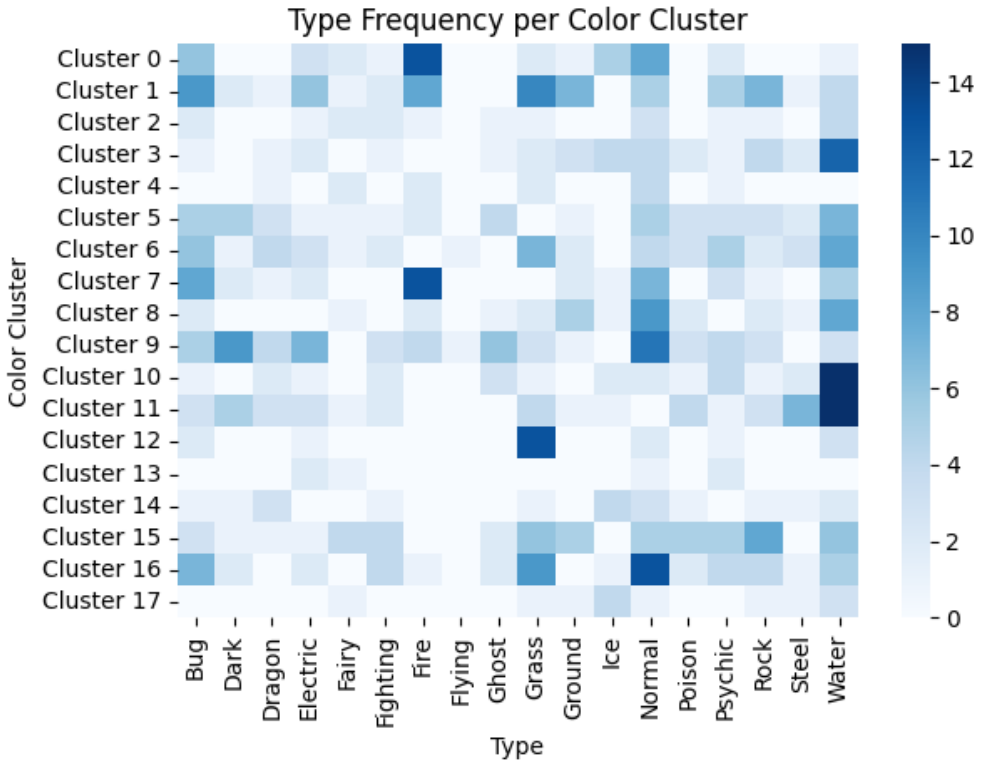
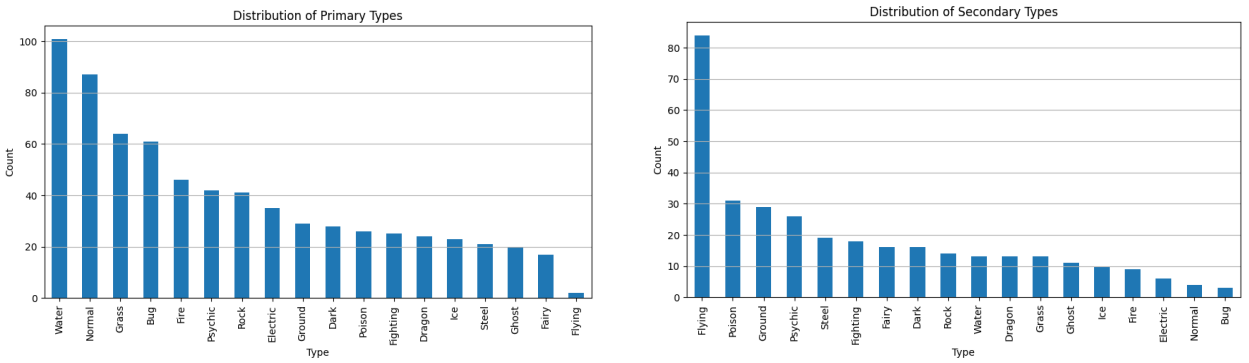


Figure 11: Heatmap of primary type frequency across color clusters.



(a) Distribution of primary types in the dataset.

(b) Distribution of secondary types in the dataset.

Figure 12: Type distribution comparisons between primary and secondary typings.

Conclusions

This project demonstrates that deep learning can be effectively used to classify both Pokémon identities and their elemental types using only visual inputs. Our binary MLP classifier achieved up to 78% accuracy in distinguishing Pokémon from real animals by using flattened image pixels, showing that color and shape alone can be quite informative. However, tuning the number of iterations was crucial to avoid underfitting or overfitting.

The CNN-based type classifier, built on ResNet18 and enhanced with average color input, reached an accuracy of 58%. While not perfect, it revealed meaningful relationships between visual features and type labels. However, our dataset was limited to 692 usable base-form Pokémon, excluding alternate forms, Mega Evolutions, and regional variants. This restriction reduced label diversity and may have contributed to class imbalance. Our K-Means clustering and typing distribution analyses further revealed that certain types—such as Water, Fire, and Normal—are disproportionately likely to appear as primary typings, while others like Fairy, Ice, and Ghost more frequently occur as secondary. This distributional bias could unintentionally skew model performance or lead to underrepresentation of visually distinct but infrequent types. Future work should consider techniques such as type balancing, re-weighted losses, or expanded datasets that incorporate alternate forms to more accurately capture the diversity and complexity of Pokémon design.

For future work, we propose several enhancements. One approach is to pre-label representative color scales for each type and use cosine similarity to measure alignment between a Pokémon's average color and these type prototypes. Additionally, Generative Adversarial Networks (GANs) could be employed to generate synthetic Pokémon-like samples, which would help balance the dataset and expand the training space. While our analysis focused only on base-form Pokémon, we aim to incorporate alternate forms, Mega Evolutions, and regional variants to increase classification difficulty and better reflect real-world diversity. Finally, combining structured data with deeper image models or developing ensemble methods that blend classical machine learning and neural networks could further improve classification performance.

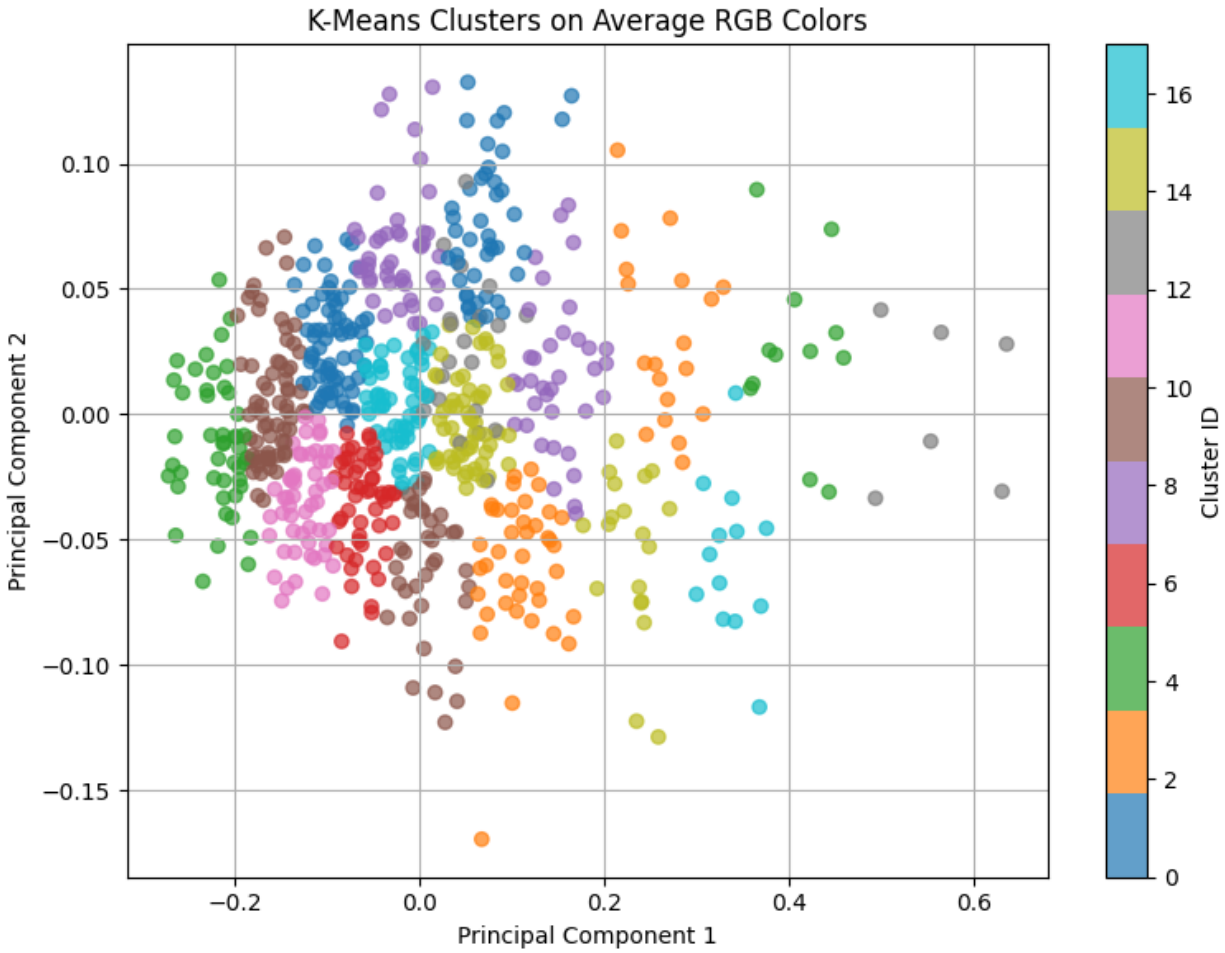


Figure 13: PCA-reduced RGB space colored by K-Means cluster ID.

```
Per-Type Top-2 Accuracy:
Bug       : 33.33%
Dark      : 16.67%
Dragon    : 0.00%
Electric  : 40.00%
Fairy     : 0.00%
Fighting  : 0.00%
Fire      : 0.00%
Flying    : 14.29%
Ghost     : 0.00%
Grass     : 0.00%
Ground    : 30.77%
Ice       : 0.00%
Normal    : 0.00%
Poison    : 30.77%
Psychic   : 0.00%
Rock      : 0.00%
Steel     : 0.00%
Water     : 60.87%
```

Visual showing what K-Means majority voting classified each cluster as.

Code Appendix

The complete code used in this project is available in the following Jupyter notebooks:

- `FinalProject_Multi_Classifier.ipynb` – multi-label typing classifier with image and color inputs
- `FinalProject_Binary_Classifier.ipynb` – binary Pokémon vs. animal classification model

Both notebooks are included in the project submission folder.

References

Arnaud58 (2019). Landscape pictures. https://www.kaggle.com/datasets/arnaud58/landscape-pictures. Background data set.

Betetta, J. (2021). Complete pokédex v1.0.0. https://www.kaggle.com/datasets/joshuabetetta/complete-pokedex-v100. Pokedex data set.

Bulbapedia (n.d.). Bulbapedia pokémon encyclopedia. Retrieved from https://bulbapedia.bulbagarden.net/wiki/Main_Page.

Iamsouravbanerjee (2021). Animal image dataset (90 different animals). https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals. Animals data set.

Maca11 (2016). All pokémon dataset. https://www.kaggle.com/datasets/maca11/all-pokemon-dataset. Pokedex data set.

Pokémon Database (n.d.). Pokémon database. Retrieved from https://pokemondb.net/.

Pokémon GO Hub (n.d.). Pokémon go hub pokédex. Retrieved from https://db.pokemongohub.net/.

Pratama, K. V. (2020). Pokémon images dataset. https://www.kaggle.com/datasets/kvpratama/pokemon-images-dataset. Pokemon images data set.

PyTorch Documentation (2024). Resnet18 architecture. https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet18.html. Accessed May 2025.

scikit-learn Developers (2024). scikit-learn documentation. https://scikit-learn.org/stable/. Accessed May 2025.

Serebii.net (n.d.). Serebii pokédex. Retrieved from http://serebii.net/.

Veekun (n.d.). Dex downloads. Retrieved from https://veekun.com/dex/downloads.

Appendix A:

Google Drive Folder Link