What's that Pokémon?

Tariq Zahroof
(tzahroof@stanford.edu)
Stanford University

Abstract

As of June 2018, Nintendo's Pokémon series was the highest-grossing international franchise of all time, earning $60 billion since creation. The Pokémon video games are integral part to the franchise, where trainers identify the typing of Pokémon to deal super-effective hits and win. But, as Nintendo's franchise becomes increasingly saturated, are the visual designers maintaining an informative visual design to assist players and ensure player retention?

I explored 2 neural network architectures, different due to input, to output a typing label vector (18 types; at least 1 but no more than 2 types per Pokémon):

- Convolutional Neural Network (CNN): Input of 64x64x3 Pokémon image
- Shallow Neural Network (SNN): Input of 1-Hot Vector of a Pokémon's colors

Data

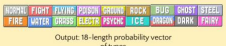
I custom-made my dataset. Pokémon sprites were harvested from Pokémon Showdown!, an online competitive Pokémon battling simulator. Each new Pokémon game had the previous generation's Pokémon and new, unseen Pokémon. Sprites were saved as 3-channel RGB images, with 18-length vector corresponding type labels. Images included both front and back portraits.

In the spirit of the games, the test set was the from the latest Generation (7). Data was divided into Train/Test, as Train-Dev was not representative of true distribution. With 807 unique Pokémon, there were 5160 training images and 366 test images, fed in at minibatch sizes of 32.

CNN:
- The following data augmentation was randomly performed:
  - Rotation
  - Translation
  - Scaling
  - Horizontal flip

SNN:
- The popular Python colorgram package was used to extract the top 5 colors.
  - The colors were saved as a 15-unit vector, stacked on top of each other.

Input: minibatch of augmented images



Output: 18-length probability vector of types

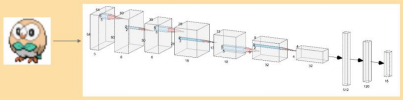
All Pokémon sprites are original property of Nintendo.

Models

Both networks used Adam optimization with ReLU activation functions for the hidden layers. A final sigmoid activation function was applied to the final layer to obtain the type probabilities. Additionally, both networks were trained on cross-entropy loss and a learning rate of 0.01.

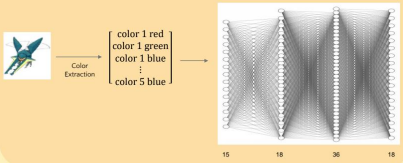
CNN

- RGB 64x64 image input
- 3 (Convolution -> Max Pool) layers -> 3 FC layers;
  - 5x5 Filter, 1 stride, 0 padding
- Batch normalization and dropout applied after each Max Pool layer



SNN

- 1-hot vector embedding input
- 3 FC layers
- Batch normalization and dropout applied after each activation layer



Loss Function and Prediction

Both networks outputted an 18-length vector from a sigmoid activation layer. As such, traditional cross-entropy loss was used for calculation.

$$L = \frac{-1}{N}\sum_{i=1}^{N} y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)$$

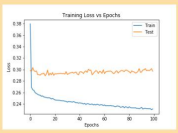
The loss function trained a vector of probabilities indicating the likelihood of the Pokémon belonging to a type. The following prediction algorithm was used to determine the Pokémon's type combination:

- Choose the 2 of the types with the largest probabilities
- If either has a probability > 0.2:
  - Set the type combination as the types that satisfy the above condition
- Else:
  - Set the type combination as the type with the largest probability

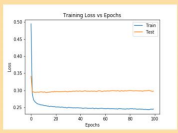
Results

Several variations of the CNN and SNN models (different number of hidden layers, layer size, learning rate, etc.) were tried before choosing the following models. Average F1 score was used to evaluate model performance, although recall, precision, and accuracy metrics were collected for error analysis.

CNN



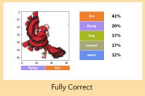
SNN



Model Performance

Model	F1	Recall (%)	Precision (%)	Top 3 Accuracy (%)	Top 5 Accuracy (%)
CNN	0.24	19	24	32	46
SNN	0.20	17	25	28	41
Tariq	0.48	46	50	66	89

Prediction Example



Fully Correct

Partially Correct

Incorrect

References

- J.L. Yuan and T. Fine, "Neural-network design for small training sets of high dimension," in *IEEE Transactions on Neural Networks*, 1998.
- K. Yao, G. Zweig, et al, "Recurrent neural networks for language understanding," *INTERSPEECH*, 2013.
- G. Hinton, N. Srivastava, et al, "Improving neural networks by preventing co-adaptation of feature detectors," in *arXiv*, 2012.
- H. Soares, "Who is that Neural Network?" *Journal of Geek Studies*, 2017.

Discussion

Results were decent, although predictably low. Regardless of model tuning and structure, the test error would stabilize at 0.3 (unlike the train-dev error, which continued to decrease), suggesting that the training distribution is not representative of the test distribution. This makes sense, since Nintendo wants to create new, original Pokémon, and would thus strive to make them different from previous generations.

To approximate Bayes error, I also predicted Generation 7's type combinations. Despite being a veteran player and intimately familiar with generations 1-6, I only had a F1 score of 0.48, suggesting that some Pokémon types were nebulous or misleading. For example, the incorrect prediction example can be reasonably guessed as any combination of [bug, steel, flying, dragon]. Only its name, Vikavolt, would suggest its electric typing, therefore implying that naming is important for type prediction. Furthermore, Vikavolt's special ability is "Levitate," which gives it some of the properties of a flying type.

Finally, the training set was relatively small and over-sampled, as generation 1 Pokémon had 7 different sprite incarnations, while later generations were less represented. Thus, general sprite shape (e.g. the shape of birds and fighters) were harder for the CNN model to identify by the limited unique Pokémon. As such, it relied on identifying color-based types, and therefore avoided shape-based types (hence the "nan" in the precision column).

CNN

	Recall	Precision		Recall	Precision
normal	49%	13%	fire	33%	41%
fighting	0%	nan%	water	78%	32%
flying	4%	29%	grass	31%	33%
poison	29%	58%	electric	32%	67%
ground	0%	0%	psychic	13%	15%
rock	5%	40%	ice	0%	0%
bug	0%	0%	dragon	0%	nan%
ghost	0%	nan%	dark	22%	12%
steel	0%	0%	fairy	0%	nan%

SNN

	Recall	Precision		Recall	Precision
normal	35%	16%	fire	7%	33%
fighting	0%	nan%	water	78%	40%
flying	20%	11%	grass	48%	71%
poison	17%	29%	electric	32%	40%
ground	0%	nan%	psychic	28%	15%
rock	7%	25%	ice	0%	nan%
bug	5%	17%	dragon	0%	nan%
ghost	0%	nan%	dark	0%	0%
steel	0%	nan%	fairy	2%	6%

Tariq

	Recall	Precision		Recall	Precision
normal	77%	45%	fire	56%	50%
fighting	42%	24%	water	60%	53%
flying	53%	47%	grass	57%	50%
poison	50%	57%	electric	44%	36%
ground	40%	40%	psychic	15%	33%
rock	38%	71%	ice	0%	0%
bug	85%	79%	dragon	70%	88%
ghost	36%	62%	dark	33%	6%
steel	67%	60%	fairy	24%	36%

Future Work

1. Train the CNN on real-world images to guess Pokémon distribution from the real-world distribution (pictures of dragons, birds, etc.)
2. Use Pokémon names as inputs to guess typing association
3. Build a CNN for each Pokémon type
4. Complain to Nintendo