PokeNet: Predicting Pokemon Card Features Through Deep Learning

Eddy Albarran and Piper Keyes {albarran, pckeyes}@stanford.edu
CS230 Spring 2018

Introduction

We aim to develop Deep Learning networks (PokeNet, collectively) to predict various features of Pokemon cards. Specifically, our models predict type, hit points (HP), and card price. We employed three architectures (5 layer neural net, convolutional neural net, and DenseNet) to accomplish these tasks. Ultimately, we hope to expand the functionality of PokeNet to create a real life version of the famed and beloved Pokedex, fulfilling a life-long dream of many millenials that aspired to become Pokemon Masters.

Dataset and Preprocessing

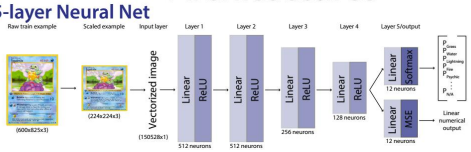
We scraped our dataset from pkmncards.com and rescaled the card image files to 224x224 pixels. We vectorized the images and concatenated the vectors into a single X matrix. Y labels were also vectorized. We collected all 9791 Pokemon cards in existence and divided them into Train/Dev/Test sets as follows: Train = 7832 cards (~80%), Dev = 979 cards (~10%), Test = 980 cards (~10%).

Earlier sets examples

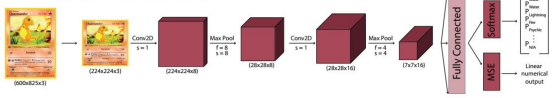
Later sets examples

Architectures

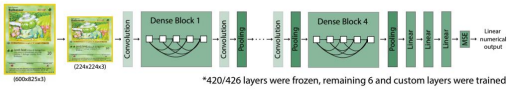
5-layer Neural Net



Convolutional Neural Net



DenseNet-121



*420/426 layers were frozen, remaining 6 and custom layers were trained

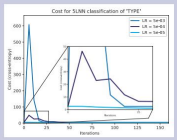
Predicting Pokemon Type

5-layer Neural Net

Parameter tuning

	Iteration 1: identify optimal learn rate			Iteration 2: address bias by increasing epoch number			Iteration 3: adress variance with L2 norm			
Learning rate	5.0e-3	5.0e-4	5.0e-5	5.0e-4	5.0e-4	5.0e-5	5.0e-4	5.0e-4	5.0e-5	5.0e-5
Epoch num	150	150	150	250	350	250	350	350	350	350
Beta value							1.0e-3	1.0e-4	1.0e-3	1.0e-4
Train accuracy	0.146	0.930	**0.959**	0.963	0.979	0.984	**0.996**	**0.997**	0.997	0.999	0.995
Dev accuracy	0.158	0.914	**0.953**	0.951	0.951	0.959	**0.971**	**0.978**	0.976	0.971	0.973
Final cost	2.334	0.370	0.225	0.178	0.104	0.101	0.050	0.618	0.114	0.780	0.136

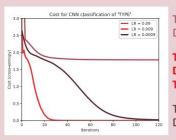
Network training



Final model (5e-4 lr, 350 epochs, 0.001 beta) test accuracy: 0.9755

Convolutional Neural Net

Network training

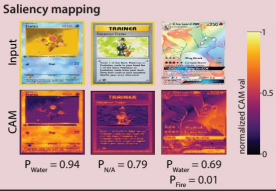


Train: 0.62
Dev: 0.54

Train: 0.97
Dev: 0.97
Test: 0.97

Train: 0.95
Dev: 0.94

Saliency mapping



$P_{Water} = 0.94$ $P_{N/A} = 0.79$ $P_{Water} = 0.69$ $P_{Fire} = 0.01$

Predicting Pokemon HP

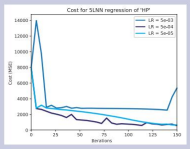
5-layer Neural Net

Parameter tuning

Learning rate	5.0e-3	5.0e-4	5.0e-5	5.0e-4
Epoch num	150	150	150	250
Train mse	61.598	23.804	24.732	18.730
Dev mse	61.097	23.688	24.755	20.146
Final cost	5367.1	577.6	661.8	363.9

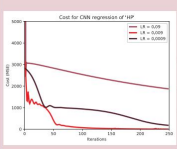
Final model (0.0005 lr, 250 epochs) test rmse: 20.819

Network training



Convolutional Neural Net

Network training



Train: 43.23
Dev: 42.69

Train: 8.57
Dev: 9.82
Test: 8.59

Train: 13.39
Dev: 15.99

Saliency mapping

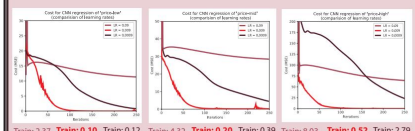
Pred HP: 93.2
True HP: 100

Pred HP: 47.1
True HP: 30

Predicting Card Price

Convolutional Neural Net

Network training



Train: 2.37 **Train: 0.10** Train: 0.12
Dev: 2.43 **Dev: 0.12** Dev: 0.16
Test: 0.11

Train: 4.32 **Train: 0.20** Train: 0.39
Dev: 5.97 **Dev: 0.32** Dev: 0.47
Test: 0.24

Train: 8.03 **Train: 0.52** Train: 2.79
Dev: 2.65 **Dev: 0.65** Dev: 2.68
Test: 0.58

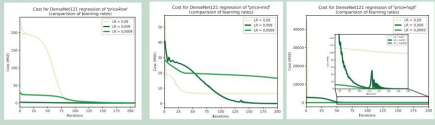
Saliency mapping

Pred mid price: $49.90
True mid price: $64.90

Pred mid price: $51.27

DenseNet-121

Network training



Train: 2.09 **Train: 0.07** Train: 0.19
Dev: 2.08 **Dev: 0.09** Dev: 0.16
Test: 0.10

Train: 2.55 **Train: 0.15** Train: 0.36
Dev: 1.59 **Dev: 0.24** Dev: 0.58
Test: 0.26

Train: 9.76 **Train: 0.30** Train: 1.59
Dev: 2.07 **Dev: 0.39** Dev: 1.43
Test: 0.59

References

[1] Huang, Gao; Liu, Zhuang; van der Maaten, Laurens; Weinberger, Kilian Q. Densely Connected Convolutional Networks, 2018.
[2] https://github.com/flyyufelix/DenseNet-Keras
[3] Zhou, Bolei; Khosla, Aditya; Lapedriza, Agata; Oliva, Aude; Torralba, Antonio. Learning Deep Features for Discriminative Localization, 2016.
[4] https://pkmncards.com
[5] https://github.com/philipperemy/tensorflow-class-activation-mapping/blob/master/class_activation_map.py
[6] CS230 programming assignments "Tensorflow Tutorial" and "Convolutional model Application v1"