Deep Learning Game-Piece Balance in the Strategy Card Game *Magic: the Gathering*

Daniel F. Berenson
berenson@stanford.edu

1 – Introduction

Magic: the Gathering is a popular strategy card game. It is played by two players, each of whom brings their own deck of special cards. These cards represent spells and creatures that the players can cast and summon to do battle with their opponent. This project will focus specifically on creatures because they have better defined features.

Creatures are balanced based on characteristics of each creature. These include *power*, *toughness*, *keyword abilities*, and *subtypes*. The key feature that balances the effects and power levels of the creatures for an interesting gameplay experience is the *mana cost* (or *converted mana cost*, abbreviated *CMC*). Over the course of a game, both players build up their supply of a resource called *mana* that is required to summon creatures. In general, more powerful creatures have higher mana costs, which means they cannot be summoned until later in the game, and they require more of an investment.

When the game designers design new cards, they often start by deciding on the card's power level and abilities, and then need to choose an appropriate mana cost to make the card fair and balanced with respect to other cards that already exist. Alternatively, the designers might choose to design a card with a particular mana cost, and then need to adjust the power level and abilities to make the card fair and balanced (https://magic.wizards.com/en/articles/archive/making-magic/design-101-2003-04-21-0).

In addition, there are five *colors* in *Magic*: white, blue, black, red, and green. These represent different gameplay flavors: for example, specializing in swarming the battlefield with many small creatures, or playing a single larger creature. Each creature has a particular *color identity* representing which color it is associated with: creatures can have zero, one, or multiple colors in their color identity. Similarly as with mana cost, game designers might design a card's power level and abilities and then choose its color, or choose to design a card of a certain color and then need to fill in its power level and abilities. Sometimes there are multiple plausible choices for a color: a 4/4 creature with flying and vigilance has historically been a mono-white creature, but officially and infamously it could also be black-green. (https://magic.wizards.com/en/articles/archive/making-magic/make-choice-part-1-2018-02-12).

This project uses deep learning on the existing card pool to create a card-design decision support tool. Using other information about the card, it can predict an appropriate mana cost; or using other information about the card, it can predict an appropriate color identity.

2 – Data Set and Features

An online repository, https://mtgjson.com/, contains a list of all 20,000+ unique *Magic: the Gathering* cards. This data set is then narrowed down to the 10,000+ unique creature cards. The data set includes structured data on each creature including its power, toughness, keyword abilities, other rules text, creature subtypes (e.g., Angel, Warrior, Skeleton), creature color (white, blue, black, red, green, multicolored, or colorless), card rarity (rare cards are permitted to be more powerful for a given mana cost than common cards), and mana cost. Feature engineering makes these features interpretable by the neural network, for example converting a card with keywords "Flying, Vigilance" into binary entries in new features called kw_Flying and kw_Vigilance. In the basic model, the rules text on each card is simply converted to an integer number of words, based on the notion that the length of the rules text may correlate with other relevant features of the card. The NLP model uses natural language processing to better understand this free text.

3 – Approach

First, features are engineered as described above. Next, *y* variables are isolated for the two distinct model tasks: mana cost prediction and color identity prediction. Then, the data are split into training, dev, and test sets at a 60-20-20 ratio, which was chosen due to the relatively small total sample size of ~10k unique cards. Next, the data are scaled using a MinMaxScaler with the goal of equalizing the parameter gradients in all directions. A Sequential model with the hyperparameters listed under "Results" is compiled. For the mana cost prediction model (which has a regression task), there is a single output neuron with a ReLu activation function and a mean squared error loss function. For the color identity prediction model (which has a multi-class classification task because creatures can be positive for any number of the five colors), there are five output neurons each with sigmoid activation functions and binary crossentropy loss functions. Then, the models are trained. Finally, the models are evaluated. For the mana cost prediction model, a series of violin plots shows the correlation between true and predicted mana costs on the dev set. For the color identity prediction model, a series of precision/recall/F1 score tables are produced. As a sanity check, the user can also select an individual real card or create their own card, and ask the model to predict the mana cost and color.

For the NLP model, the text boxes of the cards are vectorized into integer indices and then represented as dense embeddings using the Glove model. Despite the large size of the Glove word database, there are still many words that appear on *Magic* cards that are not in Glove: these include unusual fantasy words like "treefolk," unusual compound words like "nondinosaur," and text box mana costs. To partially alleviate this problem, when a card's own name appears in its text box it is replaced by "CARDNAME" in a pre-processing step. To account for this, the Embedding layer of the model is set to trainable so that these uncommon words' embeddings can be learned. The model then uses an LSTM layer before several Dense layers and finally the output layers, just as in the vanilla version.

I was not able to complete work on the combined model because I lacked the required Keras chops. However, my approach was going to be to take the NLP model described in the previous

paragraph and truncate it after one dense post-LSTM layer. Then, the activations of that layer would be concatenated with the (processed, but not yet neural-networked) structured data used in the vanilla model. The resulting matrix, containing information about the structured data as well as the text of each card, would then be run through several dense layers before yielding the final output.

4 – Code

https://github.com/dfberenson/MtG

5 – Results

Hyperparameter tuning on the non-NLP model, evaluating the hidden layer depth, units per later, presence of dropout, dropout rate, and epochs of testing, was run for 200 trials on each model, which took about 2.5 hours on my CPU. The hyperparameters of the single best trials are below, along with the ranges for the top five trials. The MSE and F1 scores were evaluated on the dev set.

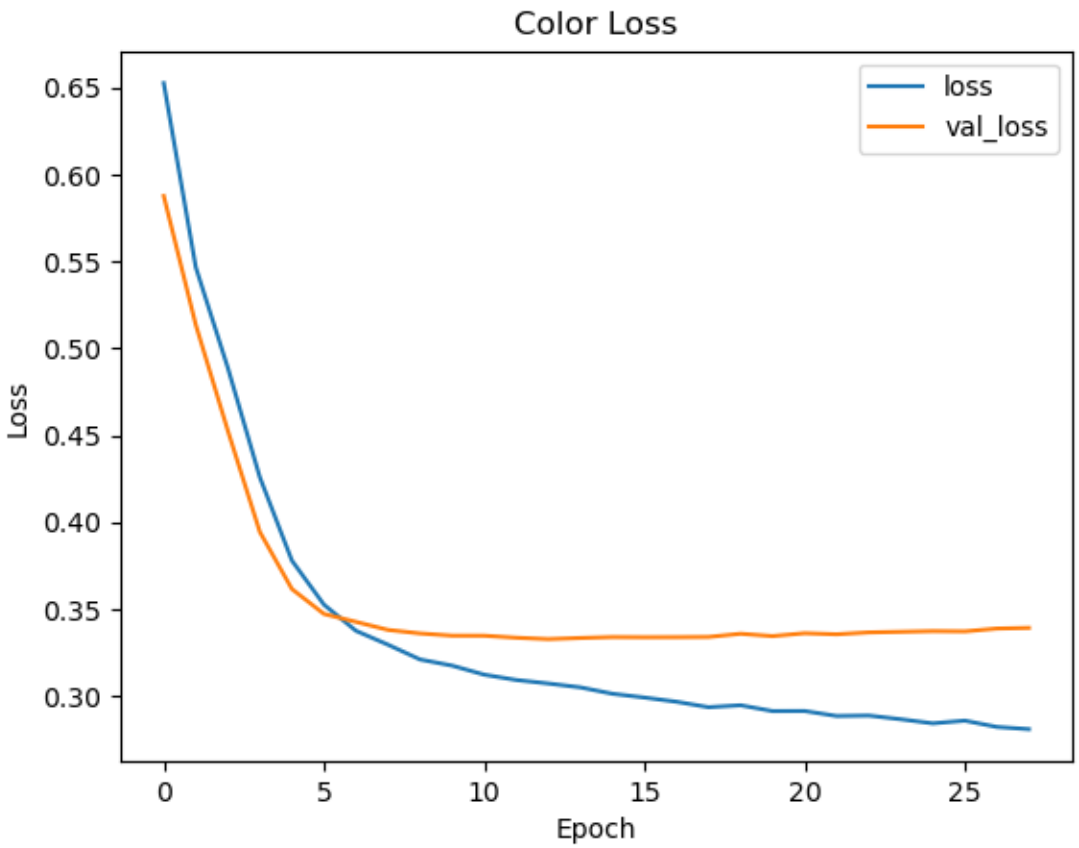
> The best hyperparameters for CMC were:
> Hidden dense layers = 2 *(range: 2,3,4)*
> Initial dense layer size = 128 *(range: 64,128,256)*
> Use dropout = True *(range: True,False)*
> Dropout rate = 0.2 *(range: 0.1,0.2, NA)*
> Epochs = 28 *(range: 19,25,28,39,15)*
> CMC MSE = 1.11 *(range: 1.11,1.12,1.13)*
> Color F1 = 0.695 *(range: 0.688,0.695,0.697)*
>
> The best hyperparameters for color were:
> Hidden dense layers = 3 *(range: 2,3)*
> Initial dense layer size = 64 *(range: 64,128,256)*
> Use dropout = False *(range: True,False)*
> Dropout rate = NA *(range: 0.01,0.05,0.1, NA)*
> Epochs = 29 *(range: 21,22,26,29,50)*
> CMC MSE = 1.23 *(range: 1.16,1.18,1.20,1.23,1.25)*
> Color F1 = 0.701 *(range: 0.699,0.700,0.701)*

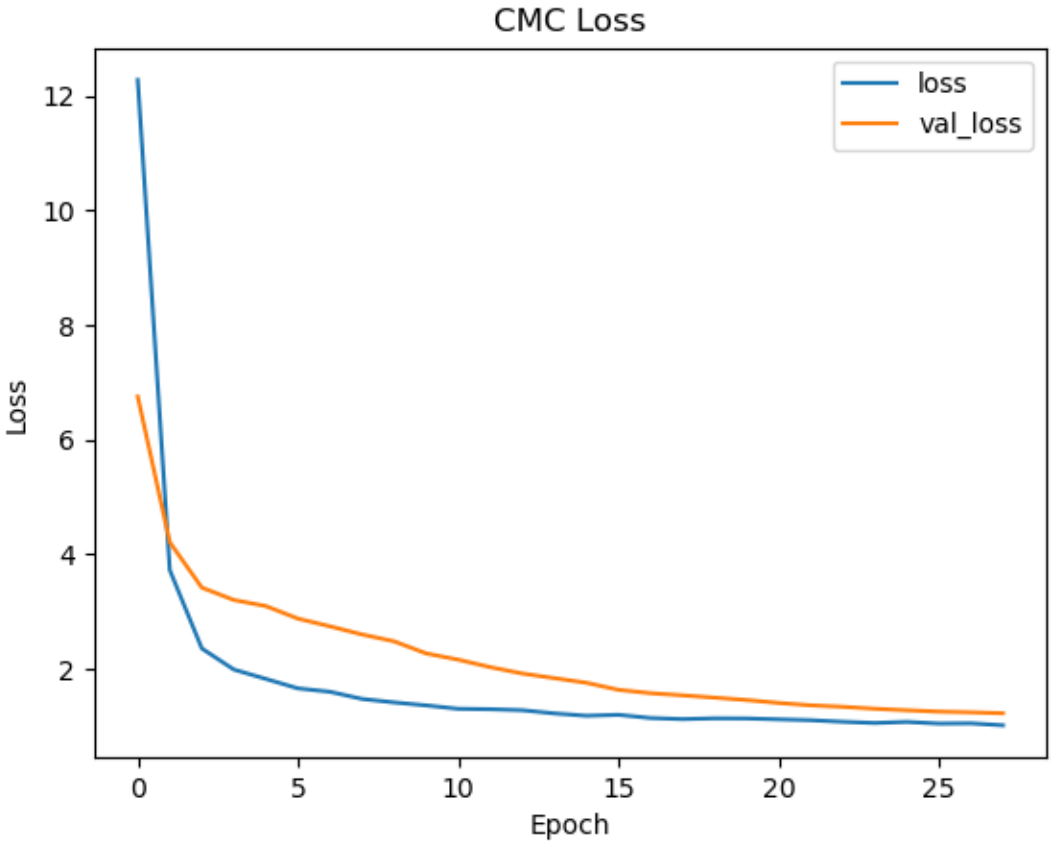
The ranked indices of the results of all 200 trials for the two models were plotted against each other, and interestingly had no correlation (*data not shown*). This was initially surprising because I expected that similar model architectures would be effective for both tasks since they would identify important features of the data. However, it turns out that many model architectures achieved very similar MSE and F1 scores, so it didn't matter too much which architecture was chosen.

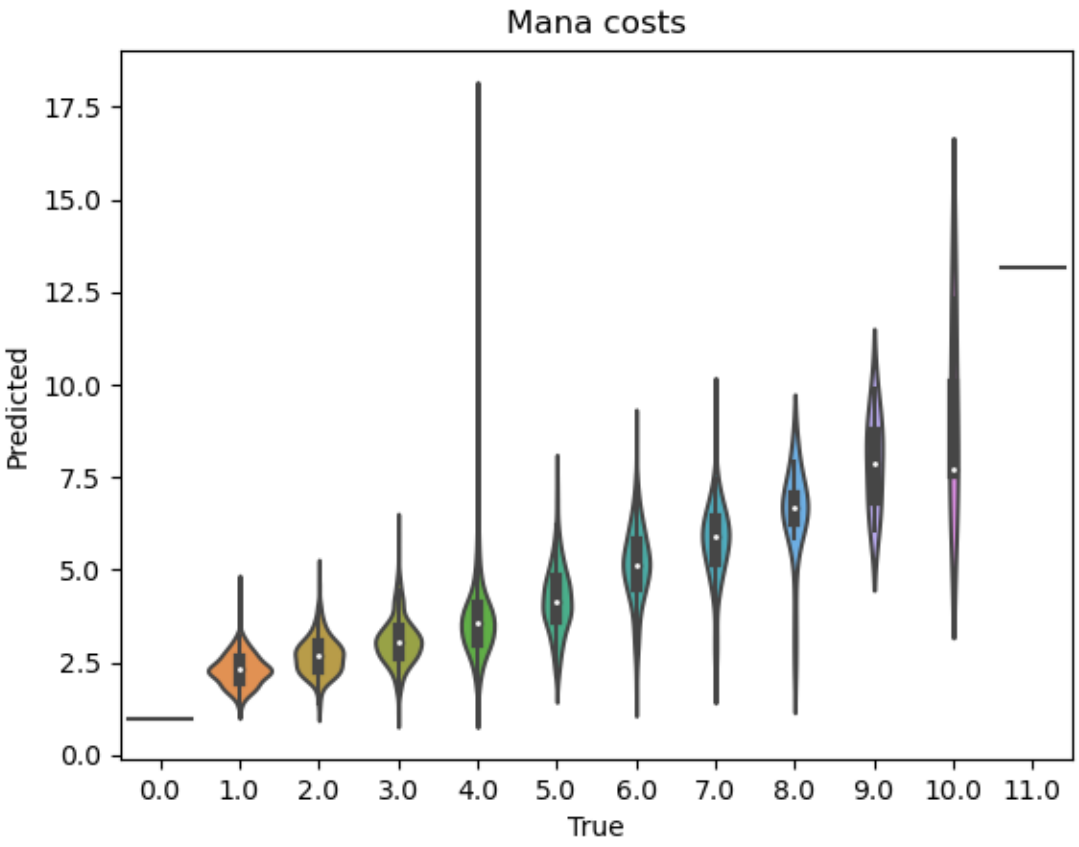
In particular, the best CMC model achieved a color F1 score within 1% of the best color model, while the best color model achieved a CMC MSE that was 10% worse than the best CMC model. Therefore the hyperparameters of the best CMC model were chosen for both final non-NLP CMC and color models. Overall I was impressed by the accuracy of this model, as expressed through the mean squared error in the low 1 range and an F1 score of around 0.7. The fact that the model accurately predicted the CMC and color of individual cards and of novel user-entered cards was also impressive.

The NLP model alone did not achieve good accuracy on either metric (mean squared CMC error around 3, and color binary crossentropy loss of 0.55). I attribute this to my inexperience with NLP model architectures, to the small data set, and to the fact that there is a high Bayes error to this task as a lot of important information is absent.

The hybrid structured-data and NLP model would, I expect, have been a significant improvement over the vanilla model, but I was not able to get it running to test it.




CMC Loss
loss
val_loss
Loss
Epoch
12
10
8
6
4
2
0
5
10
15
20
25



CMC model mean squared error is 1.1871771685726007

White:

	precision	recall	f1-score	support
0	0.90	0.95	0.92	1562
1	0.80	0.67	0.73	517
accuracy			0.88	2079
macro avg	0.85	0.81	0.82	2079
weighted avg	0.87	0.88	0.87	2079

Blue:

	precision	recall	f1-score	support
0	0.89	0.96	0.92	1603
1	0.80	0.60	0.69	476
accuracy			0.87	2079
macro avg	0.85	0.78	0.80	2079
weighted avg	0.87	0.87	0.87	2079

Black:

	precision	recall	f1-score	support
0	0.88	0.96	0.92	1575
1	0.82	0.58	0.68	504
accuracy			0.87	2079
macro avg	0.85	0.77	0.80	2079
weighted avg	0.86	0.87	0.86	2079

Red:

	precision	recall	f1-score	support
0	0.88	0.96	0.91	1586
1	0.80	0.57	0.67	493
accuracy			0.86	2079
macro avg	0.84	0.76	0.79	2079
weighted avg	0.86	0.86	0.86	2079

Green:

	precision	recall	f1-score	support
0	0.89	0.94	0.91	1555
1	0.79	0.65	0.71	524
accuracy			0.87	2079
macro avg	0.84	0.79	0.81	2079
weighted avg	0.86	0.87	0.86	2079

Ruric Thar, the Unbowed is predicted to have CMC [[5.6629725]], and its true CMC is [6.0]

['Ruric Thar, the Unbowed is predicted to have color identity RG, and its true color identity is G,R']

My creature is predicted to have CMC [[3.0435607]]

My creature is predicted to have color identity U

Started at: 2021-05-31 22:07:54.147513

Ended at: 2021-05-31 22:08:08.728071