INVESTIGATING SEMANTIC PROPERTIES OF IMAGES GENERATED FROM NATURAL LANGUAGE USING NEURAL NETWORKS

by

Samuel Ward Schrader

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Samuel Ward Schrader

Thesis Title: Investigating Semantic Properties of Images Generated from Natural Language using Neural Networks

Date of Final Oral Examination: 8th March 2019

The following individuals read and discussed the thesis submitted by student Samuel Ward Schrader, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Dr. Bogdan Dit, Ph.D.	Chair, Supervisory Committee
Dr. Casey Kennington, Ph.D.	Member, Supervisory Committee
Dr. Francesca Spezzano, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Dr. Bogdan Dit, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

ACKNOWLEDGMENTS

After a career in education, I came to Boise State to expand my education in computer science and develop the skills needed for a second career. I have been incredibly lucky to be able to pursue my passions and interests professionally, and Boise State choosing to fund my education and research allowed me to once again follow this path and not be a drain on my family. I am incredibly grateful for the opportunity.

I am grateful to Dr. Bogdan Dit for taking me as his graduate assistant. I was consistently humbled by his kindness, expertise, and support in helping me grow as a researcher and allowing me to investigate areas of high interest to us both, but which were not well worn or established areas of expertise for either of us.

I am grateful to Dr. Casey Kennington for his role as co-advisor, professor, and mentor. His instruction, advice, and friendly words, were always extraordinarily helpful. I have tremendous room to grow, but it must be said that I am a much better student and researcher because of his influence.

I would also like to thank Dr. Spezzano for serving on my committee, and the Computer Science Department administration for all they do. Susie Gillikan and the other members of the CS administrative staff do an incredible job at making the department welcoming, productive, and positive.

Finally, I am grateful to my family for always supporting me in my education and professional pursuits. I could not have made it here without them.

ABSTRACT

This work explores the attributes, properties, and potential uses of generative neural networks within the realm of encoding semantics. It works toward answering the questions of: If one uses generative neural networks to create a picture based on natural language, does the resultant picture encode the text's semantics in a way a computer system can process? Could such a system be more precise than current solutions at detecting, measuring, or comparing semantic properties of generated images, and thus their source text, or their source semantics?

This work is undertaken in the hope that detecting previously unknown properties, or better understanding them, could lead to new or improved methods of encoding and processing semantics in a computer system. Improvements in this space could affect many systems that make semantically based decisions. Being able to detect general or specific semantic properties, semantic similarity, or other semantic properties more effectively could improve tasks such as information retrieval, question answering, duplication (clone) detection, sentiment analysis, and others. Additionally, it could provide insight into how to better represent semantics in computer systems and thus bring us closer to general artificial intelligence.

To explore this space, this work starts with an experiment consisting of transforming pairs of texts into pairs of images via a generative neural network and exploring properties of those image pairs. The text pairs were known to either be textually and semantically identical, semantically similar, or semantically dissimilar. The resultant image pairs are then tested for similarity via a second neural network based process to

investigate if the semantic similarity is preserved during the transformation process and thus, exists in the resultant image pairs in a quantifiable way.

Preliminary results showed strong evidence of resultant images encoding semantics in a measurable way. However, when the experiment was conducted on a larger dataset, and with the generative network more thoroughly trained, the results are weaker. An alternative experiment conducted on different datasets and configurations produced results that are still weaker than the preliminary experiments. These findings lead us to believe the promise of the preliminary results was possibly due to semantics being encoded by the vectorization of the words, and not by the generative neural network. This explanation seeks to clarify why, as the generative neural network took a larger role in the process, the results were worse, and as it took a smaller role, the results were better. Further tests were conducted to establish this belief and proved supportive.

TABLE OF CONTENTS

ABSTRACT v

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ABBREVIATIONS xiv

1 Introduction 1

1.1 Contributions 4

1.2 Thesis Organization 4

2 Background 6

2.1 Artificial Intelligence 6

2.2 Neural networks 7

2.3 Generative neural networks 9

2.4 Information Retrieval 10

2.5 Natural Language Processing 11

2.6 Corpus 11

2.7 Preprocessing 11

2.8 Glove 12

3 Related Work 13

4 Approach 16

5 Experiments 19

5.1 Preliminary Experiments 19

5.2 Primary Experiments 26

5.2.1 Model 1: Flowers Dataset 27

5.2.2 Model 2: Coco Dataset 32

5.2.3 Discussion of Results 41

5.3 Investigation of Glove's impact 43

6 Limitations and Threats to Validity 48

7 Conclusions 50

8 Future Work 53

REFERENCES 55

A Primary Experiments Technical Details 59

LIST OF TABLES

5.1 Summary of the results of the preliminary experiment 24
5.2 Results from the model with the GNN trained on the flowers dataset at 1,000 epochs 31
5.3 Results from the model with the GNN trained on the Coco dataset at 100 epochs 36
5.4 Results from the model with the GNN trained on the Coco dataset at 10 epochs 40
5.5 Summary of the results of when comparing Glove (baseline) and Cerberus on the Bug Duplicate dataset 44

LIST OF FIGURES

2.1 An illustration of the basic layout of a simple neural network 9

3.1 A visualization of the methodology used by Microsoft's CNTK for measuring image similarity. Images input into the tool are output as 512 point matrices that can be compared 15

4.1 An outline of the novel model used in this work. We call the model Cerberus due to its use of three neural networks. Text is input into a GNN, which outputs an image. That image is then fed into the Microsoft CNTK, which outputs a 512 point matrix that represents visual aspects of the image. 17

4.2 An detailed overview of the Cerberus methodology. Bugs are preprocessed, tagged, turned into images, then evaluated 18

5.1 An outline of Cerberus with pairs 21

5.2 An illustration of the question that will be used to measure if semantics are encoded into images by GNNs in a measurable way: Would the similarity of resultant matrices correlate with the text similarity? 22

5.3 A pair of images generated from identical bug reports from the preliminary dataset. They are used to confirm that the model is producing images in a consistent manner. The resultant matrices have a Euclidean distance of 0. 23

5.4 A pair of images generated from semantically similar (duplicate) bug reports from the preliminary dataset. The resultant matrices have an average Euclidean distance of 21. 23
5.5 A pair of images generated from semantically dissimilar (non-duplicate) bug reports from the preliminary dataset. The resultant matrices have an average Euclidean distance of 25. 23
5.6 An image produced from a bug report input into the GNN after it had been trained on the flowers dataset at 1,000 epochs 28
5.7 Once again used as a sanity check, this image shows that identical texts produced identical images when fed to the GNN trained on the flowers dataset at 1,000 epochs 28
5.8 Images produced from 5 pairs of duplicate bugs transformed by the GNN trained on the flowers dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 28
5.9 Images produced from 5 pairs of non-duplicate bugs transformed by the GNN trained on the flowers dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 28
5.10 An image produced when 10 sentences describing a flower were input into the GNN when trained on the flowers dataset at 1000 epochs 29
5.11 An image produced when 1 sentences describing a flower was input into the GNN when trained on the flowers dataset at 1,000 epochs 29
5.12 The image produced when a blank text file was input into the GNN when trained on the flowers dataset at 1,000 epochs 30

5.13 An image produced when 1,035 words from a random website was input into the GNN when trained on the flowers dataset at 1,000 epochs 30
5.14 Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 34
5.15 Images produced by a text file containing the single word "red" and a blank text file transformed by the GNN trained on the Coco dataset at 1,000 epochs 35
5.16 Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 100 epochs. All images are similar to each other but are generated from distinctive texts 36
5.17 Images produced from five pairs of duplicate bugs transformed by the GNN trained on the Coco dataset at 100 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 36
5.18 Images produced from five pairs of non-duplicate bugs transformed by the GNN trained on the Coco dataset at 100 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 36
5.19 Images produced by a text file containing the single word "red" (left) and a blank text file (right) transformed by the GNN trained on the Coco dataset at 100 epochs 37

5.20 Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 10 epochs. All images are similar to each other but are generated from distinctive texts 38
5.21 An in-depth view of three images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 10 epochs 38
5.22 Images produced by a text file containing the single word "red" (left) and a blank text (right) file transformed by the GNN trained on the Coco dataset at 10 epochs 39
5.23 Images produced from five pairs of duplicate bugs transformed by the GNN trained on the Coco dataset at 10 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 39
5.24 Images produced from 5 pairs of non-duplicate bugs transformed by the GNN trained on the Coco dataset at 10 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on. 40
5.25 Precision (y-axis) for Glove and Cerberus across different thresholds (x-axis), using the Euclidean distance 46
5.26 Precision (y-axis) for Glove and Cerberus across different thresholds (x-axis), using the cosine similarity 46

LIST OF ABBREVIATIONS

GNN – Generative Neural Network

NLP – Natural Language Processing

IR – Information Retrieval

AI – Artificial Intelligence

CHAPTER 1

INTRODUCTION

Understanding, organizing, searching, comparing or otherwise processing semantics (i.e., meaning) from human input is a challenging and ongoing problem in Natural Language Processing (NLP) [16], Information Retrieval (IR) [32], Sentiment Analysis (SA) [23], and many other Artificial Intelligence (AI) related tasks [31]. The primary manner in which humans communicate semantics is via language, which is most easily represented in a computer system as text. It is easy for a human to understand and derive meaning from text, but encoded representations of text in a computer system are binary representations of letters, spaces, and characters which are in and of themselves semantically meaningless. When semantics are encoded as binary text representations, the only similarity that is easy for a computer system to perceive are those that result in similar or identical binary strings.

The choice to encode semantic information in this way has significant consequences, as the decision-making capacity of an artificial intelligence system is limited by the methodology used to store and process the system's data. An artificially intelligent system can only make decisions about things that it can perceive and process. If it can only process natural language as binary representations of text, it can only make distinctions at that level of abstraction. To address this issue, most current approaches involve processing binary strings of text at the word level (removing

special characters, tokenizing, stemming, removing stop words, etc.), followed by organizing the text as individual words, bi-grams, tri-grams, or n-grams, and bagging the results. Each word in a corpus (i.e., a collection of documents, or "bag"), can also be vectorized based on the probabilities of its surrounding words, in an attempt to encapsulate and capture the context of those words. This level of abstraction allows a computer to make decisions and comparisons that are more similar to those of humans evaluating language. Operating at this level of abstraction has proven useful and effective as vectorized, mapped, or otherwise manipulated text (e.g., using term frequency-inverse document frequency, cosine similarity, word-to-vec distance, etc. [19, 20]) can then be compared based on similarity, or otherwise processed in a manner that seeks to make decisions about the semantics of the text involved, based on the properties that are quantifiable at that level of abstraction.

However, by closely examining the process or exploring the limits of performance, it becomes apparent that significant semantic information is lost during these processes, and AI systems are not yet close to "understanding" the semantics being processed. In order to achieve a general AI that can perceive information in a general manner and make generally intelligent decisions, the system must be able to store and process semantics in a general way, and significant progress on the problem of semantics still needs to be made [21].

It is thus worthwhile to explore other methods for encoding and processing semantics in pursuit of methodologies that could better support general intelligence, and empower systems to make more semantically sound decisions. Better methods, or increased understanding that could lead to improving existing methods, could improve fields such as software engineering (e.g., duplicate bug report identification [34] – as shown in the preliminary experiments (see Chapter 5.1), source code clone detec-

tion [29], establishing traceability between source code and documentation [2]), recommender systems [28] (e.g., question identification for questions answering, similar profile recognition, etc.), or any system that needs to make decisions based on semantic similarities or semantic properties of language. Just as the methods of semantic encoding using word vectorization have been far-reaching, if a better method for encoding and processing semantics is found, the impacts could be significant [26, 34]. Additionally, an improved general approach to representing semantics in a computer system could make progress toward a machine being able to better "understand" concepts and ideas that humans experience more feasible. The distance between what humans want to communicate with a machine about, or want a machine to make decisions about, and the data that a machine is making decisions with, would be reduced. This would likely improve processes and implementations rooted in ideas or concepts that could be better represented in this new technology and not a more limited textual representation of that ideal.

One unexplored methodology for encoding and processing semantics is to use a generative neural network to generate images based on text and then use image processing tools to measure and compare the properties of the resultant images. If generated images store semantics in a manner that is robustly searchable, quantifiable, and processable by machine decision making, it is possible that such a process could be more effective in some areas, or hold some advantage over current methodologies in some areas.

Potential applications of an effective model are extensive. As the preliminary experiments will illustrate (see Chapter 5.1), bugs reports could be represented as images and similar images could identify bugs that are semantically similar and therefore likely to be duplicates (previously reported). Complex combinations of

information could be represented as images, like purchaser profiles in financial data. Such images could then be used to identify profiles that are likely to commit fraud, or profiles that are likely to be a good investment, or a good target for marketing. Almost any type of idea represented in textual data could be converted to an image before it is processed if advantages of this approach was able to preserve the original semantics.

1.1 Contributions

The results of this thesis exploration suggest that semantic properties are preserved in the images created from textual information, and are encoded in a measurable way. Examination of the results of text to image generative neural networks provide strong evidence that semantics are preserved through the process of converting semantics into language, then into text, then into Glove word vectors, and finally into an image generated by a neural network. We call this novel approach Cerberus, due to the entire process leveraging 3 separate neural networks.

It is the hope of the author that this initial exploration into this uncharted space will encourage further work from the community, which will map out this space and find applications for the properties found.

1.2 Thesis Organization

The next chapter (i.e., Chapter 2) provides the background for all the techniques and approaches used in our thesis and Chapter 3 describes the related work. These chapters build the foundation and motivation for the Cerberus approach defined in Chapter 4. Chapter 5 presents the design and results of the exploratory study that

analyzed whether the Cerberus approach was able to embed semantic information from textual information into images, and the threats to validity are described in Chapter 6. Finally, Chapter 7 concludes this thesis and Chapter 8 outlines some potential directions for future work.

CHAPTER 2

BACKGROUND

The approach proposed in this thesis combines multiple existing technologies and approaches that must be understood in order to put the work in context and fully understand its nature and implications. The following subchapters describe the techniques incorporated in our Cerberus approach.

2.1 Artificial Intelligence

Artificial Intelligence (AI) [31] is the discipline and study of creating computer algorithms and capacities that mimic human intelligence. The reach of this thesis has an impact in the field of AI because in order for machines to make decisions that are based in semantics, the ability of the system to store, process, and search semantics is fundamental.

Though AI is a label that can be attributed to a wide range of areas, it is generally aimed at any work that seeks to make systems or machines that perceive an environment and make decisions based on achieving a goal. Recently the field is more and more interested in AI that cannot only navigate an environment in pursuit of a goal, but can continuously learn from data and update its system to continually learn while it operates.

An interesting problem in AI is that even though much progress has been made in this field, solved problems, such as character recognition or voice recognition, tend to get moved out of the category of AI. This results in a field that can feel like it has not made huge progress in the past century, but if one considers human intelligence as a collection of hundreds or thousands of abilities, many of them have been solved by specific systems that can do one thing well, like recognize pictures of animals, or play a game. These smaller solutions are what we refer to as narrow, or weak AI.

General AI is a system that can learn and perceive more broadly, make decisions in many areas, learn new skills, and develop understanding. If General AI is simply a collection of narrow AIs working in concert, then we have made great progress toward this goal. However, if there is something more, something connected to understanding, and being able to represent and reason around semantics, then our current victories may never get us meaningfully closer to general AI. If so, research like this thesis will be needed to help close the gap between the way humans process semantics and the capacities of machines. It is the hope and belief of the author that if this work, and work like it, can lead to better computer representations of semantics, that progress could be made toward systems that can operate on better semantic models and achieve something closer to human learning and behavior, through being able to process in a manner that is closer to human understanding.

2.2 Neural networks

Neural networks [7] are a technology inspired by the function of biological brains. They are modeled after the behavior of a series of neurons that fire (or do not fire) in response to different stimuli. Neural networks consist of a series of nodes and

connections, organized in layers, that contain weights and biases that will pass on different output values in response to different input values. The number of nodes and connections, as well as the manner in which they are connected, can vary a great deal in different types of neural networks. However, in each case, the overall model is a complex mapping of a series of inputs to a corresponding series of outputs.

Weights and biases within the network are generally initialized randomly, but are not left random (as this would produce an arbitrary output). In order to make the neural network produce the desired output, the network is trained with training data. Training data is a series of inputs associated with a corresponding desired output. The values of weights and biases are changed to better align the input with the output. This is generally done through a process called backpropagation, which measures the difference between a neural network output and the desired output (called the "loss"), and makes changes in the values of the appropriate weights and biases in order to decrease the loss and more closely align the output with the desired output.

The backpropagation process is done iteratively. Each cycle of running through the training data and performing backpropagation is called an "epoch". Different neural networks will exhibit different behavior depending on the number of epochs performed, the number of nodes, the number of layers of nodes, the amount of training data, the variability of the training data, the properties of the input (strongly affected by preprocessing), and even more variables.

Neural networks are incredibly flexible and customizable. It is valuable to think of them as universal function approximators. They will not generally be able to map inputs to outputs in a perfectly and easily understood manner, but they can map just about anything that can be represented as consistent computer input to just about anything that can be represented by consistent computer output. In each situation,

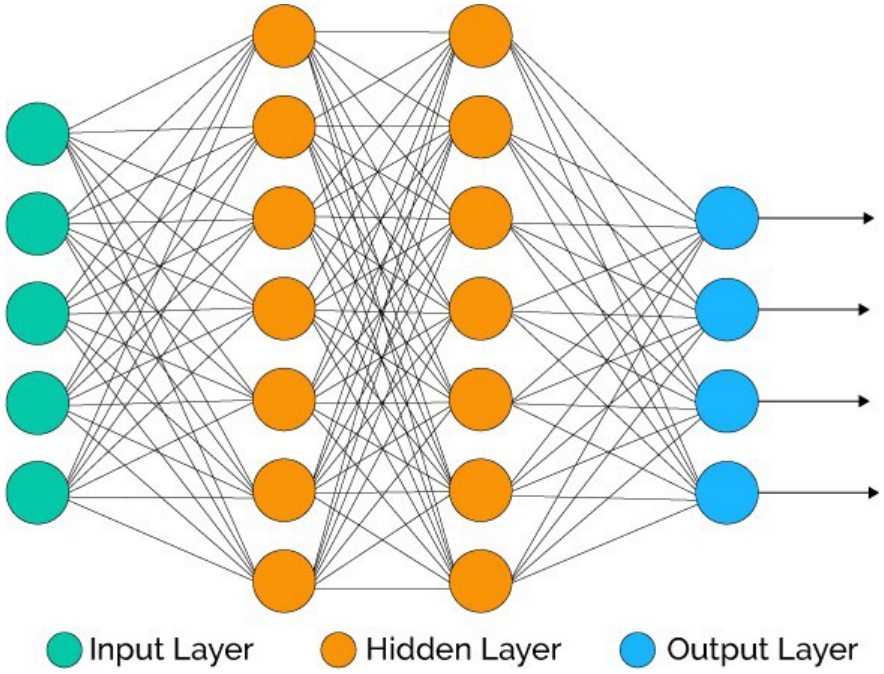


Figure 2.1: An illustration of the basic layout of a simple neural network

they will create a functional mapping that seeks to align the desired input to the desired output.

2.3 Generative neural networks

Generative Neural Networks (GNNs) [38] are a special class of neural networks (see Chapter 2.2) that have a great deal more variability and complexity in their output. Most traditional neural networks will output a binary classification or label from a series. However, generative neural networks tend to have very limited restrictions on their output. In the case of this thesis, the generative neural network can generate any series of RGB values[1] for a series of 100 pixels. The number of different possible

[1]Red, Green, Blue, or RGB values are numerical representations of colors that a computer can display by adding the appropriate amount of each color to a pixel which will result in the desired

outcomes for this output is many orders of magnitude higher than the most successful neural network applications. However, this freedom is a two-edged sword mainly because even though GNNs can produce a wide array of possible outcomes that can feel creative and predictive, they are notoriously difficult to train and it can be very hard for a neural network to successfully map a large variety of inputs to such a wide possibility of outputs [38]. This general limitation of GNNs has a significant impact on the work presented in this thesis (see Chapter 6).

2.4 Information Retrieval

Information Retrieval (IR) [32] is a specific area combining concepts from AI, NLP, and computer science that is concerned with the ability of computers to find and retrieve information that is relevant to a search need. This may be the ability to search the world wide web for content, search a collection of questions for semantic duplicates, search a catalog of movies or games for an item of interest, or many other tasks. The work in this thesis will explore new methodologies in a fairly well established IR task, namely the task of identifying semantically similar or semantically identical texts. The specific case will be a system's abilities to identify any bug report as being duplicate to a previously submitted bug report, a common practice in software engineering [34].

---

color

2.5 Natural Language Processing

Natural Language Processing (NLP) [16] is an area of computer science that works to process, store, and compute human language in computer systems that allow for tasks such as dialog systems, text analysis, automated speech-to-text transcription, and more. The field is primarily interested in representing and processing human language. However, it should be pointed out in the context of this work that human language is primarily useful because it contains, or represents, semantics. The ability to represent semantics in a more efficient and effective manner in computer systems could have significant ramifications on the field of NLP, especially in the sub-field of natural language understanding.

2.6 Corpus

A corpus is a source body of text in Natural Language Processing (NLP) [16]. It will determine the vocabulary of the system, as well as the norms and expectations of an NLP system. A vocabulary consists of all unique words that appear in the document, or series of documents, that comprise the corpus. The choice of the corpus will have a significant impact on an NLP system as its content will determine what words can be processed, how common a word is, what words tend to surround other words or what the context of a word's usage generally is.

2.7 Preprocessing

In order to process text in a consistent manner in a computer system, it is helpful to process the text using a series of predefined steps, standardized by Information

Retrieval (IR) [32]. Common IR practices involve normalizing the text and changing all cases to lowercase, removing stop words (i.e., common words that provide little to no semantic value, or are so common in a corpus that they do not provide unique information about a sentence, such as “in”, “a”, “the”, etc.), lemmatization and stemming [27] (i.e., converting words to their roots). For example, words like “played”, “Played”, “play”, “playing”, and “playful”, would be converted to their root form “play”, instead of being represented by five different words. The IR preprocessing will reduce the vocabulary mismatch problem and will significantly lower the complexity of the system by decreasing the vocabulary size.

2.8 Glove

Glove [26, 20] is a tool produced by researchers from Stanford University that converts words into mathematical vector representations. These vectors are the product of probabilities of surrounding words to a base word in a corpus. This representation produces vectors that have been shown to preserve semantics because the probabilities of surrounding words are semantically meaningful [26, 20]. Glove has been successfully applied to many experiments and is likely primarily responsible for the success achieved by the model introduced in this work.

CHAPTER 3

RELATED WORK

Although to the best of our knowledge no similar approach exists in the literature, a substantial body of work exists for developing and understanding different components used in the approach proposed in this thesis. Specifically, work in neural networks, image processing, semantics, NLP, and development of other methods to encode semantic properties are relevant to this thesis. Neural networks are complex tools and difficult to fully understand. They allow for very complex mappings of a set of inputs to a set of outputs. If trained and tuned properly, neural networks have a capacity to perform complex tasks at human or even superhuman levels of performance and on a wide range of inputs, including text and images. Many authors have found success at using them to perform tasks related to semantics using natural language [6, 9, 12].

Although neural networks [7] (see Chapter 2.2) have been around for many decades, their capacity, attributes, and limitations are still not fully understood. In 2001, Caruana and Lawrence showed that what had become "conventional wisdom" about the limitations of backpropagation (i.e., that it could not result in a model that generalizes well) did not hold [3]. Over the past five years, deep learning [17] has broken through many perceived limitations. Hurdles in speech-to-text accuracy, image classification, and more, have been overcome by progress in this arena. However, the limitations of

deep learning are still not fully understood and the debate continues as to what its long term potential is [17].

It is the opinion of the author that the complexities of the technologies, and the range and depth of their potential applications and modifications make for a scenario where one should be slow to declare hard limits, and empirical data and experimentation should be encouraged in order to further define the limits and attributes. A younger but equally vibrant technology exists in generative neural networks [38] (see Chapter 2.3). Few limitations have been suggested, and exciting work is being done in leveraging semantics in text and turning them into pictures [10], as well as using pictures to generate semantically appropriate descriptive text [13].

The ability of neural networks to rank the similarity of text based semantics has been shown by researchers at the Institute of Computing Technology, Chinese Academy of Sciences in Beijing (ASB) [24]. The Chinese ASB's approach may appear to be a similar approach to this work's at first glance, but it is actually quite different. Their research illustrates the ability to encode similarity between two texts as a single image that consists of pixels that individually represent the word similarity between different combinations of words from each text, and then uses a neural network to analyze the resultant image and output a score that represents the level of similarity between the two texts that were used to generate the single image. On the other hand, Cerberus uses a neural network to generate two images, each one from its own single text input, and then compare the similarity of the generated images. Also different from the Chinese ASB's approach, the encoding of the semantics for this work is computed via a generative neural network and not by a deliberate algorithm. Of equal impact to this experiment is the work that has gone into image processing and generation. Finding and tracking semantics in the domain of images provides many

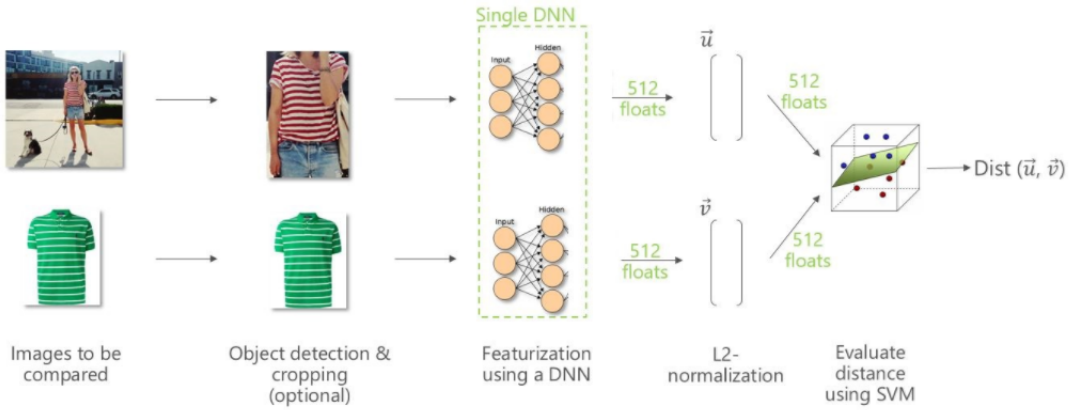


Figure 3.1: A visualization of the methodology used by Microsoft's CNTK for measuring image similarity. Images input into the tool are output as 512 point matrices that can be compared

unique complexities, challenges, and opportunities [13, 1, 3]. However, this very complexity is the source of the driving questions of this research. If the processes were more simple, the answer to the driving questions might be simple to derive, or even be intuitive. However, the reality of the situation is simply not the case, and it was therefore the belief of the author and the thesis committee that the experiment had merit and should be pursued.

Lastly, it is important to reference the work of Patrick Buehler, Senior Data Scientist at Microsoft. Buehler shows the ability of a neural network to rank the similarity of two pictures [25]. This is done by using a single deep neural net to convert images into a matrix of 512 normalized floating point numbers. The resulting matrices can then be numerically evaluated to measure the degree of similarity between two images (see Figure 3.1). This work and toolset built by Microsoft (Microsoft CNTK), are used in the experiments conducted in this thesis.

CHAPTER 4

APPROACH

This thesis creates and applies Cerberus, a novel approach to explore and examine the potential of encoding semantics into pictures in a manner which machines can process and make decisions on. As discussed in the previous chapters, this approach uses multiple neural network-based technologies, Glove, a generative neural network, and Microsoft CNTK as illustrated in Figure 4.1.

For ease of reference, we have named the novel approach Cerberus, after the mythical three-headed dog, due to the fact that a total of three neural networks are utilized. The first two neural networks are used to generate pictures from text. Specifically, Glove transforms text into word vectors, followed by a generative network which transforms vectors into images. The third neural network examines the properties of the images using Microsoft CNTK.

The implementation of Cerberus (see Figure 4.2) consists of a series of steps, beginning with a transformation of source texts into images, and ending with a comparison of images in a manner measurable by a machine.

The texts chosen for source texts were bug reports that had been tagged as "duplicate" or "non-duplicate" from the popular open-source Integrated Development Environment Eclipse[1]. Specifically, the dataset was extracted from the curated

[1]https://www.eclipse.org/

Figure 4.1: An outline of the novel model used in this work. We call the model Cerberus due to its use of three neural networks. Text is input into a GNN, which outputs an image. That image is then fed into the Microsoft CNTK, which outputs a 512 point matrix that represents visual aspects of the image.

Eclipse dataset by Lamkanfi et al. [5]. The bugs were parsed and paired as duplicate, non-duplicate, or identical. Image generation was accomplished with a generative neural network modified from a Keras based GNN built by Xianshun Chen, a data scientist and software architect working out of Singapore [35]. The resultant pictures were then fed into a modified version of the Microsoft CNTK neural network that output 512 floating point matrix representations of each picture. The resulting matrices were then compared for numerical similarity (Euclidean distance and cosine similarity).

This process allows us to quantify semantic encoding in the images by having images generated from text that we know to be semantically similar and measuring how similar the images are. If the semantics are being encoded in a measurable way, then we will be able to identify images as being similar when they have been generated from semantically similar text.

To measure this in a systematic manner, bugs are ordered into pairs and fed through Cerberus. The pairs, already known to be semantically similar or not (i.e., bug duplicates or not), can then be examined for image similarity. Walking through an implementation of Cerberus in the experiments in Chapter 5 will provide more

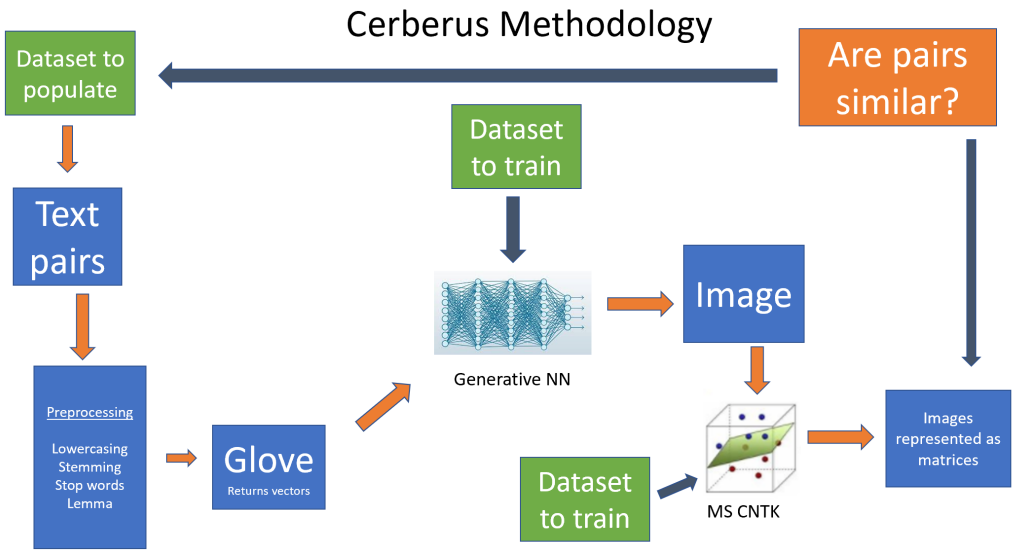


Figure 4.2: An detailed overview of the Cerberus methodology. Bugs are preprocessed, tagged, turned into images, then evaluated

details about the specific steps, datasets, and configuration used.

CHAPTER 5

EXPERIMENTS

This experiments chapter details the results of the exploratory study that investigates whether semantic information can be captured and embedded in images. This chapter is divided into three parts: preliminary experiments of Cerberus on a small scale dataset (Chapter 5.1), the primary experiments of Cerberus on a larger dataset (Chapter 5.2), and finally, an investigation into the impact and role played by Glove in the Cerberus approach (Chapter 5.3).

5.1 Preliminary Experiments

This chapter details the preliminary experiments performed using Cerberus, and provide some initial evidence that semantic information from text can be encoded into images.

Following the Cerberus approach (see Chapter 4), we describe the dataset and emphasize four important steps:

Step 1: Generate Dataset of Text Pairs

Although the Cerberus approach is generic and can be applied on any dataset (from any domain) containing textual information, we chose a real-world problem from software engineering, namely the detection of bug duplicates [34]. Development of

large open-source software typically involves maintaining issue tracking systems that keep track of features and bugs that need to be implemented or fixed. When new bug reports are created, triagers (i.e., developers in charge of a preliminary evaluation of bug reports) need to determine if a bug is a duplicate of existing ones, and mark them as duplicate if they are. This operation would reduce the number of open bugs in the bug repository and reduce the developer effort. When dealing with hundreds or thousands of bug reports submitted daily, it is important to have an automated approach for the detection of bug duplicates.

The dataset was chosen from the Eclipse bug repository following a similar methodology described by Wang et al. [34]. Specifically, we randomly selected 225 bug reports containing 44 pairs of duplicate bug reports, that were already annotated by the triagers, which we used as a "gold set" (which is sometimes referred to as a "ground truth" or "oracle") in our evaluation.

Step 2: Convert Text Pairs to Image Pairs

The Keras text-to-image generative neural network was used for converting text into images [35]. The generative network was trained on Pokemon[1] images and descriptions, and although its structure did not allow for use at scale (one of the challenges to be overcome in the primary work), it served as a prototype (proof-of-concept) that could be expanded and scaled upon.

Bug pairs were fed into the generative network and three sets of ten images were generated (due to the scaling issue). The sets of ten consisted of five image pairs

[1]Pokemon are cartoon-like characters with appearances and abilities that are related to their descriptions.

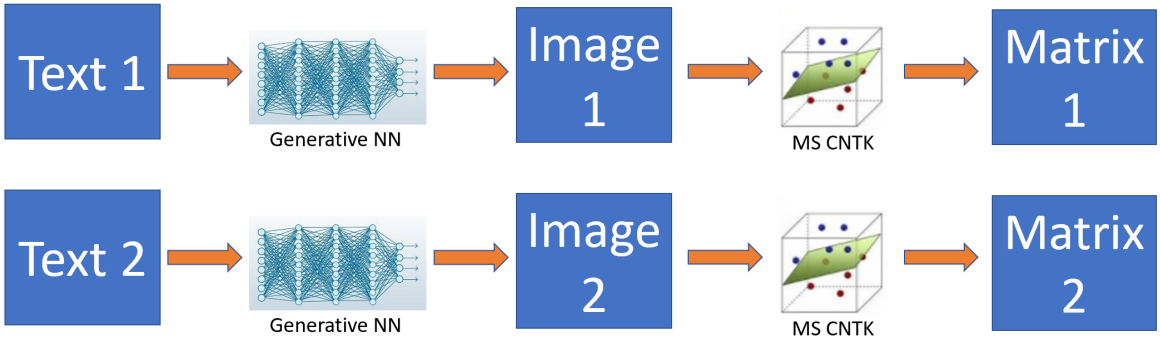


Figure 5.1: An outline of Cerberus with pairs

each. One set of exact duplicate text pairs, one set from semantically similar but textually different text pairs (tagged duplicates), and one set from unrelated text pairs comprised the three sets of ten.

Step 3: Measure Image Pair Similarity

The Microsoft CNTK repository for measuring image similarity was used, with slight modifications [25]. This allowed for the transformation of each image into a 512 floating point matrix representing the properties of the image. CNTK was configured with a pre-trained model based on the Coco [14] dataset, but was "refined" (further trained and modified) on a series of images of clothing that represent either spotted, striped, or leopard print. This means that the output matrices were pushing resultant values that were most effective at making distinctions in visual patterns between those three categories. The difference of each matrix pair's Euclidean distance was summed to get a preliminary measure of the distance between the matrices on average.

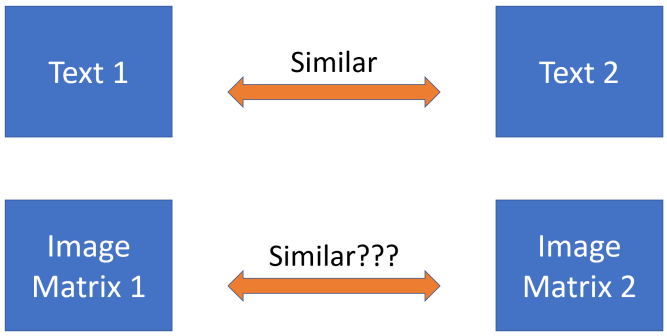


Figure 5.2: An illustration of the question that will be used to measure if semantics are encoded into images by GNNs in a measurable way: Would the similarity of resultant matrices correlate with the text similarity?

Step 4: Check for Correlation between Image Pair Similarity and Text Similarity

The results were examined to determine if the image pairs' matrix similarity (Euclidean distance) correlated with the semantic similarity of the source texts identified by the dataset. In other words, we examined if the Euclidean distance of duplicate and identical bug pairs was greater than the Euclidean distance for non-duplicate bug pairs.

A sanity check was performed on the results of the image pairs produced from bug reports having identical text. Even at a glance, it is easy to determine that the images are identical (see Figure 5.3 for an example). This was the case for all pairs that were created from identical text. Each image pair looked identical and the difference between the resultant similarity matrices was measured as 0, as expected.

Most of the semantically duplicate bugs were judged as being closer than semantically non-duplicate bugs (see Figure 5.4 for an example), however, as expected, semantically different bugs (i.e., non-duplicate bugs) resulted in dissimilar images

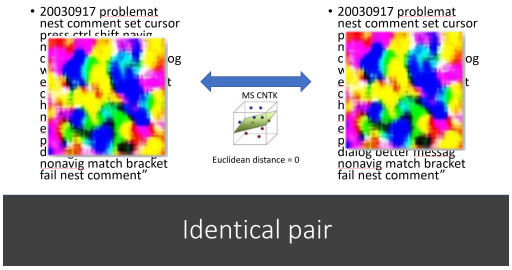


Figure 5.3: A pair of images generated from identical bug reports from the preliminary dataset. They are used to confirm that the model is producing images in a consistent manner. The resultant matrices have a Euclidean distance of 0.

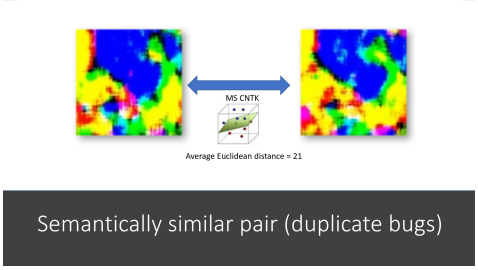


Figure 5.4: A pair of images generated from semantically similar (duplicate) bug reports from the preliminary dataset. The resultant matrices have an average Euclidean distance of 21.

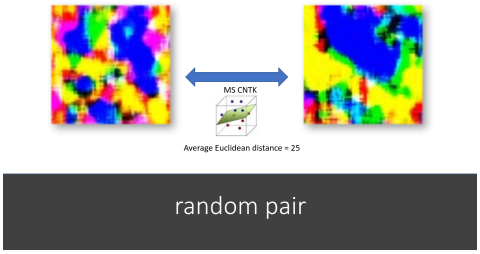


Figure 5.5: A pair of images generated from semantically dissimilar (non-duplicate) bug reports from the preliminary dataset. The resultant matrices have an average Euclidean distance of 25.

that were quantifiable distant from one another (see Figure 5.5 for an example).

Setting a threshold at 22, representing a natural "gap" between values that were clustered above/below the threshold, the model would classify bug duplicates with an accuracy of 70%, precision of 66%, recall (i.e., true positive rate) of 80%, false positive rate of 40% and an F1 value of 69%.

These promising results show potential in the methodology, but the number of

Metric	Results
Greatest measured Euclidean distance	33
Average Euclidean distance of non-duplicates	25
Average Euclidean distance of duplicates	21
Threshold	22
Accuracy	70%
Precision	66%
Recall (true positive)	80%
False positive	40%
F1	69%

Table 5.1: Summary of the results of the preliminary experiment

examples is not at a statistically significant level, and the supporting neural networks were not trained or tuned in a manner directly supporting the characteristics of the data. These limitations and results gave rise to many questions: Would the results hold at scale? How greatly would the results differ if the generative network was trained on data similar to the test data? Could the image generator be trained with making images that better capture and focus on the words and properties that are most important in the domain? If so, could this result in images that have more differentiation and representation of the words and word order that tend to be most important in the target domain? Could a generative network be trained for this type of task specifically? What is a good picture of a bug report? Is it possible to use the whole model in concert to train pieces of it? Could this create a network specialized at making pictures that represent semantic similarity well? How would results differ on more natural language, like a dataset for duplicate questions? How much could the results improve if the similarity measure network was trained or refined on data that is similar to the test data? Are the similarity measures indicative of just word similarity, and not as much order? Are there ways to better understand and control what the generative network is sensitive to and align that with differences that are important

in the target space? If the process did specialize well when focused and tuned for a specific domain (like many neural networks do), could it generalize well? Could the image generator be trained on all English? Could the similarity network be general enough to work on anything? What are the bounds of the methodology? How long can the input texts be? How semantically dense can the text be? Would a well-developed model follow the trend and result in a powerful tool that has great results when properly tuned and in ideal circumstances, but lacks easy general application, and lacks the visibility and transparency of the process to allow for large-scale trust and implementation, or significant clear insight?

Suffice it to say that one step with promising results created far more questions than it answered.

These questions merited further investigation and experimentation. Even with the networks using pre-trained and arguably inappropriate models, the system looked like it worked, and, as the questions above indicate, there are large degrees of potential flexibility and adaptation in the experiment's model and tools. Therefore, we scaled and tuned this experiment in an attempt to pursue the answers to a subset of these questions and gain more insight and understanding into the potential of encoding semantics into images via neural networks.

In order to build a system that could answer these questions, the first questions to address were "Would the results scale?" and "How greatly would the results differ if the generative network was trained on data similar to the test data?". The image generator could be trained with making images that better capture and focus on the words and properties that are most important in the domain. This could result in images that have more differentiation and representation of the words and word order that tend to be most important in the target domain. In addition, an appropriate

dataset would need to be found, and the model would need to be modified to be able to train on it.

The preliminary experiments resulted in limited but promising results, identification of significant challenges to overcome, and raised additional questions and avenues of investigation.

5.2 Primary Experiments

In order to investigate the original driving questions of whether images can capture and embed semantic information from text, as well as to explore the questions added by the preliminary experiments (Chapter 5.1), the Cerberus approach required a generative neural network with two primary attributes:

First, the generative neural network needed to be trained on a sufficiently large dataset that should allow for a large range of semantic mappings. This characteristic would allow us to test the main theory that a GNN may be able to generate images in a manner that maps semantics into images. If the training was not robust, then the mapping would theoretically be more random in nature. This is because the values of weights and biases in the network that will map words to image properties will initially be random and only training would change them. Therefore, a large dataset was used to produce mappings in the model that would be reflective of a large range of relationships between text and corresponding picture properties.

Second, the generative neural network should have an ability to generate images from textual information at scale. This would allow the experiment to process enough text pairs that the resultant image pairs could be evaluated in a manner that is statistically sound.

To incorporate these changes into Cerberus, a large number of technical requirements had to be met. We will mention and discuss a few of the important steps and details in this chapter, and we refer the interested reader to our Appendix A for a comprehensive list of technical details.

5.2.1 Model 1: Flowers Dataset

Motivation for the Oxford 102 Dataset

The University of Oxford 102 Category Flower Dataset is a dataset that has been used multiple times to successfully create adversarial generative neural networks generating images from text with good results [38]. It consists of 8,189 unique pictures of flowers, and textual descriptions of each picture. For these reasons it appeared to be a good candidate for the experiment and was chosen.

Implementation Details

The dataset consists of *.jpeg* images, that were converted to *.png* images to be compatible with the generative network. The pictures' pixel ratios were changed to be consistent, square, and an appropriate size for the available hardware. The neural network training was done at 1,000 epochs, and the model was tested for fitness for the experiment.

Results at 1,000 epochs

Overall, the results looked promising and positive. Visual inspection did not raise any immediate concerns (see Figure 5.6 for an example), although there was a possibility that the duplicate bugs were not all visibly as similar to the human eye as the

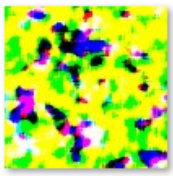


Figure 5.6: An image produced from a bug report input into the GNN after it had been trained on the flowers dataset at 1,000 epochs

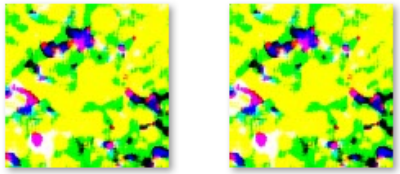


Figure 5.7: Once again used as a sanity check, this image shows that identical texts produced identical images when fed to the GNN trained on the flowers dataset at 1,000 epochs

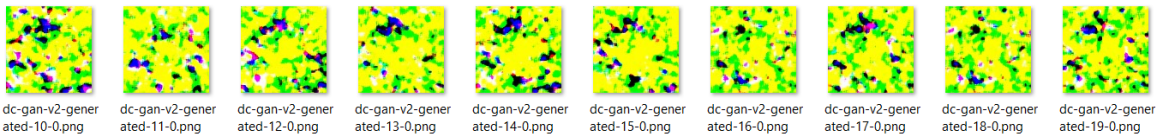


Figure 5.8: Images produced from 5 pairs of duplicate bugs transformed by the GNN trained on the flowers dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

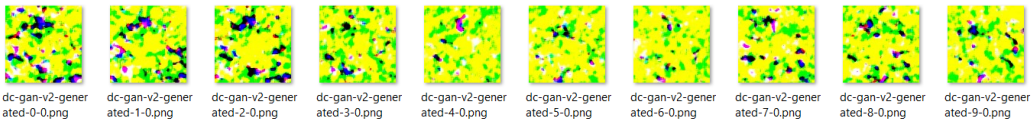
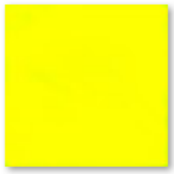


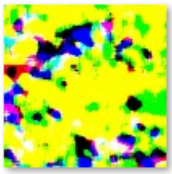
Figure 5.9: Images produced from 5 pairs of non-duplicate bugs transformed by the GNN trained on the flowers dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

preliminary outcomes were. Once again as a sanity check, identical text pairs were checked to see if they produced identical image pairs (see Figure 5.7). The sanity check was positive, however, the results of other tests were more concerning.



dc-gan-v2-gener
ated-4-0.png

Figure 5.10: An image produced when 10 sentences describing a flower were input into the GNN when trained on the flowers dataset at 1000 epochs



dc-gan-v2-gener
ated-9-0.png

Figure 5.11: An image produced when 1 sentences describing a flower was input into the GNN when trained on the flowers dataset at 1,000 epochs

The generative neural network ran with a set of ten sentences describing a flower to generate an image. This was done because the ten sentence format is the format that was used for the image descriptions in the original dataset. Intuitively, the generative neural network should produce something similar to the flower described in the training text when the training text is run through the GNN, or at least show aspects that are reflective of elements described in the texts.

There was no immediate and obvious reason the network would produce a solid yellow image with a description from the training set. More tests were run to explore the nature and function of the GNN's model.

In order to better understand this behavior and the attributes and behavior of the generative neural network, different texts were fed through the network.

It appeared that the tendency of the model to map to yellow was a default value

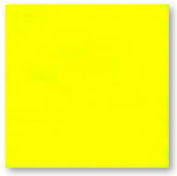


Figure 5.12: The image produced when a blank text file was input into the GNN when trained on the flowers dataset at 1,000 epochs

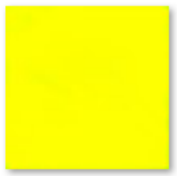


Figure 5.13: An image produced when 1,035 words from a random website was input into the GNN when trained on the flowers dataset at 1,000 epochs

or direction, and thus a blank text would map to a yellow image (see Figure 5.12). In addition, it appeared that the tendency of the model to map to yellow increased as the quantity of text increased and there was more text to map. As shown in Figure 5.13, a yellow image was obtained when 1,035 random words were used as input into the GNN. However, the single line and bug data show that text which is less related to the training text, at least in small amounts, resulted in patterns with more variability.

It is possible that the training had fallen into a strange local minimum where the best answer it could find to minimizing the loss function was to map everything to the same color. These results triggered a series of other questions. What about the fact that a blank text file resulted in a yellow image? Were all the images the same color of yellow, or was the model using subtly different shades of yellow to map to, which CNTK could potentially detect?

To answer these questions, we explored the possibility of the CNTK neural network to identify similarities that were not obvious to the human eye. CNTK examined the pairs and converted them into matrices. The results are summarized in Table 5.2. The matrix pairs had an average Euclidean distance 14.94 for non-duplicate pairs, and an average Euclidean distance of 14.50 for duplicate pairs. These results were not encouraging because they showed a lack of detectable difference in the resultant images controlling for the factor of source texts being semantically similar or not.

Type of text/image/matrix pair	**Average Euclidean distance**
Non duplicates	14.94
Duplicates	14.50

Table 5.2: Results from the model with the GNN trained on the flowers dataset at 1,000 epochs

The pixel values of the yellow output were evaluated and they were all the same shade of yellow. Why would seemingly all text, in large enough quantity, map to this? Was it just the words in the training data that pushed results towards yellow?

These new results sparked numerous avenues of further investigation, but answering all these questions would be beyond the scope of this thesis. Instead, we continued our original exploration and investigated some other factors that constitute Cerberus.

The methodology used to convert text vectors into a single factor (allowing for sentence descriptions of variable length) representation was changed from a sum of vectors methodology into an average of the word vectors, and the system was retrained and tested, but the results did not appear to be significantly different.

We conjectured that either a new generative network would need to be reused or created from scratch, or we could investigate a new dataset to determine if it had any influence on the results. The issue seemed to be related to the words in the training dataset being used for generation (the model would tend toward yellow with few

words if the words were related to flowers or flower descriptions). It is also important to emphasize that the training data for flowers do not have a very robust vocabulary. This is intuitively true as there simply are not that many different words people tend to use when describing flowers. Moreover, when sampling some of the dataset textual descriptions, we discovered that some had too many lines, and some even had descriptions of flowers that were conflicting with other descriptions. Enough doubt was cast on the dataset being an appropriate fit that a new dataset was investigated.

In addition, we conjectured that a major problem could be that lack of vocabulary in the training set, and the range of words used in the test vocabulary that does not appear in the training vocabulary. Therefore, for the next step in our experiment (see Chapter 5.2.2), we searched for a new dataset with a more diverse vocabulary that would provide more variety in its mappings and allow for the identification of any possible error in the building process that might have been made in the first iteration described in this chapter.

5.2.2 Model 2: Coco Dataset

Coco [14] is a well-known dataset in computer vision and AI. Its current version contains approximately 330K images, each with five descriptions per image. It is extremely diverse in terms of vocabulary and image variety.

Motivation for the Coco Dataset

The Coco dataset consists of a wide variety of pictures with a much larger vocabulary resultant from the descriptions being about a wide array of scenes, items, people, and animals. We hypothesized that this wider breadth of vocabulary and training would result in more meaningful mappings for the text descriptions used as input.

Intuitively, if the words have not been trained to image properties, then the mappings would be random, as a result of the initial random weights and biases in the nodes of the neural network. Therefore a large vocabulary in the training data and a large, but consistent enough to be trainable, set of corresponding images should have the best chance of resulting in consistent and semantically meaningful mappings.

Referring to the question "How greatly would the results differ if the generative network was trained on data similar to the test data?" from the Preliminary Experiments (Chapter 5.1), even though we are still using bug report descriptions as input, Coco should result in a significantly greater frequency of test vocabulary appearing in the training vocabulary.

Implementation Details

Coco descriptions were provided into one giant JSON[2] string representation. Java scripts were written to parse and extract those descriptions and pair them with images for training. We used the image processor XnView [36] to convert in batches *.jpg* images into *.png* files. About 2% of the dataset images could not be converted to the *.png* format and were removed from the analysis. In addition, we excluded images that were not resized correctly. The corresponding text files for the excluded images were discarded as well. After this data cleansing process, we obtained 4,234 images and corresponding text files that we used to train a new model.

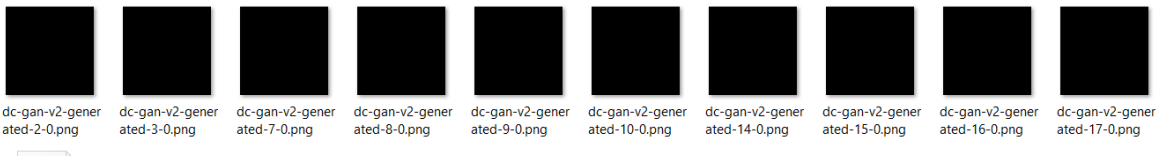


Figure 5.14: Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 1,000 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

Coco 1,000 Epochs of Training

This chapter describes the results obtained while training the generative neural network model on the Coco dataset using 1,000 epochs.

The results were no longer solid yellow as in the Flowers dataset (see Chapter 5.2.1), instead, the results exhibited a similar pattern and were all black (see Figure 5.14), which was indicative of no significant improvement. These results indicate that trained at a large number of epochs, the model mapped all its original training data to the same n-dimensional point, and that at least for words in the training vocabulary, mappings would not be discriminative for semantic differentiation because all the mapping resulted in identical output. In other words, regardless of the input, the output would be black for every pixel.

Unanticipated behavior in neural models can often be the result of overtraining. In order to investigate if the overtraining factor caused this strange behavior in the result, the model was re-trained using a lower number of epochs and evaluated through a couple of tests. The first test used the text "red" and the second was a blank (i.e., empty) text file (see Figure 5.15).

[2]https://www.json.org/

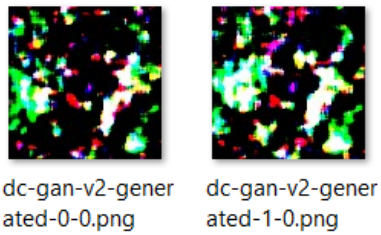


Figure 5.15: Images produced by a text file containing the single word "red" and a blank text file transformed by the GNN trained on the Coco dataset at 1,000 epochs

It is an important difference that in this model, a blank text does not result in a solid color. This suggests that the problem could be overtraining and resulting in a local minimum if all words in the training vocabulary tend in that direction, but it is not a default value that a blank text file would still be mapped to. This was an encouraging sign in our exploratory study.

Coco 100 Epochs of Training

The same images and texts were used to train the model at 100 epochs.

As indicated in Figure 5.16, it was reassuring to observe more than just a solid color, but the visual similarity of the images is concerning as the texts are all different. More investigation was needed to determine if overtraining was the culpable factor, but to a lesser extent.

CNTK examined the pairs and converted them into matrices. The matrix pairs had an average Euclidean distance 18.02 for non-duplicate pairs, and an average Euclidean distance of 17.41 for duplicate pairs, as summarized in Table 5.3. There was very little measurable difference on average between the images that are products of semantically similar text strings compared to those that are semantically dissimilar.

The pictures generated from the single word "red" and blank text (see Figure 5.19)

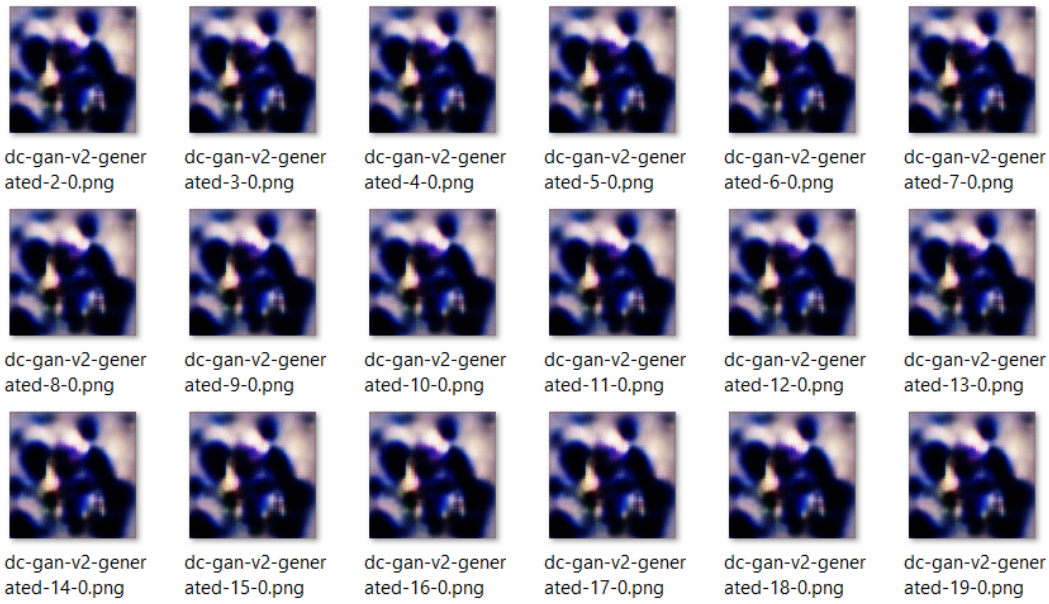


Figure 5.16: Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 100 epochs. All images are similar to each other but are generated from distinctive texts

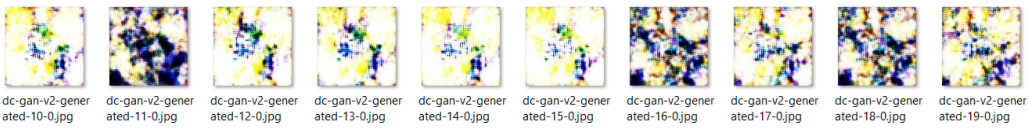


Figure 5.17: Images produced from five pairs of duplicate bugs transformed by the GNN trained on the Coco dataset at 100 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

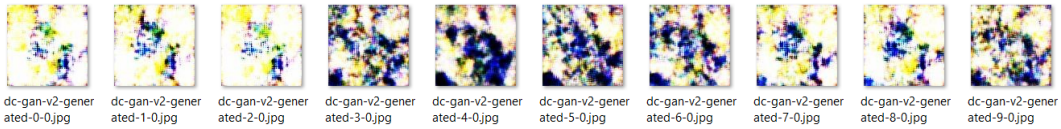


Figure 5.18: Images produced from five pairs of non-duplicate bugs transformed by the GNN trained on the Coco dataset at 100 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

Type of text/image/matrix pair	Average Euclidean distance
Non-duplicates	18.02
Duplicates	17.41

Table 5.3: Results from the model with the GNN trained on the Coco dataset at 100 epochs

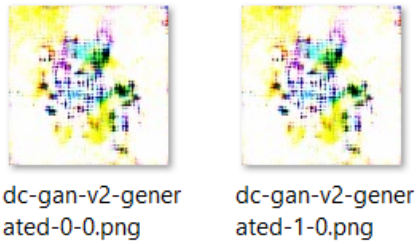


Figure 5.19: Images produced by a text file containing the single word "red" (left) and a blank text file (right) transformed by the GNN trained on the Coco dataset at 100 epochs

look significantly different from their 1,000 epoch training counterparts (see Figure 5.15) but it is concerning that they look very similar and that the "red" input shows no indication of being mapped to red-like image qualities, and does not seem to be any more "red" than a picture generated from a blank text.

Coco 10 Epochs of Training

The same images and texts were used to train the model at 1,000 and 100 epochs were used to train the model at 10 epochs and the results are presented and discussed in this chapter.

Although the images in Figure 5.20, which were generated from distinct input texts, still do not resemble the images originally linked with their descriptions, it is interesting and positive that there is more differentiation between them. It looks like not all of the training vocabulary is being mapped in a similar direction, which is encouraging. However, much of the vocabulary is being mapped in a similar direction, and we observed that at 1,000 epochs (see Chapter 5.2.2) all the vocabulary is mapped in an identical manner, suggesting that the variability encountered at 10 epochs may stem from the randomness that has not yet been trained out. Therefore, we conjecture that if semantic similarity (or difference in text) is represented by visual similarity (or

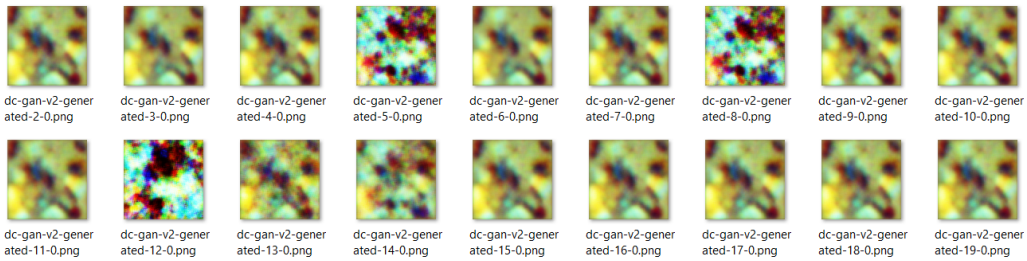


Figure 5.20: Images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 10 epochs. All images are similar to each other but are generated from distinctive texts

Figure 5.21: An in-depth view of three images produced by text files containing five sentence descriptions from training examples transformed by the GNN trained on the Coco dataset at 10 epochs

difference in the images), then that representation is likely not due to the generative neural network training (since the more it trains, the more indistinctive the results are), but that the similarity and difference are present due to other factors. For instance, they could stem from initial values propagated from Glove vector generation.

It is ironic that the images produced at this level of training are reminiscent of the flower-like images that were expected from training the generative neural network on the flower dataset (see Chapter 5.2.1).

The pictures generated from the single word “red” and blank text (see Figure 5.22) look strikingly similar, a pattern observed when the model was trained with 1,000

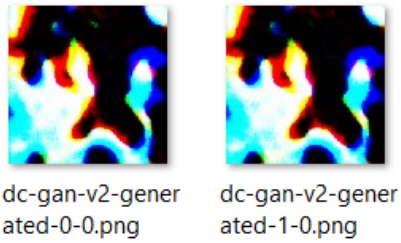


Figure 5.22: Images produced by a text file containing the single word "red" (left) and a blank text (right) file transformed by the GNN trained on the Coco dataset at 10 epochs

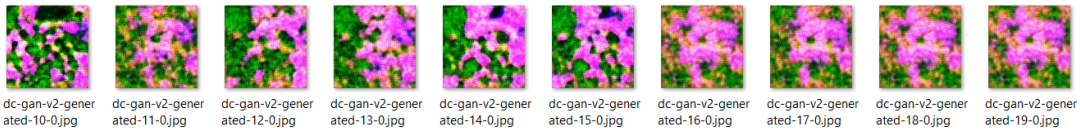


Figure 5.23: Images produced from five pairs of duplicate bugs transformed by the GNN trained on the Coco dataset at 10 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

epochs (see Figure 5.15) or 100 epochs (see Figure 5.19). In other words, even at 10 epochs the textual information embedded in images is not discriminative enough for distinct text.

Analyzing the bug reports, Figure 5.23 and Figure 5.24 illustrates a subset of sample images produced from five duplicate bug reports and sample images generated from five non-duplicate bug reports, respectively. At a first glance, the images of the first pair of duplicate bugs (i.e., the first two images from the left in Figure 5.23) are dissimilar, whereas the remaining four pairs of duplicate bugs are very similar, which is in line with our expectations. On the other hand, the fourth pair of non-duplicate bugs (i.e., the seventh and eight images from the left in Figure 5.24) are similar, which is not something we anticipated, whereas the remaining four pairs of non-duplicate bugs are very dissimilar.

To get a more quantifiable analysis of the differences between the images, we examined the pairs of duplicates and non-duplicates using CNTK, and converted

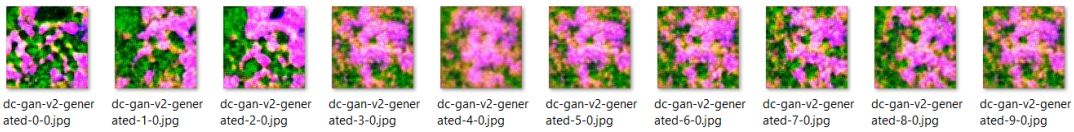


Figure 5.24: Images produced from 5 pairs of non-duplicate bugs transformed by the GNN trained on the Coco dataset at 10 epochs. The first two images represent the first pair, the third and fourth images represent the second pair, and so on.

Type of text/image/matrix pair	Average Euclidean distance
Non-duplicates	17.77
Duplicates	14.20

Table 5.4: Results from the model with the GNN trained on the Coco dataset at 10 epochs

them into matrices. The matrix pairs had an average Euclidean distance 17.77 for non-duplicate pairs, and an average Euclidean distance of 14.20 for duplicate pairs, as described in Table 5.4.

The average distance of the semantically similar text pairs (i.e., 14.20) is smaller on average than those that are not (i.e., 17.77). This difference between distances is higher than the differences in distances for the model trained with 100 epochs (i.e., 17.41 for duplicates compared to 18.02 for non-duplicates as shown see Table 5.3). It is promising to see results that have the same pattern as the results in the Preliminary Experiments (see Chapter 5.1). However, since these results improve as training decreases, it is possible that the similarity is not attributed to the model mapping semantics to images in a trainable way, but that the similarity was already (i.e., inherently) there, and the generative network may be working against encoding semantics in the images in a quantifiable way. Perhaps the GNN is not working to encode semantics at all, but simply altering the process. In other words, it is possible that the less training is performed on the model would result in fewer weights and biases that map from text to image, and the less it can harm the process by training it in an unhelpful manner.

It is counter-intuitive that the less the GNN is trained the better the results are. Moreover, it is puzzling why a single word with strong ties to image qualities (i.e., "red") produce such similar output to a blank text (as shown in Figure 5.22).

At this point, results are more and more indicative that the semantics that appear to be present in the generated images may be a result of the semantically meaningful uniqueness of the vectors produced by Glove. Showing evidence that such semantic content persists through a generative neural network, even with largely randomized mappings, is itself a contribution in the field, but a neural network capable of mapping textual semantic representations to images with more semantic content retained or clarified would, of course, be of much greater value.

5.2.3 Discussion of Results

The Primary experiments described in Chapter 5.2 modified a generative neural network to accept large datasets. After training on a large dataset produced inconclusive results (Chapter 5.2.1), another version was built using a new (and more vocabulary diverse) dataset (Chapter 5.2.2). The entire process was tedious and complex, involving numerous technical problems and inventive solutions. Eventually, the models were able to produce image outputs for textual information used as input.

The results of the created models were surprising and interesting, exhibiting behavior that was unexpected and suggestive of the learning task for the neural network being too complex. This would be understandable as mapping thousands of semantic concepts with images via a summed vector of word representations, to thousands of numerical representations of images is a significant task for a single neural network. Some tests showed a complete lack of semantics being preserved in the process (see Chapter 5.2.1). However, as training was decreased (see Chapter 5.2.2),

evidence of semantics being encoded in the images increased in a quantifiable way, as captured by the Euclidean distance.

It is the belief of the author that these results support, but do not prove, that semantics can be encoded by a GNN in a measurable way, even when the GNN consists of largely randomized mappings. Due to the fact that results were superior when using a network with limited training, we conjecture that the majority of semantic encoding that is consistent with the texts is residual from the Glove vectorization, and not due to the generative network successfully mapping semantics in images in a manner learned from training.

There are a huge number of variations in the details about how one can approach this method, and regarding the thousands of factors and future directions that can be investigated further. However, all these potential promising directions are beyond the scope of the exploratory study presented in this thesis. An extremely small sample of factors that can be altered includes different neural network types, a different number of layers, different training examples, using an adversarial network to modify the pictures, training the CNTK similarity network with appropriately tuned data, tuning and parameterizing the neural networks, and more, could be viable approaches. Neural networks are extremely fine tunable, which means that there may be potential in this general approach. However, the results of these specific experiments indicate that the training on this particular GNN and these datasets, does more to obfuscate the semantic encoding than support it. As we can see in the results for these models, the more comprehensive and thoroughly trained the generative network is, the worse the semantic preservation is. One one hand, this could be explained by the fact that the mappings we are asking for are not directly related to the input and output of the GNN, but are concepts related to both, ideas that we are trying to represent in

different ways. It can be difficult to fine tune and train a neural network even when the inputs and outputs are simple and directly observable. Perhaps more steps, and even simpler steps, are needed to connect the semantics on each end of the process and allow for training that would preserve the semantics instead of distorting them.

In future work, it would be desirable to try building a different base model, or make fundamental changes to the neural network. The preliminary results (see Chapter 5.1) looked promising, and they were worth investigating further, but it appears to be the case that the semantic encoding and detection that was measurable was due to Glove (see Chapter 5.3), and any GNN (as long as it was consistent) representing Glove visually would produce results that CNTK would be able to measure as similar.

5.3 Investigation of Glove's impact

In order to further explore our conjecture described in Chapter 5.2.3, which discusses the results of the Primary experiments, this chapter describes additional experiments that were performed to investigate the degree of Glove's impact on the results, as one of the many factors in the Cerberus model. A new model consisting only of Glove was built and evaluated as a baseline when comparing it against Cerberus, on the same Eclipse bug dataset (see Chapter 4). For these experiments, in addition to the Euclidean distance metric, we also computed the cosine similarity, a common metric used in Information Retrieval [32]. The cosine metric can compute the similarity between two vectors and output a value between $[-1..1]$, where the value 1 represents identical vectors and the value -1 represents diametrically opposed vectors. In other words, the higher the value, the more similar the two compared entities (vectors) are.

The measures of how successful each approach (i.e., Glove or Cerberus) is at

making predictions provide evidence of the approach's ability to measure semantic similarity. A widely used measurement is precision [32], which represents the percentage of correct guesses, at a particular threshold, that a method achieves when predicting if a bug pair was duplicate or non-duplicate, based on the Euclidean distance or cosine similarity of the pair.

In addition to comparing the precision of the different methods, this experiment also evaluates the average measures for duplicate bugs and non-duplicate bugs. It should be intuitive that an effective similarity measure should be able to detect similar items as being closer on average, and dissimilar items being further on average.

All of the code involved in these additional tests are available upon request, by contacting the author, and will be provided in the form of a Jupyter Notebooks [11]. These notebooks import Glove, the bug dataset, output from CNTK, and other supporting files. They organize the data into $dataframes$ with Pandas[3] and generate the results needed to make predictions and measure the precision.

Metric	**Duplicate**	**Non-Duplicate**
Average Euclidean distance, Glove	24.137	52.370
Average Euclidean distance, Cerberus	17.057	17.733
Average cosine similarity, Glove	0.7767	0.5533
Average cosine similarity, Cerberus	0.8809	0.8719

Table 5.5: Summary of the results of when comparing Glove (baseline) and Cerberus on the Bug Duplicate dataset

Table 5.5 summarizes the results of comparing the baseline Glove with Cerberus. An examination of the averages would indicate that Glove is more discriminative in distinguishing semantic similarity. However, a closer, manual examination reveals a

[3]https://pandas.pydata.org/

higher volatility with Glove. Glove's measures contain significantly higher high values and lower low values.

The precision values are illustrated in Figure 5.25, for the Euclidean distance, and Figure 5.26, for the cosine similarity.

When evaluating the precision charts we want to emphasize that the two thresholds are not equal for each model (i.e., the thresholds do not have the same scale). A theoretically ideal threshold for the Euclidean distance with Glove would be 38.20, which represents the middle distance between the average duplicate (i.e., 24.13) and the average non-duplicate (i.e., 52.37). However, the Cerberus approach does not measure any positive examples at that distance, since its own halfway point between the averages is at 17.3, and thus, classifies 0 out of 44 duplicates correctly.

Perhaps the fairest comparison using Euclidean distance is with two thresholds, one drawn at each theoretically ideal value for each method. Using these thresholds, Glove's precision is at 24%, significantly lower than Cerberus's precision at 53%. This large discrepancy in favor of Cerberus may be largely due to Cerberus reducing the volatility of the glove vectors. Even though it looks like Cerberus is doing a better job, coser examination is needed.Glove recalls 36/44 duplicate pairs and Cerberus only 20/44 duplicate pairs. Cerberus has a higher precision because it has lower volatility in its outputs, and the dataset is highly unbalanced.

Another observation we want to make is that due to the dataset being imbalanced (i.e., 44 duplicate pairs and 25,156 non-duplicate pairs) a general reduction in volatility could result in a general bias toward guessing negative to be more consistent, and make a method look more successful than is meaningful. That said, because this is a measure that is meaningful both for duplicates (semantic closeness) and for non-duplicates (semantically distant), being able to identify negative examples

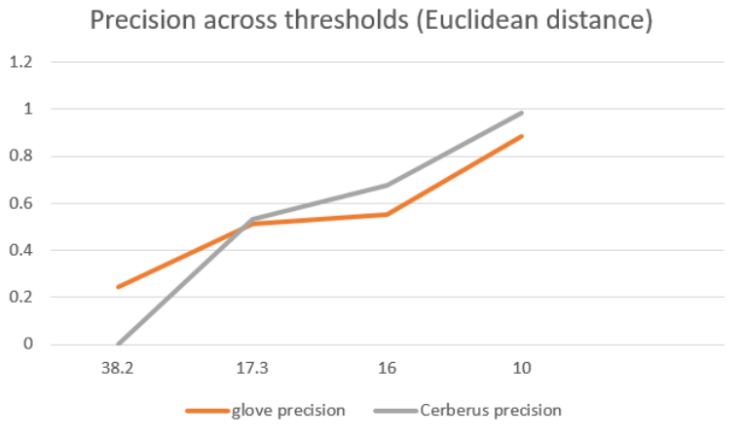


Figure 5.25: Precision (y-axis) for Glove and Cerberus across different thresholds (x-axis), using the Euclidean distance

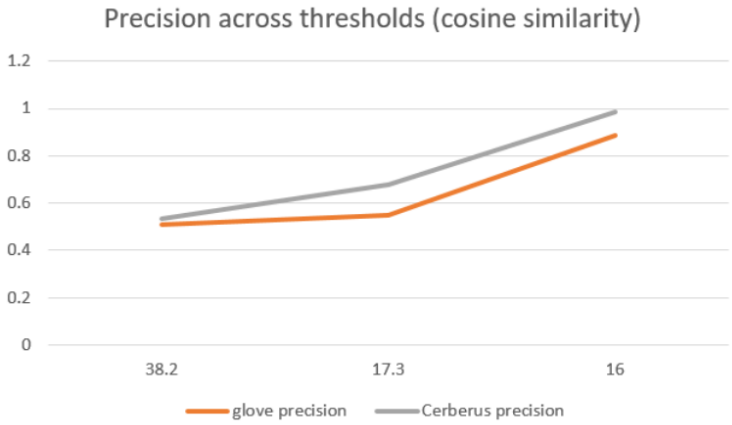


Figure 5.26: Precision (y-axis) for Glove and Cerberus across different thresholds (x-axis), using the cosine similarity

is equally important to the interpretation. In other words, most non-duplicate bugs should be semantically disparate and any methodology seeking to measure semantic similarity should not only be able to identify semantically close items, but should also be able to identify semantically dissimilar ones. It is of significant importance to note that the models are in no way trained on their ability to distinguish between duplicate and non-duplicate pairs. Therefore, the majority of concerns and problems

that arise with an imbalanced dataset are not applicable here. The models cannot learn to simply guess negatively all the time because they are not learning based on the outcome. The total precision can be misleading as it will be more reflective of how good the method is at predicting negative examples, but that is remedied by analyzing at the precision of the positive examples alone, as we did.

When using cosine similarity (see Table 5.5 and Figure 5.26), Glove outperforms Cerberus by a large margin across the board. Glove should be seen as a clear winner and supports the conjecture that most of the predictive power of Cerberus is influenced by one factor, namely Glove. It appears that the majority of promising results (see Chapter 5.1 and Chapter 5.2.2) were a direct result of Glove, and the current version of Cerberus (though it does provide a reduction in volatility and shows promising results by some measures) is hamstrung by the generative neural network not yet performing as needed in order to be useful and contribute to the entire Cerberus approach.

CHAPTER 6

LIMITATIONS AND THREATS TO VALIDITY

One of the main threats to the validity of the exploratory investigation presented in this thesis is the behavior of the models when generating images based on texts that were in their own training data. It is intuitive, and a well-known fact in the realm of generative networks, that a generative neural network training on a type of data should produce images that are similar to the training images when fed training text. After all, that is the whole point of the training, and the primary trait that the loss function should represent. However, the GNNs of Cerberus, when trained at 1,000 epochs did not perform in this manner (see Chapter 5.2.2).

If the models were working in a manner similar to other related work, such as Stanford's work on scene graphs [10], what would the results of looking for semantic similarity throughout the process have looked like? This is a conversation left for future work.

Another threat to the behavior of the models was the lack of differentiation in the images generated. Even when fed semantically different words, and thus, very different initial vectors (e.g., "red" and a blank text), the resultant images looked very similar. Differentiation among results appeared to be much more linked to the total number of words in the source text, and the number of those words that appear in the training data. Additionally, these characteristics are not consistent across all

our experiments. For example, in the Coco 10 epochs model, the images produced from the training data had three pictures that were different from the rest (see the 4^{th}, 7^{th}, and 11^{th} pictures in Figure 5.20). The reason why only these three pictures would differ is not immediately apparent.

Additional threats to validity include the choice of the datasets, the choice of the neural networks, the configuration, tuning or parameterization of the neural networks, including the number of layers. All these factors could have significantly influenced the results.

To place this work in proper context it should also be mentioned that this thesis was proposed as an exploratory study, full of novel approaches that stretched and grew the knowledge base of the research community.

CHAPTER 7

CONCLUSIONS

The main takeaway of the work presented in this thesis is the preliminary evidence that semantics can be preserved through the process of generating images from text via a generative neural network. Our exploratory study illustrates that even a largely randomized GNN constructing images will produce content that likely preserves semantics to a degree. If an appropriate model can be built to better preserve and represent those semantics, especially if more data and more training can lead to better representation and preservation of semantics, then many potential applications in the areas of IR, NLP and AI await.

Additionally, the outcome of this work suggests that a mapping from a summation, or average, of word vectors to an image is a rather complex step, which depends largely on the depth of the network, the number of training examples, and fine-tuning or manipulation of the network.

Given the number of processes, variables and factors involved, and lack of strong evidence in results produced by Cerberus, make it difficult to draw definitive conclusions about the results. There are three different neural networks included in Cerberus, each built on many levels of abstraction. Even considering the issues of the "yellow" image output (see Chapter 5.2.1), and the "black" image output (see Chapter 5.2.2), appearing as such strong evidence that the base model is broken, we

cannot know that the model would not have worked better with modifications made to the number of layers in the network, more data, or other changes to variables. The entire recent success of deep learning is due to the observation that even though many of these models did not work well, as the models approached very large datasets and very high levels of training, the trajectory of the results was toward improvement. Consequently, investments were made in data and processing to the point that very deep networks were possible, and their performance has changed the landscape of AI.

In the world of AI and data science, much of the fine-tuning and model modification can be reduced to just changing variables at random and shifting data until the results look better, because models are treated as black boxes. The sheer number of difficult to see and understand variables and properties can lead to modes of thinking and approaches that seem almost magical or superstitious. It can be easy to assign credit or blame to specific parts of a system, or the system at large, without complete knowledge of all the variables and their intricate interactions.

The fact that less trained GNNs produce results with easier to detect semantic information in the Primary experiments (see Chapter 5.2) suggests that the Glove training alone is doing the majority of the work in semantic encoding, and Chapter 5.3 provides strong evidence that Glove alone produces sufficient semantic distinction that the encoding performed by Cerberus is due to its work.

Some have treated the dark, and not fully understood corners of deep learning and neural networks as holding an almost magical promise, and believe that these systems are close to completely solving language or vision. It is the opinion of the author that the pursuit of "magic" in the neural network will likely continue to be a fruitless search. It is true that these systems are big and complex, and there are many properties that are not fully understood. It should be reiterated that this fact

was a major reason this exploratory thesis was undertaken. However, these systems are ultimately limited by what they are, mathematically functional mappings of an input to an output.

Considering the brain as a similar system, neuroscientists do not consider the brain to be completely explained. There are likely a number of significant differences in the brain that are beyond the borders and limitations of even the deepest neural networks of today and the near future. If a similar structure is needed for general AI to behave in a similar fashion, then even the deepest and most well trained neural networks may not be able to get us there.

CHAPTER 8

FUTURE WORK

On top of the foundation that semantics can likely be preserved through the generative process, many unanswered questions still remain. The strongest of these is: What would the results of the Primary experiment be on a model that is behaving as expected?

It is uncommon to finalize a thesis where the primary future work is to restructure and perform the central experiment once more. That scenario may even be considered a failure, but such is the case for early work in investigating unexplored aspects of an area. It was a risk that was knowingly engaged with. However, it is worth noting that there is also a growing movement in the research community to publish negative results [18, 22]. If the results of such work show any promise, there are many potential options to improve the system. Beyond the obvious fine-tuning and modifications of the neural networks to obtain better results, it would be worth investigating if there was a practical way to train the system as a whole. This would allow optimization to distribute to the whole, and not localize to particular subsections that that could potentially be working against each other at different stages of Cerberus.

Another option is to simplify the model to the word level, instead of working at the sentence level. This could reduce the complexity of the search space, facilitating making observations, adjustments, and drawing conclusions. Much of this work

suffered from the inherent complexity in each of the many moving parts of the central experiment. Any simplification could be beneficial, and reducing the word space may be the most direct way to do it.

It should also be noted that many of the initial driving questions remain, and could provide useful direction for future work: How would results differ on more natural language, like a dataset for duplicate questions? How much could the results improve if the similarity measure network was trained or refined on data that is similar to the test data? Are the similarity measures indicative of just word similarity, and not as much order? Are there ways to better understand and control what the generative network is sensitive to and align that with differences that are important in the target space? If the process did specialize well when focused and tuned for a specific domain (like many neural networks do), could it generalize well? Could the image generator be trained on all English? Could the similarity network be general enough to work on anything? What are the bounds of the methodology? How long can the input texts be? How semantically dense can the text be? Would a well-developed model follow the trend and result in a powerful tool that has great results when properly tuned and in ideal circumstances, but lacks easy general application, and lacks the visibility and transparency of the process to allow for large-scale trust and implementation, or significant clear insight? Could a generative network be trained for this type of task specifically? What is a good picture of a bug report? Is it possible to use the whole model in concert to train pieces of it? Could this create a network specialized at making pictures that represent semantic similarity well?

There is no shortage of future work to be done in this area. It is the hope of the author that his work will provide a useful early step of knowledge and experience that will be an aid to other researchers that wish to explore this space.

REFERENCES

[1] K. Aberman, J. Liao, M. Shi, D. Lischinski, B. Chen, and D. Cohen-Or. Neural best-buddies: Sparse cross-domain correspondence. *CoRR*, abs/1805.04140, 2018.

[2] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, Oct 2002.

[3] R. Caruana, S. Lawrence, and L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 381–387, Cambridge, MA, USA, 2000. MIT Press.

[4] R. Caruana, S. Lawrence, and C. Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. volume 13, pages 402–408, 01 2000.

[5] Eclipse. Eclipse open source development project bug dataset. `https://github.com/ansymo/msr2013-bug_dataset`, 2013. Accessed: 8-April-2019.

[6] J. Gao, , M. Gamon, and and. Modeling interestingness with deep neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October 2014.

[7] S. Haykin. *Neural Networks: a Comprehensive Foundation*. Prentice Hall PTR, 1994.

[8] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[9] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc., 2014.

[10] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. *CoRR*, abs/1804.01622, 2018.

[11] P. Jupyter. Jupyter notebook - open-source software, open-standards, and services for interactive computing. `https://jupyter.org/`. Accessed: February 12, 2019.

[12] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.

[13] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[15] Z. Lu and H. Li. A deep architecture for matching short texts. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1367–1375. Curran Associates, Inc., 2013.

[16] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[17] G. Marcus. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631, 2018.

[18] N. Matosin, E. Frank, M. Engel, J. S. Lum, and K. A. Newell. Negativity towards negative results: a discussion of the disconnect between scientific worth and scientific culture, 2014.

[19] T. Mikolov. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[22] A. Mlinarić, M. Horvat, and V. Šupak Smolčić. Dealing with the positive publication bias: Why you should really publish your negative results. *Biochemia medica: Biochemia medica*, 27(3):1–6, 2017.

[23] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, Jan. 2008.

[24] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. *CoRR*, abs/1602.06359, 2016.

[25] S. D. S. M. Patrick Buehler. Image similarity ranking using microsoft cognitive toolkit (cntk). `https://github.com/Azure/ImageSimilarityUsingCNTK`, 2016. Accessed: 24-August-2018.

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

[27] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[28] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, Mar. 1997.

[29] C. K. Roy, J. R. Cordy, and R. Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Sci. Comput. Program.*, 74(7):470–495, May 2009.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of research. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[31] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[32] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[33] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110, New York, NY, USA, 2014. ACM.

[34] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 461–470, May 2008.

[35] R. D. Xianshun Chen. Keras-text-to-image. `https://github.com/chen0040`, `https://github.com/chen0040/keras-text-to-image`, 2018. Accessed: 14-August-2018.

[36] XnSoft. Xnview - image organizer and processor. `https://www.xnview.com/`. Accessed: February 12, 2019.

[37] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[38] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *CoRR*, abs/1612.03242, 2016.

APPENDIX A

PRIMARY EXPERIMENTS TECHNICAL DETAILS

This appendix enumerates a list of technical details for the Primary Experiments described in Chapter 5.2.

- An Anaconda[1] environment was set up that could run the generative neural network training, generation, and CNTK similarity processing.

- The same base model that was used for Preliminary Experiments 5.1 was used in the Primary Experiments 5.2, and the Keras[2] layers were modified to accept the training data.

- Keras uses Tensorflow[3] and training was done via a Graphics Processing Unit (GPU) on an Nvidia GEFORCE GTX 1080 Ti graphic card.

- A built-in Windows image resizer was used for altering the sizes of images.

- XnView[4] was used for converting images from *.png* to *.jpg* format, or vice versa.

- When pixel rations of images needed to change, cropping was done to preserve the center of the image

- Training image sizes were reduced to 100x100 via XnView in order to accommodate memory restrictions.

[1]https://www.anaconda.com/
[2]https://keras.io/
[3]https://www.tensorflow.org/
[4]https://www.xnview.com/

- The code bases were modified to work with said dimensions and constraints and researchers interested in reproducing or building on results should contact the author if they want notes of these changes or access to the modified code used in the experiments.

- CNTK feature output of matrices was stored as serialized pickle files and loaded into Jupyter Notebooks [11] for examination and comparison.