Convolutional Neural Networks for Color Classification of Magic: The Gathering Cards

Andrew Boram
Shippensburg University
ab8102@engr.ship.edu
Mentor: Dr. Girard

I. ABSTRACT

This paper details an experiment regarding the use of convolutional neural networks for classifying *Magic: The Gathering* cards by their Color. This article describes the main problem description, followed by the background information needed for the project (*Magic: The Gathering* basics, neural networks, and convolutional neural networks), the solution description, experiment design, and finally the results of the experiment. The experiment compares a neural network trained with color images against one trained with grayscale images. After running the experiment, reducing results, and performing statistical analysis, it can be concluded that the color neural network had a significant advantage.

II. INTRODUCTION

Magic: The Gathering (or *Magic*, for short) is a collectable trading card game invented in 1993 by Richard Garfield, and published by Wizards of the Coast. The gameplay of *Magic* involves great amounts of strategy, with plenty of complex mechanics mixed in. The level of complexity and in-depth mechanics grants many opportunities for experimentation, and is particularly attractive in the area of neural networks and machine learning. This project explores the (arguably) most important mechanic in the game, Color. The main goal of this project is to see if the color of a *Magic* card can be identified based on the artwork.

III. GAMEPLAY AND MECHANICS

Magic has two key card types, lands and spells. A land card can be played once per turn on the battlefield, and can be activated to gain *Magic*'s main resource: mana. This activation is known as "tapping", and is denoted physically by turning the card sideways on the table. At the start of each player's turn, they "untap" all cards on the battlefield under their control by turning the card back upright (denoting that they can now be tapped again). Lands also fall under the category of a "permanent", as the card itself is physically placed on the battlefield [7].

A. Mana and Basic Lands

The most common type of lands are the basic lands. The five different basic lands are Plains, Island, Swamp, Mountain, and Forest [2]. Each basic land type, when tapped, gives one of the five different colors of mana in the game. Plains provide white mana, Islands provide blue mana, Swamps provide black mana, Mountains provide red mana, and Forests provide green mana [9]. After playing enough lands, a player can tap them for mana, and use that mana to pay for spells. Most spells have specific requirements for what color of mana can be used to cast them. If a spell requires one black mana, a player cannot tap a Forest and use one green mana to cast that spell. This is the core ideology of *Magic* [3].

B. Spells

Spells are an umbrella term referring to card types that aren't lands. These types are creatures, artifacts, enchantments, planeswalkers, instants, and sorceries [12]. Creatures, artifacts, enchantments, and planeswalkers are also permanents just like lands. After casting one, a player will set it on the battlefield to denote that it is in play [12]. Instants and sorceries carry out one function and are not represented on the battlefield. As instants and sorceries are not permanents, their cards are placed in a different zone known as the graveyard after they are cast and used. Permanents can also be sent to the graveyard if a spell or other ability removes it from the battlefield. Non-land permanents such as creatures, artifacts, or enchantments can also have abilities just like lands, but they typically do not provide mana, but rather another useful function.

IV. COLOR

As mentioned previously, Color in *Magic: The Gathering* is arguably the most important aspect of the game, and definitely the most important aspect of this project. Color is not simply just the literal color of the card. In *Magic: The Gathering* terms, color has two definitions in the contexts of flavor and mechanics [3].

A. Color in Gameplay

At a high level view, the Color of a card is determined by the type of mana that is required to cast it [3]. Since Lightning Bolt (see Figure 1) requires one red mana, it is a red spell and cannot be cast with any other color of mana. There are some notable exceptions to this rule as a card's color may also be defined by a color indicator on the card, or an ability that states the card's color [3]. A card can also have multiple colors, including all five colors. Cards with more than one color are known as "multicolored", while cards with one color are known as "monocolored" [3]. For the purposes of this project, the main focus will be determined on differentiating between monocolored cards.

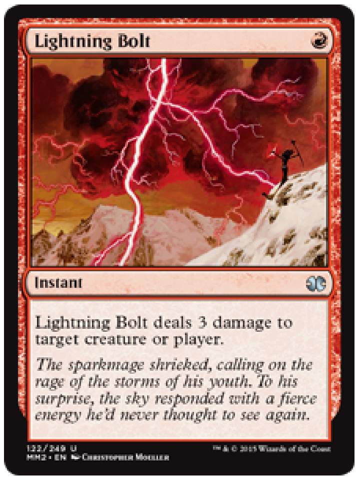

Lightning Bolt
Instant
Lightning Bolt deals 3 damage to target creature or player.
The sparkmage shrieked, calling on the rage of the storms of his youth. To his surprise, the sky responded with a fierce energy he'd never thought to see again.
122/249 U
MM2 • EN Christopher Moeller
™ & © 2015 Wizards of the Coast

Figure 1: the red card Lightning Bolt [8]

The reasoning for having multiple colors is simply balance. Having one color that is more powerful than the others would set off the entire balance of the game. Color gives the game diversity in its cards, effects, and play styles, while preventing any one deck from having every tool in the game [3]. Each Color has common abilities and mechanics that may not appear at all on a card of a different Color.

While most cards have a color, some do not have a color at all. These cards are referred to as "colorless". Colorless cards do not count as a sixth color, they are simply the absence of all colors [3]. Since lands do not have a mana cost, and are "played" not "cast", all lands are colorless even if it taps for a color of mana. Just like an ability can specify the color of a card, an ability can specify if a card is colorless.

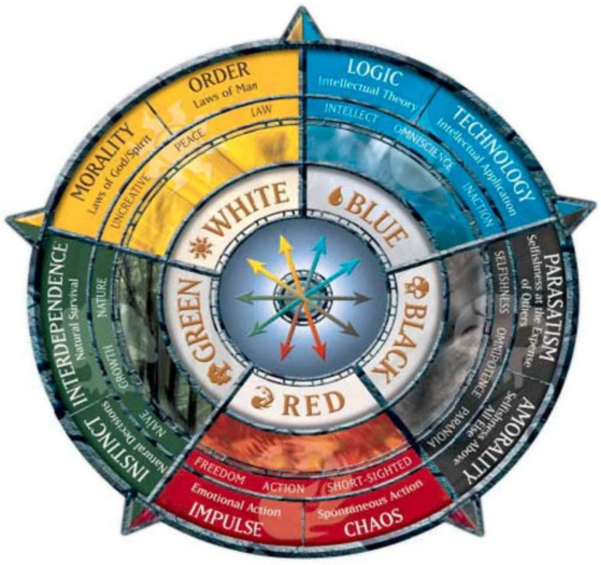


Figure 2: "The Color Wheel", a common representation of all Colors and their themes [3].

B. Color in Flavor

Each color in *Magic* also has various themes that are frequently represented in the card art. For example, it's not uncommon to see a skeleton on a black card, an angel on a white card, or a giant elephant in the forest on a green card.

In flavor and lore, each Color has its own core concepts and ideologies that shape card art and mechanics. White cards for example represent peace, law, structure, selflessness and equality. A stark contrast when compared to black cards that represent power, self-interest, death, sacrifice, and uninhibitedness [3].

V. NEURAL NETWORKS

The design of a neural network is inspired by the human brain: neurons linked together in a network. These neurons can be visualized as a function that takes in output from previous sources, uses those values in some computation, and produces some new output to be passed along to the next neuron.

A. Neurons

Neurons receive input from either an external source, or previously from another neuron in the network. Neurons form an interconnected network separated into layers that each have their own purpose. There are three broad categories of layers: input, hidden, and output. Neurons in the input layer take information from the outside world, and no computation is performed. Neurons in the hidden layer are "hidden" from the outside world, and perform a specified computation and pass the information along to the next layer. Neurons in the output layer are responsible for computations just like the hidden layer(s), but the information is then transferred back to the outside world [13].

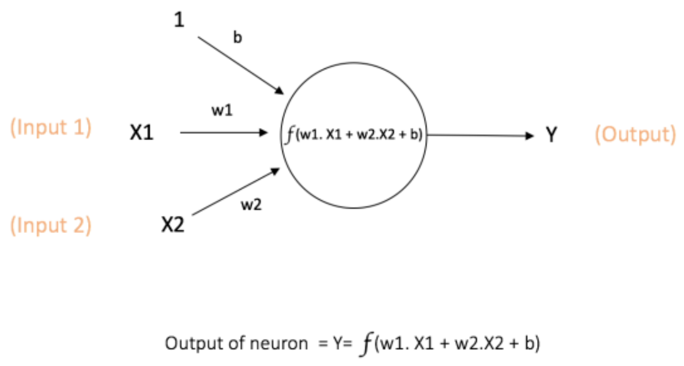


Figure 3: A single neuron [13]

B. Weight and Bias

Each neuron input has an associated "weight", which is assigned on the basis of its relative importance to other inputs [13]. When the computation of a neuron is calculated using information provided to it by previous neurons, the weight of each is first taken into account. If the provided information is from a low importance neuron, the value will not have a large effect on the receiving neuron's computation as it is not as important compared to another node with a higher weight. Each computation can also have a bias applied to it. Biases and weights do not stay the same throughout the runtime of the neural network. They are both trainable values that can be adjusted to obtain correct output [13]. At the start of the neural network, before any training, weights and biases are randomized.

C. Activation Functions

Another important aspect of neural networks is the "activation function", which is the pre-selected computation that will run at each neuron. For the output at each neuron, the weighted sum of data received from previous neurons is calculated, the bias is added, and the result is fed into the specified activation function to perform its fixed mathematical operation.

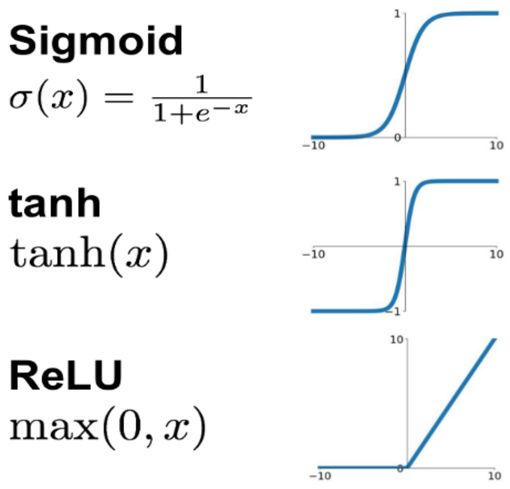


Figure 4: Common activation functions [11]

Activation functions are specifically used to introduce non-linearity to the output of a neuron [13]. This is needed in a neural network as outputs of neurons could vary wildly, and the activation function will be able to scale down this output into a much simpler range to make training easier. While there are several activation functions to choose from, the best performing in modern neural networks is the ReLU (Rectified Linear Unit) function [4]. The ReLU function simply makes all negative values zero. This is a biologically inspired function, since neurons in a brain will either “fire” (return a positive value) or not (return zero) [6].

D. Forward Propagation

This entire process of calculating and feeding data forward is known as "forward propagation". All neurons in one layer will receive input from one or more neurons in the previous layer, calculate the output, and pass it along to the next layer of neurons. Receiving neurons in the next layer will continue computation with the data received. This process continues until the output layer is reached and completed.

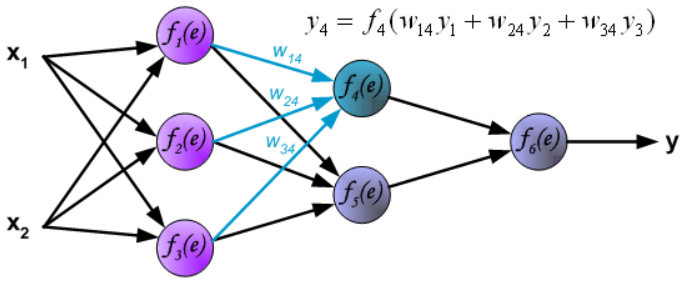


Figure 5: Forward propagation in action [13]

E. Backpropagation

The process of actually "training" or "learning" in a neural network is known as "backpropagation". Since the weight and bias values are completely random at the start, the actual values for the first pass will also be completely random. The received values are then used in what is known as a "loss function" such as MSE (Mean Squared Error) [1]. This function compares the received (random) values against what the correct result should be. If the received value is close to the correct value, the result of the loss function will be close to zero. The result of this loss function after the first pass is likely very high, as random data should be wildly inaccurate.

To minimize this cost, the weight of the previous neurons need to be adjusted, the weights feeding into those neurons need to be adjusted, and so on. This is a recursive process used to determine how sensitive the cost function is to small changes in each weight. A small change in a given weight, will cause a change in the weighted sum at that neuron, which will change the result of the activation function, which will then finally change the result of the cost function [4]. The new weight can then be determined by taking the derivative of the cost function with respect to weight, multiplying it by a predetermined "learning rate", and subtracting it by the initial weight of the neuron [1].

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

Figure 6: Derivative of the cost function with respect to the weight [4]

This process is done at each neuron, backwards through the neural network, slightly adjusting weights as needed. The learning rate is (typically) a small number used to slightly adjust the weights over each pass. If the learning rate is too high, then jumps in adjusting the weights are larger which can affect accuracy [1].

VI. CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNN) are a subcategory of neural networks that excel in areas of image recognition and classification. They get their name from the unique convolution layer, used to extract features from an input image [14]. Other layer operations include pooling and classification.

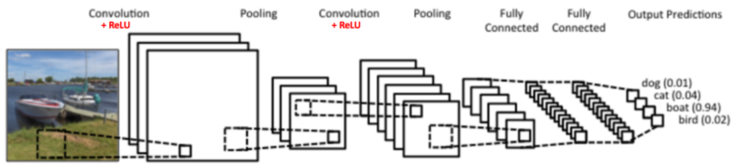


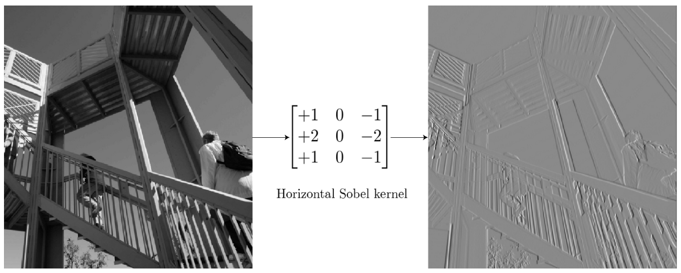
Figure 7: An example convolutional neural network [14]

A. Image Input

Images in a CNN are represented as a matrix of pixel values, with the size being the size of the source image in pixels. A standard color image will have three channels (red, green, and blue) each getting their own matrix. Color images are essentially three two-dimensional matrices stacked over each other, with each value ranging from zero to 255 representing how much of that color resides in the given pixel [13]. Grayscale images only need one matrix, where zero is black and 255 is white. As the inputs are matrices, the hidden layers in the neural network will be specific linear algebra operations.

B. Convolutional Layer

Convolution is a key element of a CNN, and it is always the first layer after the input layer. The convolution operation involves taking a kernel (or filter), which is simply a smaller matrix of weights, and "sliding" it over the input matrix. The kernel goes from column to column and row to row performing an element-wise multiplication with the part of the input matrix it is currently on, and then summing up the results into a single output pixel [5]. The "stride" value can also be adjusted to have the kernel move in larger steps across the matrix, rather than just by one. In the case of a color image, a kernel will have three channels for each color, and a convolution operation is done for each layer. Different kernels are used in a CNN to detect certain identifying features like lines or curves. A CNN learns the values of these filters on its own during the training process by learning what features to recognize [14].



$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$
Horizontal Sobel kernel

Figure 8: Example of a filter (kernel) operation [5]

A common practice for convolution is "zero-padding", where the input matrix is padded around the outside with zeroes. This is useful if elements on the edge of images are being lost in the convolution operation[14]. After each convolution operation a non-linearity function is performed, such as ReLU, on each pixel value. In this context, ReLU is still needed to introduce non-linearity to the neural network, since most of the real-world data the CNN learns would be non-linear [14].

C. Pooling Layer

The purpose of the pooling layer is to essentially reduce the size of the input. This process involves taking a smaller window of the matrix, similar to how a kernel slides over it in the convolution process, and obtaining just one value of that smaller window for the pooled matrix. In the case of Max Pooling, the largest value of the region selected by the window is chosen for the new matrix and other values are ignored and lost. Alternative methods exist such as taking the average or taking the sum (instead of just selecting the maximum value). Pooling allows the input to be smaller and therefore more manageable as a whole[14].

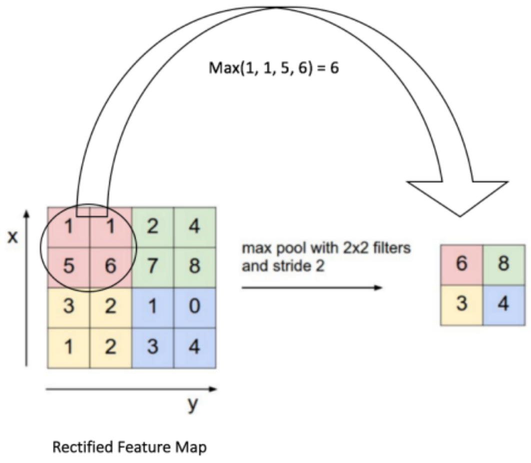


Figure 9: Max Pooling [14]

D. Fully Connected Layer

The fully connected layer is the last set of layers in a CNN after all of the convolution and pooling. The convolution and pooling layers are essentially feature extractors from the image, while the fully connected layer works on classification. The entire purpose of this layer is to use the provided features from the previous layers to classify the input image into the various classes based on the dataset [14]. For example, if the inputs were images of animals, some of the classes could be cat, dog, bird, etc. The sum of output probabilities from the fully connected layer is one. This is ensured by using the Softmax as the activation function in the output layer of the fully connected layer. The Softmax function takes the values and "squashes" them down between zero and one (that all sum to one) [14]. Through forward and back propagation, the fully connected layer learns and improves to provide correct output for the CNN.

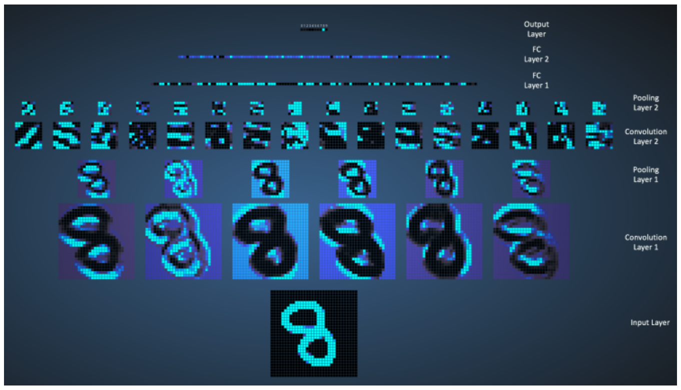


Figure 10: Example CNN processing a handwritten digit [14]

VII. PRIMARY OBJECTIVE

To test how using color images versus grayscale images affects classification accuracy of a convolutional neural network on Magic: The Gathering card art.

VIII. SOLUTION DESCRIPTION

The experiment implemented multiple convolutional neural networks in Python using the TensorFlow and Keras libraries. Each of the two groups, grayscale and color, had 10 CNNs. There were 5,000 card art images total, separated into ten groups of 500, which is 100 images for each color. Each CNN trained on nine of the groups and then evaluated on the remaining group. The groups then "rotated" until every group had been the evaluation set. Each group set ran ten times, providing a total of 100 CNNs for both color and grayscale. The kernels used were well-known edge-detection operators (Sobel, Prewitt, Scharr, and Kirsch operators).

The data gathered for this project was obtained through a public *Magic: The Gathering* database known as Scryfall. Scryfall offers high quality scans of *Magic* cards, along with a separate section for "art crops" which is only the art on the card [10]. Using a Python wrapper for this API, art crops in bulk can be downloaded with ease. Scryfall has options for filtering out specific cards based on certain criteria, and that was used to make sure only high quality scans were received.

IX. GOAL TREE AND HYPOTHESIS

Hypothesis: The CNN trained on color images will be more accurate than the CNN trained on black and white images.

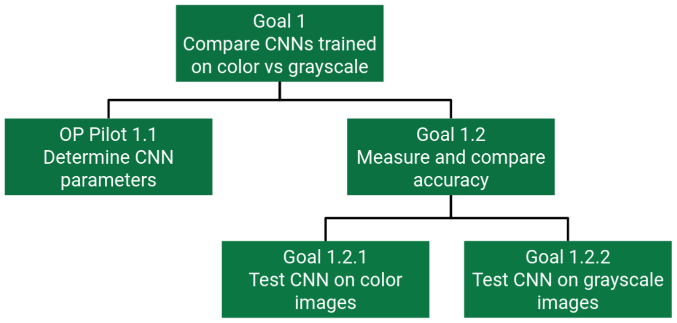

Goal 1
Compare CNNs trained on color vs grayscale
OP Pilot 1.1
Determine CNN parameters
Goal 1.2
Measure and compare accuracy
Goal 1.2.1
Test CNN on color images
Goal 1.2.2
Test CNN on grayscale images

Figure 11: Goal Tree

X. EXPERIMENT DESIGN

Factors:	Value:
Images	Color, Grayscale
Representation Factors	Random numbers in kernels, weights, and biases.

Block Design		CNN
Image Type	Color	✓
	Grayscale	✓

XII. RESULTS

The experiment result provided 100 unique data points for both the color and grayscale neural networks to be used for comparison. On average, the results from the color set had an accuracy of 44.27%, while the grayscale results only had an average accuracy of 27.89%.

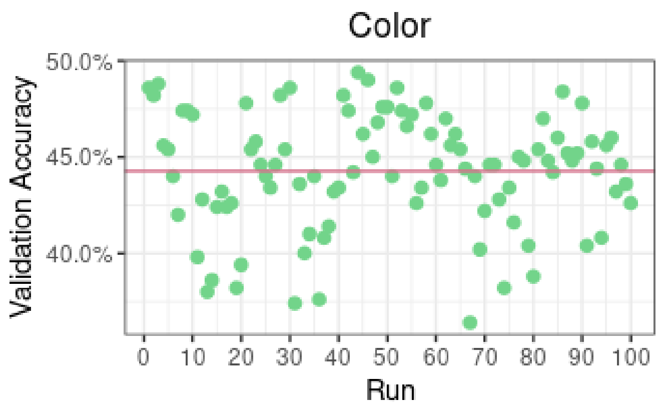


Figure 12: Color CNN Results

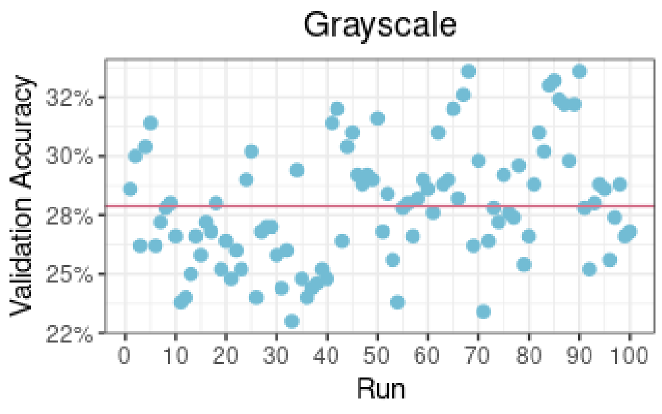


Figure 13: Grayscale CNN Results

As mentioned before, each of the separate group sets were trained and tested 10 times. In both figures there is a good bit of variability as expected. Three different t-tests are used on the data to test statistical significance. Each sample was compared using a one sample t-test against the baseline of 20% accuracy. 20% was chosen as there are 5 possible outcomes of the neural network. If chosen randomly, one would expect about 20% accuracy. A paired t-test was also used to compare both of the samples. All of the t-tests provided a large t-value and an extremely small p-value.

One Sample t-test		
Experiment	t-value	p-value
Color	80.783	< 2.2e-16
Grayscale	30.892	< 2.2e-16

Paired t-test	
t-value	p-value
52.041	< 2.2e-16

Figure 14: Statistical Analysis

Typically, a p-value less than 0.05 determines significance, and the p-value for all t-tests is well below that. This tells us that each of the datasets alone are significant compared against 20% accuracy, and each of the samples are also statistically different.

XIV. CONCLUSION

Based on the results in figures 12 through 14, it can be determined that the hypothesis is correct. Color images give the CNNs a significant advantage that grayscale images do not. Interestingly, grayscale is still better than random guessing (20%) as confirmed by the one sample t-test. Simply relying on edge detection methods doesn't appear to be enough when classifying images with a convolutional neural network.

XV. FUTURE WORK

There are potentially an unlimited number of options that could be carried on from this experiment (in terms of improving this project in particular, or otherwise). There could be a better CNN configuration that wasn't utilized here that could improve accuracy or even disprove the hypothesis. Perhaps it would also be beneficial to limit the testing set to only a few sets, to give the accuracy a better chance. It would be interesting to see if a CNN could also be used to classify the

Magic card type (Land, Creature, Instant, etc.) or some other mechanic instead.

REFERENCES

1. A. Deshpande, “A Beginner's Guide To Understanding Convolutional Neural Networks,” A Beginner's Guide To Understanding Convolutional Neural Networks – Adit Deshpande – Engineering at Forward | UCLA CS '19, 10-Jul-2016. [Online].Available: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed: 10-Apr-2022].

2. “Basic land,” MTG Wiki. [Online]. Available: https://mtg.fandom.com/wiki/Basic_land. [Accessed: 10-Apr-2022].

3. “Color,” *MTG Wiki*. [Online]. Available: https://mtg.fandom.com/wiki/Color. [Accessed: 10-Apr-2022].

4. G. Sanderson, “Neural Networks - The basics of neural networks, and the math behind how they learn,” *3Blue1Brown*, 2017. [Online]. Available: https://www.3blue1brown.com/topics/neural-networks. [Accessed: 10-Apr-2022].

5. I. Shafkat, “Intuitively Understanding Convolutions for Deep Learning,” *Medium*, 07-Jun-2018. [Online]. Available: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1. [Accessed: 10-Apr-2022].

6. L. Strika, “Why do Neural Networks Need an Activation Function?,” *Medium*, 02-Jul-2019. [Online]. Available: https://towardsdatascience.com/why-do-neural-networks-need-an-activation-function-3a5f6a5f00a. [Accessed: 10-Apr-2022].

7. “Land,” *MTG Wiki*. [Online]. Available: https://mtg.fandom.com/wiki/Land. [Accessed: 10-Apr-2022].

8. “Lightning Bolt,” *Gatherer*. [Online]. Available: https://gatherer.wizards.com/pages/card/Details.aspx?multiverseid=397722. [Accessed: 10-Apr-2022].

9. “Mana,” *MTG Wiki*. [Online]. Available: https://mtg.fandom.com/wiki/Mana. [Accessed: 10-Apr-2022].

10. “REST API Documentation,” *Scryfall Magic: The Gathering Search*. [Online]. Available: https://scryfall.com/docs/api. [Accessed: 10-Apr-2022].

11. S. Jadon, “Introduction to Different Activation Functions for Deep Learning,” Medium, 03-Feb-2022. [Online]. Available: https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092. [Accessed: 07-May-2022].

12. “Spell,” *MTG Wiki*. [Online]. Available: https://mtg.fandom.com/wiki/Spell. [Accessed: 10-Apr-2022].

13. U. Karn, “A quick introduction to Neural Networks,” *the data science blog*, 10-Aug-2016. [Online]. Available: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/. [Accessed: 10-Apr-2022].

14. U. Karn, “An Intuitive Explanation of Convolutional Neural Networks,” *the data science blog*, 29-May-2017. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: 10-Apr-2022].