Using Support Vector Machines (SVM) to Classify Pokemon Images by Type

Author: Tyler Nelson

Abstract:

Pokemon avatars have become mainstream icons as the Pokemon brand has evolved from 1994 until the latest generation released in 2019, totaling 890 unique characters spanning 18 different types. In order to classify Pokemon into their type by their character image, the RGB images were reduced to 70 principle components, then used to train multi-class & one-vs-many SVMs. After thoroughly tuning various SVM models, however, they underperform other methods like k-means clustering and convolutional neural networks.

Background:

i. The Pokemon Franchise

Over the last 22 years, the Pokemon brand has grown from a small handhold console game series to a globally-recognizable brand. In 1994, Nintendo released the first 150 biomorphic characters, spanning caraciturized mice, turtles, and venus fly traps. After 8 generations of Pokemon, the collection has grown to 890 characters spanning household objects, machine parts, and even ice cream cones.

Each Pokemon is assigned one of 18 "types", loosely aligning with the natural element closest aligned with the character (i.e. "fire", "steel", "ice", etc.) The phenotypical aspects of a Pokemon (i.e. color, shape, and size) loosely correspond to a Pokemon's type.

Figure 1: Pokemon sprites and their Nintendo-assigned types

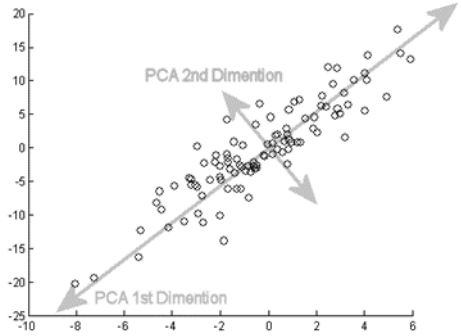
ii. Principal Component Analysis (PCA)

PCA is a longstanding statistical technique invented in 1901 by Karl Pearson. PCA is primarily used for dimension reduction, to make wide datasets more digestible in other algorithms like random forests or SVMs.

PCA is performed by finding the eigenvector (or principal component) that explains the maximum variance in a dataset. This process is repeated iteratively and a set of eigenvectors and eigenvalues (their relative importance) are returned.

It is good practice to preset a percentage of variance to preserve, prior to running PCA, to avoid data mining biases.

Figure 2: PCA applied to a 2-D scatterplot

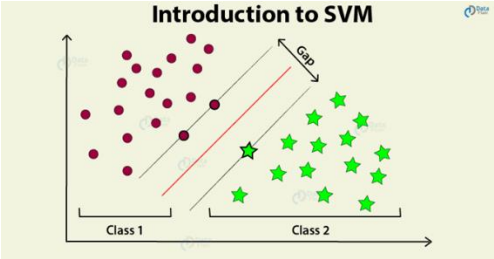


iii. Support Vector Machines (SVM)

The original SVM algorithm was invented by Vladimir Vapnik & Alexey Chervonenkis is 1963.

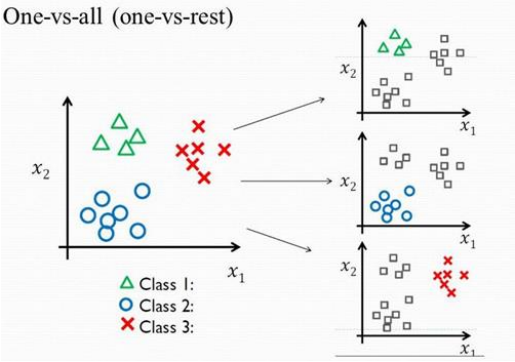
SVM's are linear classifiers that fit a hyperplane in between two (or more) groups. A key advantage of SVM's over many other linear classifiers is that SVM find optimal solutions.

Figure 3: Two-class SVM Illustration



For multi-class classification problems, to potentially increase accuracy or reduce computational complexity, one can chose to train multiple one-vs-rest SVMs (comparing each class to the collective rest) or one-vs-one SVMs (comparing each class with each other class). These do introduce additional considerations, given one-vs-rest SVMs will result in k models, based on k classes, where are one-vs-one SVMs will result in (k! / 2) models.

Figure 4: One-vs-all SVM



Data:

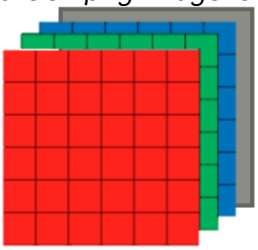
i. Data set

The dataset was scraped from the public website Serebii.com, a community forum for Pokemon fans. There are no copyright infringements, as Pokemon images are public domain, and Serebii owns no specific rights to the images.

A total of 721 Pokemon images were scraped, one for each Pokemon in generations one through six.

The format of these images follow 120 x 120 pixels x 4 color channels for 57600 unique pixels. These color channels follow .png format, with a red, green, blue, and alpha (transparency) channel.

Figure 5: .png image format



120 x 120 x 4 channels
(RGBA .png format)

Throughout the research, numerous data cleaning techniques were attempted, such as greyscale conversion. For the methodology detailed below, the images were shrunk to 60x60x3 channels, to reduce to ~18% their original size. This was done to reduce processing time for PCA and SVM training, which will be detailed further in the Methodology section.

As for the data labels, each image is labeled with "primary type", spanning 18 levels.

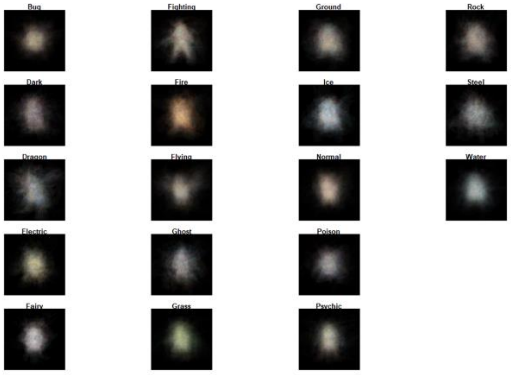
Last but not least, the images are handled as .cimg format type from the imager package, once loaded into R. Imager has a plethora of image processing functions, with some used for data augmentation.

ii. Data exploration

In order to enhance intuition around type classification, each image was grouped into its respective type group, each pixel value was averaged, then plotted.

Referencing the plot below, key aspects like color, shape, and size help differentiate each type. For example, "ice" types appear to be a white or sky blue color, where as "fire" types are an orangish red.

Figure 6: "Average" image by type



iii. Data augmentation

In order to increase the size of data to hypothetically increase accuracy, I artificially augmented it by 9x by performing the following manipulations:

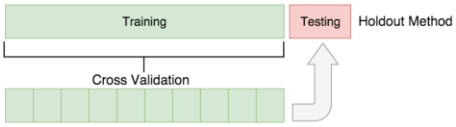
- Rotation +30 degrees
- Rotation -30 degrees
- Flip over x-axis
- Flip over y-axis
- Translation 10 pixels up
- Translation 10 pixels down
- Translation 10 pixels right
- Translation 10 pixels left

Figure 7: Example of Rotated & Flipped Images

iv. Training vs test set

The data was split with 80% reserved for training and 20% for testing, leaving room to tune the SVM using 10-fold cross-validating

Figure 8: Training vs Test Set with 10-fold cross-validation

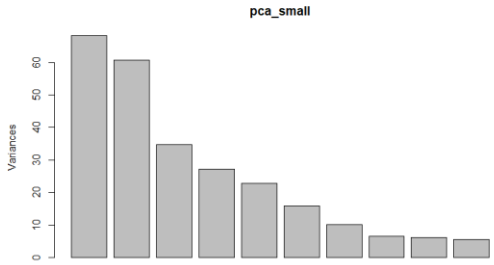


Methodology:

i. Performing Principal Component Analysis

When performing PCA on our 56400 pixels per image, it was preselected to retain 80% of the original variance, so ended up using 70 principle components. The first 10 are shown below in Figure 9.

Figure 9: Scree Plot for top 10 PC's



ii. Tuning Multi-Class and One-vs-Rest SVMs using 10-fold Cross-Validation

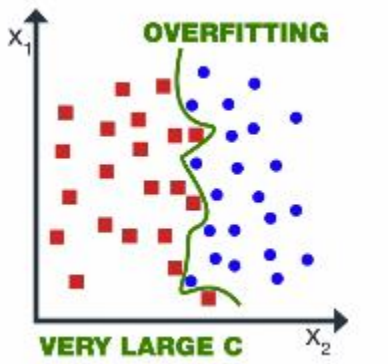
First, a multi-class SVM was tuned for optimal gamma and cost parameters. For comparison, one-vs-rest SVMs were formulated using one-hot encoding on each of the 18 types. Then, in training each of the 18 SVM models, we calibrated each of their gamma and cost variables.

Multiple kernels were tested, settling on radial kernels for their gamma & cost levers to calibrate. After testing each pair across the following sets ɣ = [.05, 0.2,0.5, 1.0, 1.5, 2] and C = [.01, 1, 4, 10, 20, 30, 60], the pair with the best performance of .853 was gamma of 0.2 and a cost of 10.

Figure 10: Bias-variance Tradeoff Between Gamma & Cost Parameters

	Large Gamma	Small Gamma	Large C	Small C
Variance	Low	High	High	Low
Bias	High	Low	Low	High

Figure 11: Diagram illustrating overfitting



The gamma parameter, explicitly adjusting the radius of comparison, ended up rather small, hinting that the highest performance model used a small radius of comparison. The highest-performance model also returned a small cost, meaning there was a low penalty for misclassification.

Figure 12: Tuning Results for SVM

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.2   10

- best performance: 0.8534256

- Detailed performance results:
   gamma cost     error dispersion
1    0.2   10 0.8534256 0.01539797
2    0.5   10 0.8540029 0.01543885
3    1.0   10 0.8540029 0.01543885
4    1.5   10 0.8540029 0.01543885
5    2.0   10 0.8540029 0.01543885
6    0.2   20 0.8534256 0.01539797
7    0.5   20 0.8540029 0.01543885
8    1.0   20 0.8540029 0.01543885
```

iii. Measuring Accuracy

After each of the 18 SVM's are calculated, the inferred probabilities for each SVM was compared for each row. The maximum probability was used to vote on the type for the given Pokemon image and compared this against the true type. Scaled probabilities were also used to control for class size in the

dataset, but did not result in better performance.

Figure 13: Voting Method

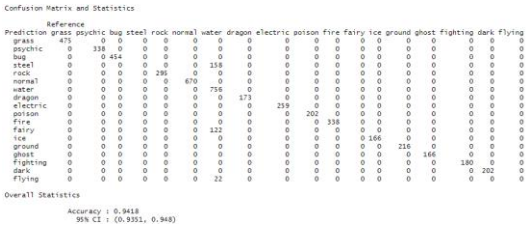
Probability of being x-type

Image #	bug	dark	dragon	electric	fairy	fighting
1	2.835320e-02	0.0185931629	0.017954804	0.0216330968	0.0147856154	1.860716e-02
2	2.845956e-02	0.0186248715	0.017937248	0.0217176416	0.0148142065	1.858400e-02
4	2.843922e-02	0.0186354087	0.018032551	0.0217133376	0.0147911930	1.859130e-02
5	2.857808e-02	0.0185792424	0.018028617	0.0216583012	0.0147985154	1.859198e-02
6	2.844717e-02	0.0185168100	0.017938539	0.0217323771	0.0148144566	1.864615e-02

iv. Examining Trained Model & Misclassified Training Images

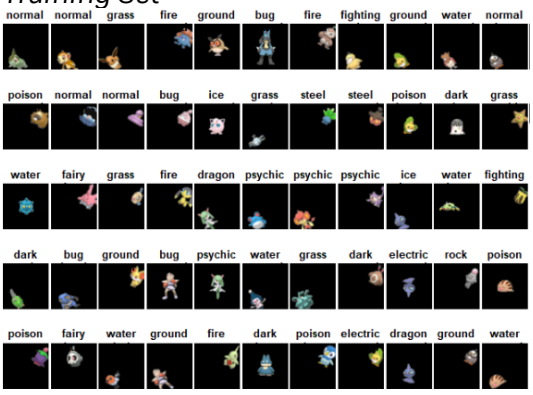
Accuracy across our trained model remains high at 94%, as expected.

Figure 14: Confusion Matrix of Trained Images



Looking at which images were indeed misclassified, intuitive trains show smaller icons that were rotated, and now find themselves in corners of the image screen.

Figure 15: Misclassified Images in Training Set



Results:

i. Classification Accuracy on Test Set

Testing both a multi-class vs. one-vs-rest approach, we show the SVM classification for both along the first two principal components.

Figure 16: Multi-class SVM Results

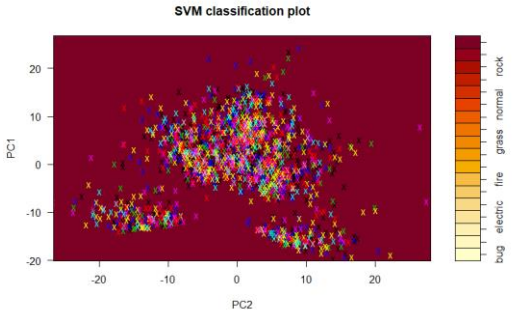


Figure 17: Confusion Matrix of Test Set (Multi-class)

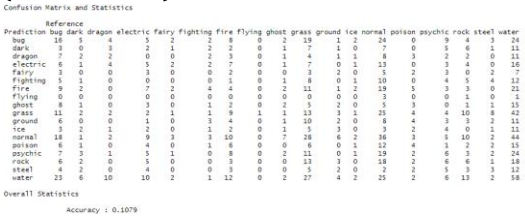


Figure 18: One-vs-Rest SVM (predicting if "water" type)

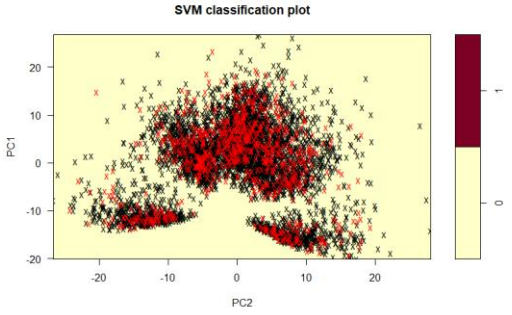


Figure 19: Confusion Matrix of Test Set (One-vs-rest)

Confusion Matrix and Statistics

Prediction \ Reference	grass	psychic	bug	steel	rock	normal	water	dragon	electric	poison	fire	fairy	ice	ground	ghost	fighting	dark	flying
grass	0	0	0	0	0	0	119	0	0	0	0	0	0	0	0	0	0	0
psychic	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0
bug	0	0	0	0	0	0	113	0	0	0	0	0	0	0	0	0	0	0
steel	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
rock	0	0	0	0	0	0	74	0	0	0	0	0	0	0	0	0	0	0
normal	0	0	0	0	0	0	167	0	0	0	0	0	0	0	0	0	0	0
water	0	0	0	0	0	0	189	0	0	0	0	0	0	0	0	0	0	0
dragon	0	0	0	0	0	0	43	0	0	0	0	0	0	0	0	0	0	0
electric	0	0	0	0	0	0	65	0	0	0	0	0	0	0	0	0	0	0
poison	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
fire	0	0	0	0	0	0	85	0	0	0	0	0	0	0	0	0	0	0
fairy	0	0	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0
ice	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0
ground	0	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0
ghost	0	0	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0
fighting	0	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0
dark	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0
flying	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0

Overall Statistics

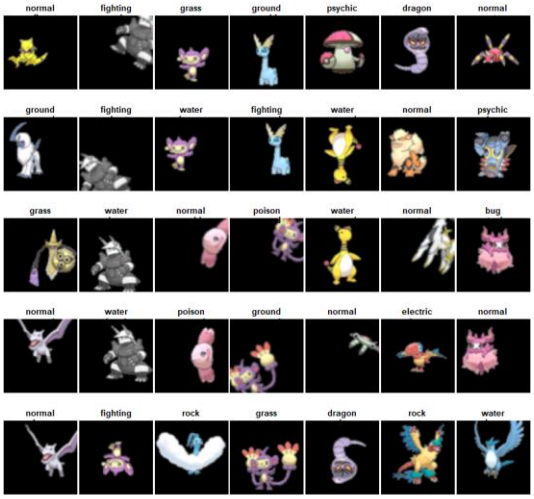
Accuracy : 0.1457

The first takeaway is that a multi-class approach, both empirically and intuitively, cannot differentiate classes from each other, with accuracy of ~11%. The second is that even in the one-vs-rest type for the largest class (water), performance remains poor. In Figure 18, the 1 ("water") zone is hidden, meaning the first & second principal component are not good differentiators between water & non-water type Pokemon. This was iteratively confirmed out to the 12th principal component, with distributions of water vs non-water type Pokemon appearing increasingly non-differentiable, pointing to a transformed dataset unable to be differentiated using multi-class nor one-vs-rest SVMs. This is empirically validated by test accuracy of ~15%, simply guessing 'water' type Pokemon, given it's the highest represented class in the dataset.

ii. Analysis of Mis-Classified Test Images

Looking at accuracy and precision across the 18 SVM's, numerous misclassified Pokemon are presented below. Unlike the training set, which saw small & rotated images misclassified, in the test set, numerous Pokemon across sizes & translations are misclassified.

Figure 20: Misclassified Images in Test Set



Conclusion:

i. Comparison to Other Methods

Comparing one-vs-rest SVM, k-means clustering, and CNN methods (published by Henrique Soares, using the same data set, but with minor data cleaning differences), average test set accuracy achieved was 15%, 33%, and 25-39%, respectively, placing PCA + SVM's at the lowest methodological preference compared to k-means clustering and CNNs. Overall, however, Pokemon types remain heterogenous classes that are difficult to classify using SVMs, clustering, or CNN methods.

ii. Future Research Areas

After observing numerous methods, future research areas that may warrant additional exploration include:

a) **K Nearest Neighbor (KNN) Clustering** - KNN's focus on relative proximity makes it an attractive candidate to investigate. Numerous Pokemon have multiple stages that relate to each other. Given these stages share characteristics like

color & shape, KNN could be a great method on picking up on these phenotypical attributes.

b) **Ensemble methods** – Given clustering & PCA methods produce unique features, perhaps it would improve accuracy to combine both into an ensemble method. In addition, tagged data accompanies the original dataset, like "stage" (baby, middle, or senior Pokemon) and "generation" (1994 vintage, 2000 vintage, 2019 vintage, etc.) which could be additional features to enhance prediction accuracy.

c) **Increased data** – Given the preceding method used only 1 base sprite for each Pokemon, if future research included additional images of each Pokemon, perhaps this could improve both accuracy & generalizability.

Works Cited:

1. Dawson, Carl. "A Guide to SVM Parameter Tuning." *Medium*, Towards Data Science, 26 Sept. 2019, towardsdatascience.com/a-guide-to-svm-parameter-tuning-8bfe6b8a452c.
2. Henrique Soares, Posted by JGS. "Who Is That Neural Network?" *Journal of Geek Studies*, 24 Feb. 2019, jgeekstudies.org/2017/03/12/who-is-that-neural-network/.
3. Hou, Janpu. "Image Classifier with SVM after Eigenfaces." *RPubs*, rpubs.com/JanpuHou/469844.
4. *Imager: an R Package for Image Processing*, dahtah.github.io/imager/imager.html.
5. Kelly, Ryan. "Support Vector Machines." *RPubs*, rpubs.com/ryankelly/svm.
6. Petukhov, Dmitrii. "Pokémon Recognition." *Medium*, Medium, 17 Jan. 2016, medium.com/@dimart/pokémon-recognition-d3ad5cadc61e.
7. "Principal Component Analysis with R Example." *Aaron Schlegel's Notebook of Interesting Things*, 19 Jan. 2017, aaronschlegel.me/principal-component-analysis-r-example.html.

Data, Labels & Script:
https://github.com/tylerjnelson8/PokemonClassification_April2020