Magical Neural Networks (MNN)

Kaiser Pister

December 16, 2017

Abstract

Magic the Gathering is a trading card game played by millions of people across the world. The game consists of approximately 18,000 unique cards which each have their own uses in the game. Each year, three new sets of about 200 cards are release by the game's creators, Wizards of the Coast. These cards are often "spoiled" bits at a time; only the name of a card will be revealed initially, later that day the rest of the card will be unveiled. By harnessing the power of deep learning neural networks, I have created a model to predict information about a partially revealed card.

1 Introduction

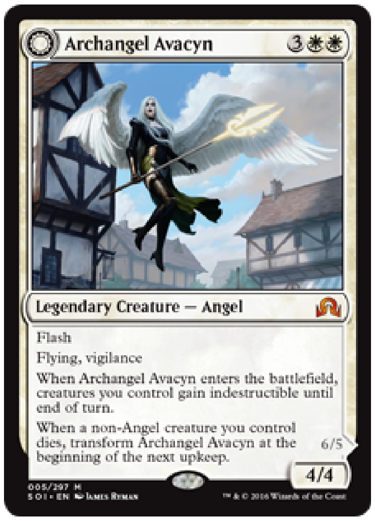
A normal Magic card has five main parts which I examine. In the top left corner is the card's **name**, the upper middle is the card's **art**, the text immediately below the image is the card's **type**, and finally, the **color** of the card is depicted by the color surrounding it all. As an example, refer to figure 1.
Let

$$Colors = \{ \text{ white, black, red, green, blue, } \varepsilon \ \}$$

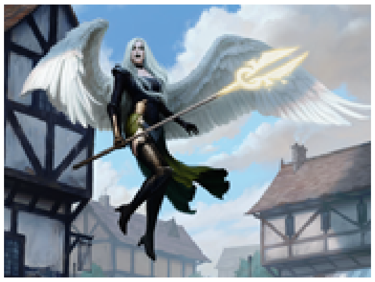
$$Types = \left\{ \begin{array}{l} \text{creature, land, instant, sorcery, en-} \\ \text{chantment, artifact, planeswalker} \end{array} \right\}$$

Every card has a set of colors $c \subset Colors$ and every card has a singular type $t \in Types$. So for the example in figure one, we would have a table like the one below.

Piece	**Instance**
Name	Archangel Avacyn
Type	Creature
Color	white
Art	figure (b)


Archangel Avacyn
3
Legendary Creature — Angel
Flash
Flying, vigilance
When Archangel Avacyn enters the battlefield, creatures you control gain indestructible until end of turn.
When a non-Angel creature you control dies, transform Archangel Avacyn at the beginning of the next upkeep.
6/5
4/4
005/297 H
SOI • EN James Ryman
™ & © 2016 Wizards of the Coast

(a) Archangel Avacyn



(b) Just the Art

Figure 1: A card and its art

Now that all of the pieces of a Magic card have their place, we can think of what to do with them. For my project, I looked at the ability of a neural net to predict a piece P, given some set of the remaining pieces. The first effort I made was in predicting the color of a card, given its name. In Magic the Gathering, each color has themes that relate to it. For example, Green is synonymous with life and large monsters (like dinosaurs), while Blue is connected to wizards who casts spells. I hope to discover a corolation between the name of a card and the theme of a card, which would then be used to predict the color. Similarly, I will attempt to predict the type of a card using the name. Finally, I experiment with Convolutional Neural Networks to predict these same features using the artwork on a card.

2 Color Predictions

Using the text name of a card, I aimed to predict which of the five colors (or null) that card would be. I had to take into account a couple considerations in constructing my network and dataset. First, a text name is not something I can immediately feed into a neural net. Second, I am working with a small dataset of approximately 18,000 unique cards. And finally, some cards have multiple colors, which might mean they need to be treated specially.

2.1 Fully Connected Architecture

Confronting the first issue I mentioned above, I decided to use Word2Vec to translate the card names into vector form. Throughout my project, I experimented with many different implementations of word to vector which I will document with each architecture. To start, I used Word2Vec's pretrained model which learned from english Wikipedia and has 1,000 dimensions[1]. Using this model has issues: primarily, not all the names in the Magic universe have an equivalent vector in the trained model. For these cards, I chose to ignore those words. Then, any cards that have no vector form at all, I remove from my dataset. While trimming an already small dataset is not the best practice, it was a necessary step to take at the time. I also lost 94 words due to lack of UTF8 compatibility – cards with characters an umlaut for example would cause problems in the Word2Vec model. This model had an space problem of being 8GB large, leading to long loading and preprocessing times.

The first architecture I worked with was a single layer of 300 nodes, and a 20% dropout layer. I used a sigmoid activation function, learning rate of 0.001, and the Adam optimizer. Over time I changed the number of layers to 3, using 150, 100, and 50 nodes in each layer. I replaced the sigmoid with tanh then relu and increased the dropout layer to 60% (with dropout between each layer). This model is shown in the appendix for clarity.

2.2 The Dataset

I found my initial dataset from Kaggle Datasets. It contained a list of 8,000 cards printed from 1995-2007. At the time, I didn't realize I was only working with about 50% of all cards printed, but when I did realize my mistake, the data I received provided interesting enough insights into the evolution of the game that I figure they are worth noting below. Using the above architecture and this dataset, I trained on 7,000 card name vectors and tested with 1338, outputting a color vector. My network plateaued at 60% accuracy (random baseline of 18%).

After getting this initial result, I spent more time looking for a better, more complete catalog. In the end, I used MTGJson[2]. With the new dataset I have about 18,000 cards from 1995-2017. I separated this into 15,000 training and 3,000 testing (randomly shuffled). My model achieved 54% accuracy.

It was interesting to me that the full set of cards actually had lower accuracy than the half set of cards. My two proposed explanations of this are that either the network was lucker in the first training case (the training and test data had an easier split) or that over the course of the last 10 years, the creators of Magic the Gathering have taken more liberties with what the color of a card represents.

2.3 New Text Models

The next variable I manipulated was the word to vector model I was using. Searching around, I found a model trained on Google News. Each vector is 300 dimensions. Approximately 1,800 cards were ignored due to unknown names with this model. Using the same 150–100–50 architecture as above, this model received 50% accuracy. Even though it has 4% loss in accuracy, the advantage of this model is only being 2GB of space, compared to the 8GB for the 1,000 dimension Wikipedia model. I looked into Stanford's training model as well: Glove[3]. Using this 300 dimension model, I was able to achieve 54% accuracy. For practical purposes, Glove is the model I used from here on out due to its performance and 2GB memory footprint.

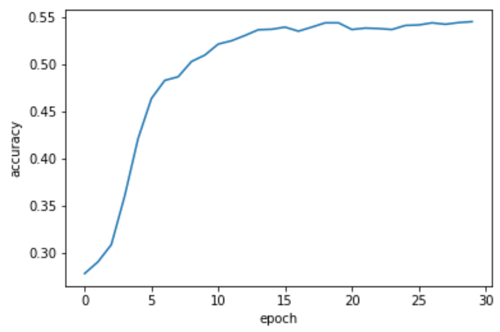


Figure 2: Glove, 150–100–50

3 Type Predictions

Next I attempted to predict the type of a card. Once again I started by predicting type based solely on the card name. Using the same architecture as the final instance of my color predictor, I now predict 1 of 7 different type categories for a card. The model results in 67% accuracy.

3.1 Combined Input

At this point, I started combining features to see if I would achieve better performance. First, I wanted to predict the type of a card given both the name and the color of the card. While it wasn't significant, the model improved by 4% to 71%. Predicting the color given both the name and the type had a similar effect of improving performance by about 3% to a total of 57%.

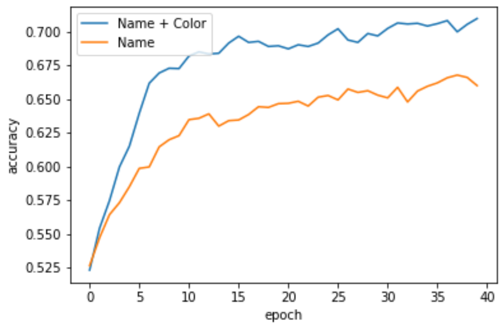


Figure 3: Predicting Types using Name + Color vs. Name

4 Convolution Experiments

I decided to expand my dataset in a new way by using the image of a card as input to predict classes of the card. Initially, this meant only using the art to predict card color and type. The structure of my network is very similar to the one mentioned before: I feed in a vector (in this case the image) and output a one–hot encoded result vector which indicates type or color. These models trained with a batch size of 128 for about 20 epochs on 13,000 card images. The model is shown in the appendix for clarity.

4.1 Just the Art

The final step I took in working on this project was constructing a CNN to predict color and type based on the image art alone. It is important to note that the image art is only the cropped art as shown in figure 1.

The dataset of the art was constructed on my own by crawling https://gatherer.wizards.com for card images and cropping them down to size[4]. I then fed these images into a simple CNN with the following results:
Color Prediction: 35%
Type Prediction: 53%

The accuracy of the Color Prediction is shown in figure 4.

4.2 The Full Card

After trying to work with just the image art and receiving low results, I wanted a boost, so I included the entire image. Obviously, I was expecting very high accuracies, because the color of a card is very clearly embedded in the image of the card itself. The type is also clear, but in text, which would be slightly

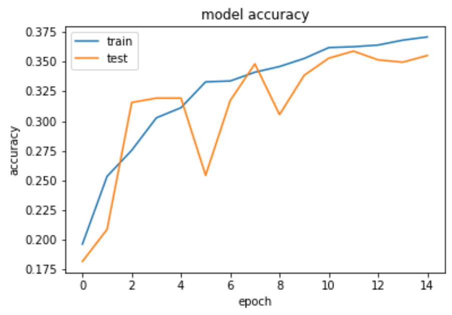


Figure 4: Color Prediction Accuracy

more difficult to predict. Using the same architecture as the above experiments but with twice as many features in each layer (eg. 256–256–128–128...).

My results are shown in the figure 5 and figure 6.

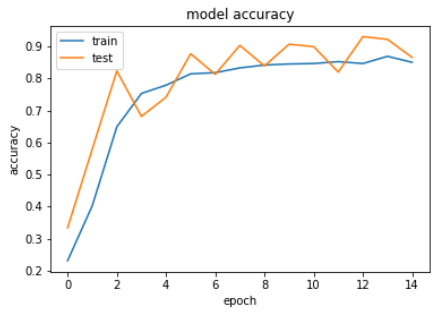


Figure 5: Predict Color on Full Card Image

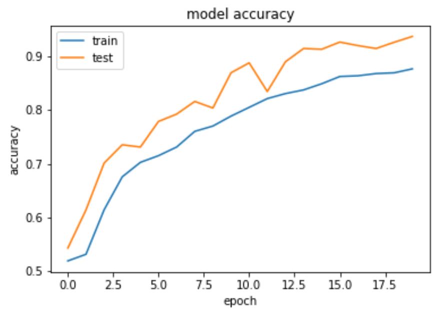


Figure 6: Predict Type on Full Card Image

4.3 Making Money

The final experiments I ran were to test if I could predict the price of a card given the full image. This task presented new challenges primarily in the form of acquiring a dataset (as explained in the bonus points section). I gathered my data from MtGGoldfish[5]. Eventually I was able to train my CNN on 15,000 card images and prices. I created buckets of prices at:

¡$0.15,¡$0.18,¡$0.25,¡$0.50,¡$1.50,¡$5.00, and above. These buckets were all fairly evenly full at approximately 2,500 items each. The largest bucket contains 4,000 / 15,000 cards meaning a semi-random predictor would guess correctly about 25% of the time.

My results of using the same architecture as previous experiments led to just over 23% accuracy meaning I cannot successfully predict price any better than just guessing. This is a section that I would like to work with more; I would improve my dataset, and think about other ways of approaching the model.

5 Bonus Points

Creating a usable dataset for my models to learn from was by far the hardest part of this project. Initially, I was able to work with a json file for all the card information; the difficulties arose when using the card images. No single site provides all 15GB of card images as a download, nor can you find simple a simple API to make image requests. As a result, the images must be scraped off of gatherer.wizards.com, and cross corrolated with the json file for information. Furthermore, I experimented with grabbing the prices of cards for an attempt to predict a card's price based on the card image. THERE ARE NO GOOD OPTIONS FOR GETTING THE PRICE. Many card sale websites are faulty and do not provide full pricing information on all cards; card prices are subject to change at a moments notice, so historical datasets are irrelevant and not preserved; successful card sale sites hide the prices behind a locked API; price information is hidden in badly written html on hard to scrape websites. Despite this, I was able to acquire the price of 10,000 cards and attempted to run my model on the dataset.

I DON'T KNOW IF THIS WILL SUCCEED.

6 Conclusion

In conclusion, I was able to predict color and type with decent accuracy using a simple word2vec encoding of the card name. Adding features helped in my predictions, but did not make major improvements. The artwork of a card does not provide very much indication of the card's color or type, although it gives more than a random guesser. Finally, as would be expected, the full card image performs above 90% accuracy on both color and type predictions.

6.1 Further Work

With more time, I would have liked to repeat many of my initial experiments with my current knowledge: changing more hyperparameters and cleaning the data more. I would also like to use my Image–Price dataset to predict what a card's price would be based soley on the image. If this model is accurate, it could be of monetary value when a new set is released.

References

[1] D. D. Team, "Deeplearning4j: Open–source distributed deep learning for the jvm." http://deeplearning4j.org, 2017.

[2] Sergio, "Mtg json." http://mtgjson.com, 2017.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[4] W. of the Coast LLC, "Gatherer database." http://gatherer.wizards.com, 1995–2017.

[5] MtGGoldfish, "Card price list." https://mtggoldfish.com/, 2017.

A Model Architectures

The architecture of my fully connected model.

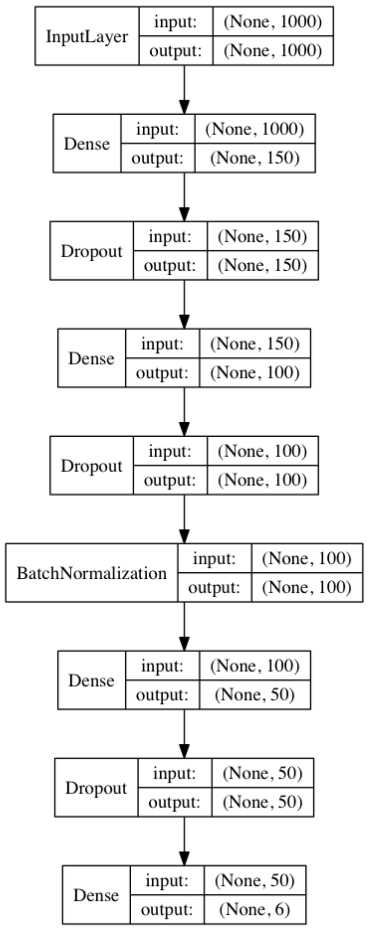


Figure 7: 150–100–50 Architecture, received 54% accuracy with the full dataset

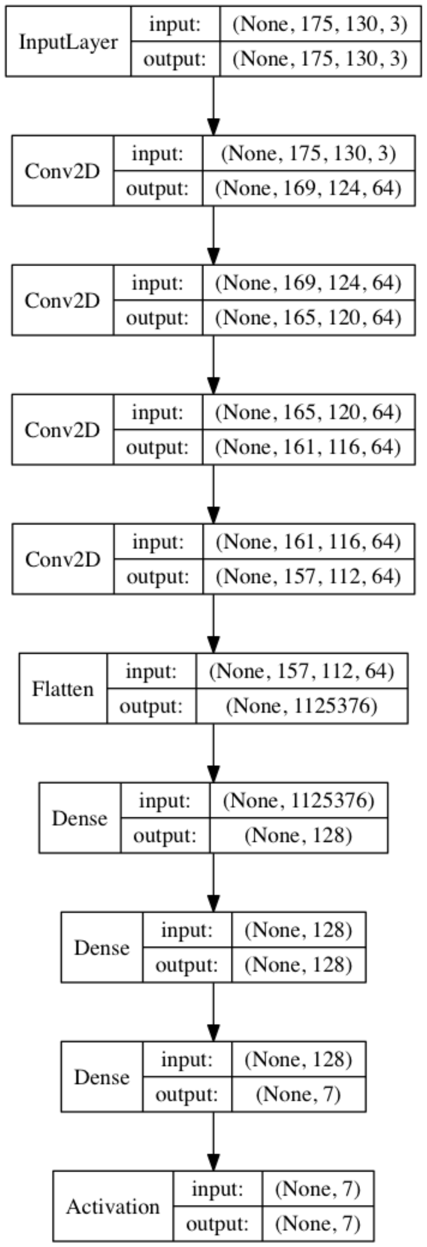

InputLayer
input: (None, 175, 130, 3)
output: (None, 175, 130, 3)
Conv2D
input: (None, 175, 130, 3)
output: (None, 169, 124, 64)
Conv2D
input: (None, 169, 124, 64)
output: (None, 165, 120, 64)
Conv2D
input: (None, 165, 120, 64)
output: (None, 161, 116, 64)
Conv2D
input: (None, 161, 116, 64)
output: (None, 157, 112, 64)
Flatten
input: (None, 157, 112, 64)
output: (None, 1125376)
Dense
input: (None, 1125376)
output: (None, 128)
Dense
input: (None, 128)
output: (None, 128)
Dense
input: (None, 128)
output: (None, 7)
Activation
input: (None, 7)
output: (None, 7)

Figure 8: The architecture of my CNN model. I doubled the features in the full card image tests. Given more time, I would do the same for the "Just the Art" tests. There are max pooling layers in between each convolutional layer that have been omitted to save space.