# Automatic Poetry Generation

Research Proposal


**STUDENT NAME:** Ryan Moreno

**COURSE NAME:** WRIT 340

**PROFFESSOR:** Harley Ramsey


**ASSIGNMENT:** Research Proposal

**DATE OF SUBMISSION:** 05 December 2018

# CONTENTS


**ABSTRACT** **3**

**BACKGROUND** **3**

I. Poetry Generation 3

A. Racter 3

B. ASPERA 4

C. McGonagall 5

II. Deep Learning 6

A. Latent Space 7

B. Variational Autoencoders 8

C. Generative Adversarial Networks 9

D. Combining VAEs and GANs 10

E. Conditioning Generation 12

F. Multifaceted Feature Visualization 13

**PROPOSAL** **14**

**REFERENCES** **17**

# ABSTRACT

My research will investigate the possibility of automatically generated poetry. There has been minimal research in this area, and the research that has been done doesn't take advantage of advanced computational techniques. The poetry generation programs that have been written are mostly rule-based. This is problematic because no one actually knows the steps necessary to create something as qualitative as poetry.

Recently there have been significant advances in creative generation in other fields, especially image generation. Specifically, deep learning has proven useful because of its ability to encapsulate abstract ideas and processes. In this way, generation can occur without a pre-designed process and even without a pre-specified goal. In this research proposal I will present past research on poetry generation, past research on creative generation using deep learning, and my plan to combine the best of these techniques towards automatic poetry generation.

# BACKGROUND

## I. Poetry Generation

While there has been substantial research on natural language processing, research on text generation has mostly been limited to practical communication such as voice assistants. Creative text generation is more difficult because there isn't a pre-specified communication goal, and the text has to have literary value beyond mere coherency. In this section I will present three past attempts at poetry generation. The first two attempts are purely rule-based. These fall short because they depend on a programmer to specify the steps necessary to write poetry. The third attempt I present uses machine learning, but doesn't use deep learning. This also falls short because, although it doesn't require a programmer to specify the steps to write poetry, it still requires a programmer to specify the qualities of a poem.

### A. Racter

The earliest attempt I found at automatic poetry generation was a program called Racter. This was written by writer-programmer duo William Chamberlain and Thomas Etter in 1983 [1]. Unlike most artificial intelligence, Racter doesn't try to mimic human thought; Racter is a completely rule-based program. This means that there is no use of advanced computational technology such as machine learning or deep learning. Based on these rules, Racter is able to follow basic grammatical necessities. For example, Racter knows how to properly conjugate verbs and track pronouns with their corresponding nouns [1]. Racter also has more sophisticated rules allowing for the development of specific poetic forms [1]. In order to mimic some level of coherency, Racter uses a dictionary file with each word having a 12-character identifier indicating its definition [1]. In this way, words with similar meanings, like sleep and dream, end up in the same sentence. Racter's poeticness largely comes from its use of repetition. By

repeating words, phrases, and sentence structures throughout a poem, Racter mimics poetic techniques [2]. Because of this, at a first glance some of Racter's poetry actually seems fairly realistic. The following is an example of one of Racter's more convincing poems:

> A lion roars and a dog barks. It is interesting
> and fascinating that a bird will fly and not
> roar or bark. Enthralling stories about animals
> are in my dreams and I will sing them all if I
> am not exhausted and weary.

Although Racter's generated texts have some qualities of poetry, they lack coherency at a large scale. The individual sentences are grammatically correct because of Racter's grammar rules, and the individual images even make sense because Racter combines words with similar definitions. However, the entire poem lacks cohesion. One area that Racter does excel in is its unique imagery. Perhaps because it approaches text generation in such a different way from humans, it combines words to create interesting and unique phrases. Racter is effective because its "odd oxymorons and quirky metaphors can appear as beat profundity" [1]. One goal of poetry is to describe the ordinary with the strange. Because oddity and quirkiness are a characteristic of poetry, poetry is a good literary form for Racter, and a good starting place for creative text generation research.

### B. ASPERA

Like Racter, ASPERA is a rule-based poetry generator. It was developed by Gervas in 2001 for the creation of Spanish poetry. Although ASPERA and Racter are both rule-based, ASPERA approaches the problem of poetry generation in a very different way. ASPERA requires the user to input an intended message, length, and mood [3]. Based on this information, it chooses a poetic form. ASPERA avoids having to understand the complicated rules of grammar by selecting an existing poem of the chosen form and replacing individual words. Each word is marked with information such as its part of speech, number of syllables, and rhyme if necessary [3]. ASPERA assigns each line a portion of the message it is intended to convey. It then systematically replaces the words of the original poem with words matching the user's intended message [3]. The following is an example of one of ASPERA's poems and its translation.

| | |
|---|---|
| Ladrará la verdad el viento airado | The angry wind will bark the truth |
| en tal corazón por una planta dulce | in such a heart for a sweet plant |
| al arbusto que volais mudo o helado. | to the bush that you are silent or frozen. |

While the poem follows grammatical rules, it is even less coherent than Racter's poetry. An additional limitation is that ASPERA requires user input, which isn't ideal for automatic poetry generation. With this in mind, ASPERA's approach to poetry creation is clearly lacking.

### C. McGonagall

McGonagall is the only poetry generation program I found that uses advanced computational techniques. McGonagall uses evolutionary algorithms to create poetry [4]. In order to use evolutionary algorithms, one must specify what qualities the candidate solutions will be evaluated on. The creators of McGonagall chose to use meaningfulness, grammaticality, and poeticness as the goals [4]. Meaningfulness was defined as the similarity between the meaning of McGonagall's chosen words and the words of a user's inputted intended message. Poeticness was defined as McGonagall's ability to follow a specific poetic form and stress pattern. Grammaticality is simply McGonagall's ability to follow grammatical rules. Because proper grammar is a necessity, all evolvable solutions are required to uphold this constraint. On the other hand, meaningfulness and poeticness are imposed using penalties, meaning that the more a poem fulfills these constraints, the higher it will be ranked [4]. Valid grammar is upheld by using pre-specified grammar operators to evolve the candidate solutions.

Evolutionary algorithms go through two main phases: evaluation and evolution. The process begins with a baseline of randomly-generated candidate solutions. These candidate solutions are evaluated, and the ones with the highest fitness are chosen to be evolved. The chosen solutions are evolved, either by crossover (swapping parts with another solution) or by mutation (randomly changing parts of its state) [4]. These mutations result in a new generation of candidate solutions, and the process repeats.

Although McGonagall's approach to poetry generation is significantly more sophisticated than Racter's or ASPERA's, its results are not particularly impressive. When evolving for both meaningfulness and poeticness, McGonagall was able to come close to proper meter and semantic similarity to an intended message [4]. However, McGonagall was never completely successful in these endeavors and its outputted poetry didn't make a lot of sense. Additionally, McGonagall is limited because it requires a user's intended message. The following poems is one of the more successful results of McGonagall's approach to poetry generation. The program was asked to write a haiku with similar semantics to the lines on the left. McGonagall's poem is on the right.

| | |
|---|---|
| The African lion, he dwells in the waste,<br>he has a big head and a very small<br>waist. | In a waste, a lion,<br>who has a very small waist,<br>dwells in a big head. |

The benefit of evolutionary algorithms comes from the distinction between genotype and phenotype [4]. In this case, the genotype is the text of a poem while the phenotype is its meaningfulness, poeticness, and grammaticality. This distinction allows the evolutionary algorithm to abstract away the steps necessary to make a poem meet these criteria. Since there aren't known algorithms to create poetry, the best approach isn't rule-based, but rather discriminative, meaning that the model generates solutions and tests them for viability before

accepting them [4]. Evolutionary algorithms are a well-documented form of generate-and-test approaches, and they avoid getting stuck at local optima [4].

Through the use of evolutionary algorithms, McGonagall is a step towards more sophisticated poetry generation than rule-based generators. However, the creators of McGonagall still have to specify the ways in which a poem is to be evaluated. While meaningfulness and poeticness are certainly integral qualities in a poem, it's unclear how exactly these should be quantified. Also, there is more to a poem than just semantics and meter. However, it's unclear precisely what makes a poem or how the different qualities of a poem should be weighted in an evaluation equation. In this way, McGonagall is still limited by the necessity of a programmer hard-coding the qualities of a poem.

Unlike the qualitative aspects of a poem like meaningfulness and poeticness, grammar is possible to quantify. It would have been a waste of computational resources for McGonagall to include grammar as a penalty-imposed metric [4]. Thanks to extensive natural language processing research, we know the steps to ensure a text is grammatical, so it is beneficial to take advantage of this knowledge and tell the evolutionary algorithm how to keep the poem grammatical rather than asking it to create its own steps through evolution. Additionally, McGonagall was more successful in meeting its goals of meaningfulness, poeticness, and grammaticality when it was required to uphold complete grammar at every step of evolution [4]. This provides evidence that explicitly specified grammar rules are beneficial.

## II. Deep Learning

The approaches to poetry generation I presented are lacking because they require a programmer to specify knowledge about poetry that we don't have. These approaches require specification of the steps necessary to create poetry, or at least a quantitative way to evaluate poetry. Deep learning offers a way to avoid these requirements.

Recently there has been substantial research on creative generation using deep learning. Much of this research has been on image generation because there are already deep neural networks (DNNs) that are good at categorizing images. For example, the Innovation Engine was created in 2015. Like McGonagall, the Innovation Engine uses evolutionary algorithms to create candidate solutions. However, the Innovation Engine uses DNNs (a form of deep learning) to evaluate the candidate solutions at every step [5]. DNNs train on large data sets in order to learn how to categorize images. In this way, no programmer has to specify how to evaluate the images; the DNNs learn on their own. In order to test the Innovation Engine, some of its creations were submitted to an art contest at the University of Wyoming. They were not only accepted, but given an award, supporting the use of DNNs to generate creative, interesting images [5]. Based on this success, I investigated different approaches to generation using deep learning. I present these deep learning techniques in this section.

### A. Latent Space

Deep learning depends on latent space, a compact and abstract representation of an object. This is most easily understood in the case of images. We usually consider images in their pixel space, that is the representation of them according to the location and color of their pixels. If we look at handwritten numbers, 8's and 3's are close together in the pixel space because they share many pixels. However, if we consider their representations in latent space, 8's and 3's begin to cluster separately because they are completely different from a semantic standpoint [6]. Because latent space is a compressed representation compared to pixel space, the latent space doesn't have room to represent images that don't correspond to something similar to a hand-written number; it must learn how to represent the relevant features efficiently [6]. This causes the latent space to encode the images based on meaningful, abstract concepts such as what number is written.

Within this latent space, all objects are semantically meaningful. Once you've accessed this latent space it's possible to generate new, valid objects in several ways. The simplest way is by walking around values in the latent space. Returning to the handwritten numbers example, if you sampled random values in the pixel space, most would be random static. However, because the latent space is a compressed representation and is semantically meaningful, if you remain in the latent space, you find objects that are meaningful. One way to explore latent space is through interpolation, varying the latent vector between two objects [7]. Interpolating between a 5 and 9 in pixel space results in intermediate steps that are invalid, that look like a 5 and a 9 were written over each other. However, interpolating between a 5 and 9 in latent space results in intermediate steps that are valid, that look like new handwritten numbers (see Figure 1).

**Fig 1.** Interpolating between handwritten numbers in pixel space (left) and latent space (right) [6]

Another way to way to explore the latent space is to augment the latent vector dimensions individually, showing what semantic concept each dimension represents. For example, the latent space of 3D chair models was shown to have dimensions representing the thickness of different parts of the chair [7].

A more complicated way to generate objects from the latent space is through latent vector arithmetic [7]. Subtracting two latent vectors results in a vector that encodes the difference between the two objects. See Figure 2 for an example of latent arithmetic on 3D chair models. When a chair without arms is subtracted from a chair with arms, this results in a vector representing the abstract concept of an arm. When this vector is added to other styles of chairs without arms, it results in the other styles of chairs with arms.

**Fig 2.** Latent space arithmetic on 3D chair models [7]

### B. Variational Autoencoders

As seen in the previous section, once we have access to latent space, we can explore that space to generate new objects. The next question is how to create the latent space itself and how to decode latent space objects into data space objects. This is where deep learning comes in. Variational autoencoders (VAEs) are made of two neural networks. One is trained to encode a data space into a more compressed latent space, and the other is trained to decode the latent space back into the data space [8]. See Figure 3 for the structure of a VAE. By training the encoding and decoding network in tandem, a latent space is created. This latent space can be explored and decoded into valid data space objects.

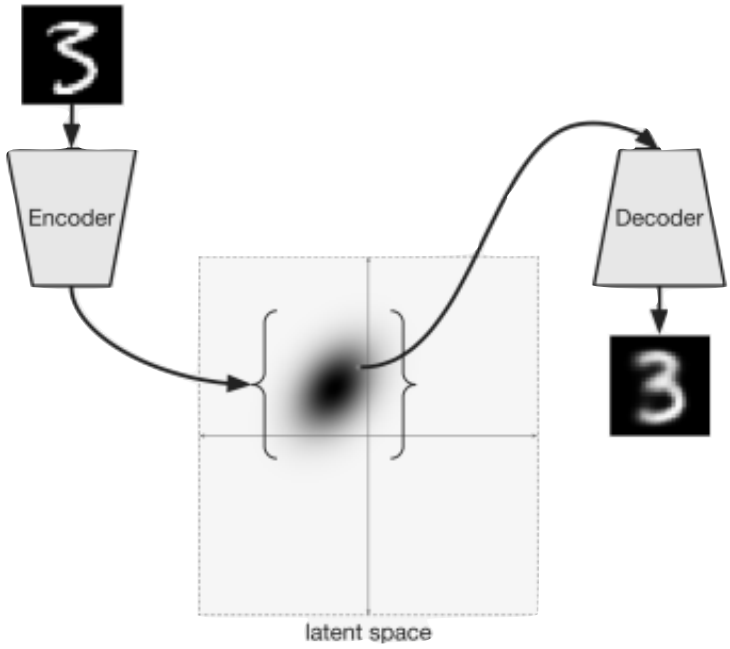


**Fig 3.** Structure of a variational autoencoder
http://ijdykeman.github.io/ml/2016/12/21/cvae.html

The encoder and decoder neural networks are trained using stochastic gradient descent, rewarding the networks when decoding of the latent space results in objects in the data space that are similar to the original data set [8]. This maximizes the probability of a given point in the latent space corresponding to an original data point [8]. The maximum likelihood principle says that, "if the model is likely to produce training set samples, then it is also likely to produce

similar samples, and unlikely to produce dissimilar ones" [8]. Because of this principle, training the neural networks to reproduce objects from the original data set results in a latent space that can also generate objects that are novel, but still similar to the original data set and therefore semantically meaningful. Through neural network training, the encoder learns to model the abstract concepts of these objects within the latent structure without the programmer needing to specify how to do so [8].

A related use of VAE is part completion, where an object with missing parts is provided and the VAE attempts to complete the part. Nash and Williams showed this to be successful in the completion of 3D models. They began with 3D models of planes, bikes, and chairs from which they removed individual parts [9]. They then asked the VAE to complete the models. They found that the VAE was successful in creating complete, semantically meaningful objects that differed from the originals [9]. See Figure 4 for examples of their part completion results.

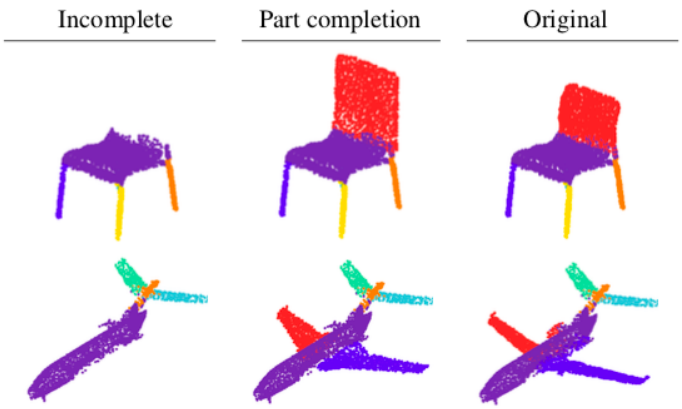


**Fig 4.** Using variational autoencoders to complete partial objects [9]

### C. Generative Adversarial Networks

Another deep learning approach to object generation is generative adversarial networks (GANs). GANs are made up of two neural networks that are trained in competition. The first neural network is generative and creates new objects. The second network is discriminative and tries to guess if a given object is created by the generator or is part of the training set [10]. See Figure 5 for the structure of a GAN. These two neural networks can be thought of as a counterfeiter and a cop. The generator network is like the counterfeiter, creating fake objects, while the discriminative network is like the cop, trying to determine which objects are fake [10].

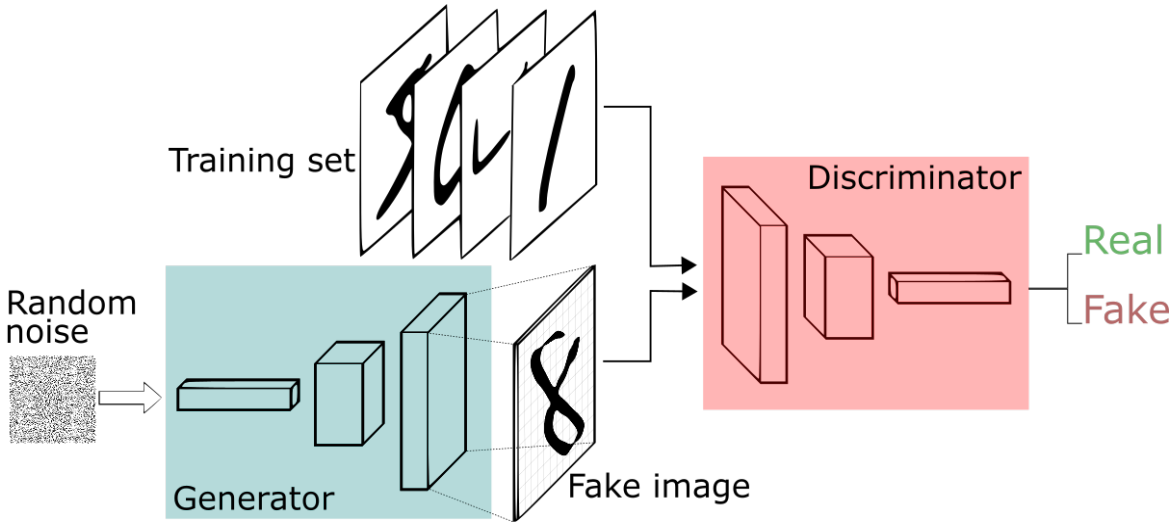


**Fig 5.** Generative adversarial network diagram
https://skymind.ai/wiki/generative-adversarial-network-gan

GANs are a prominent technique used in creative generation. They have proven themselves viable generators for music, images, and 3D models among other fields [9, 11, 12]. Figure 6 shows ten face images generated by a GAN. The faces highlighted in yellow are the faces from the training set most similar to their neighbors. The fact that the generated images are not identical to any of the training set images demonstrates the GAN's ability to generate novel objects. The fact that the generated images are recognizable as faces demonstrates the GAN's ability to learn the abstract concepts behind the faces and create semantically meaningful images.

**Fig 6.** Results of face generation using a GAN [12]

### D. Combining VAEs and GANs

Variational autoencoders and generative adversarial networks have both proven themselves successful at creative generation, and some research has gone into combining the two. For example, Larsen researched generating face images using a VAE/GAN combination. As depicted in Figure 7, the decoder network of the VAE is used as the generator for the GAN and the discriminator network of the GAN is used as the similarity metric for the VAE [13].

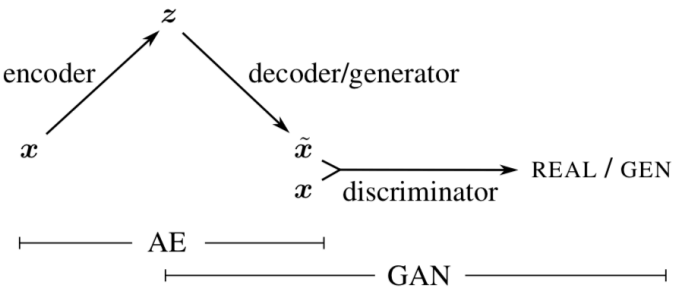


**Fig 7.** Diagram of combined VAE/GAN [13]

Combining the VAE and GAN benefits both models. The combination benefits the GAN because the generator starts from something meaningful (a position in latent space) rather than random noise. More substantially, the combination benefits the VAE by providing a similarity metric based on a neural network (the GAN's discriminator) rather than a simple pixel distance equation [13]. As previously described in the section on latent space, deep learning provides an understanding of abstract ideas (like the presence of eyes and noses in face images) which pixel distance does not. To understand the significance of this, consider an image A and a different image B which is image A shifted a few pixels to the left. A good similarity metric would say that image A and B are very similar [13]. The GAN's discriminator would be able to say that image A and B are similar because it understands the images at a semantic level. On the other hand, pixel distance might say that they are substantially different because the pixels don't line up. Because of this, VAEs trained with a pixel distance similarity metric often end up with fuzzy results [13]. Figure 8 shows the results of Larsen's research. The faces generated solely from the VAE are significantly fuzzier than those generated from the GAN or VAE/GAN combination. The GAN and VAE/GAN result in similar quality faces images. The GAN faces seem to combine feminine and masculine features on the same face more than the VAE/GAN, but the differences are not significant. The success of the VAE/GAN combination supports combining the two frameworks as a viable technique.

**Fig 8.** Face generation using a VAE (top), a GAN (middle), and a VAE/GAN combination (bottom) [13]

### E. Conditioning Generation

The generative adversarial network model can be adjusted to be more robust. One way to enhance GANs is to condition the generator, influencing the objects that the generator creates. The goal of conditioning is to have the generator create novel objects particularly similar to the condition object. Conditioning occurs after the GAN is trained. A conditioner network is created with neural layers opposite the generator network in structure [11]. As depicted in Figure 9, the condition objects are deconstructed and hooked up to the generator's layers in order to influence the objects generated [11]. This setup allows the level of creativity versus similarity to the condition object to be be adjusted by connecting the condition object to more or fewer layers in the generator's neural network [11].

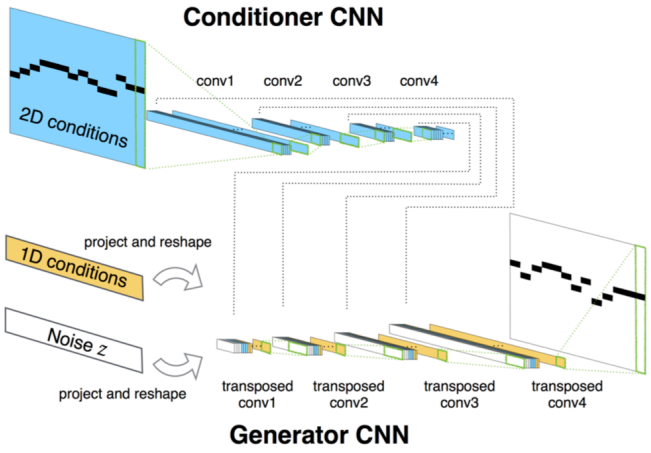


**Fig 9.** Diagram of a conditioned generative neural network [11]

In Yang's research, conditioned generators are used to create music. Yang conditions the generator in two different ways. In the first model, each bar of music is conditioned with the previous bar of music by connecting the condition object to all four layers of the generator's neural network [11]. In the second model, each bar of music is conditioned with its own chord progression by connecting the condition object to all four layers of the generator's neural network. In the second model, each bar is also conditioned with the previous bar. However, the condition object of the previous bar is only connected to one layer so that the chord condition dominates [11]. Figure 10 shows the results of Yang's study. MidiNet 1 is the first model, conditioned on the previous bar, and MidiNet 2 is the second model, conditioned on the current bar's chord as well as the previous bar. The other three models are previous well-researched music generators that don't use deep neural networks.

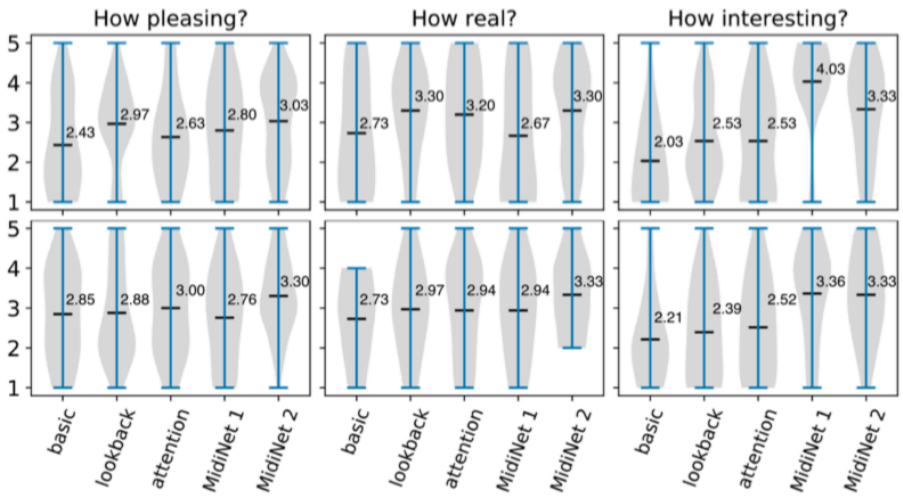


**Fig 10.** Results of music generators as rated by people with musical backgrounds (top) and people without musical backgrounds (bottom) [11]

The fact that both MidiNet models perform comparably or better than the other three models (especially in interestingness) supports the use of deep neural networks in creative generation. That MidiNet 2, which is conditioned twice, performs better than MidiNet 1, which only conditions once, in its pleasantness and realness supports the conditioning of generative networks. Yang's approach is successful because conditioning each bar with a chord progression makes the generated music more realistic, and conditioning each bar on the previous bar adds a sense of continuity throughout the piece.

### F. Multifaceted Feature Visualization

The section on latent space discusses various ways of exploring the latent space to investigate how generator networks encode abstract concepts. This section will demonstrate how to inspect the structure of the discriminative network. A common way to investigate which nodes encode which features is through activation maximization. In this technique, objects are evolved to maximally set off a given node [14]. The evolved object is then qualitatively examined to determine what the given node has learned to look for. For example, in the image domain, one node in a discriminative network might fire in response to images that look like apples. Thus, the evolved image that maximally activates this node will look somewhat like an apple.

The problem with activation maximization is that neural networks are more complicated than this technique assumes. Rather than encoding one feature, each node represents an abstract concept which can contain multiple features. For example, one research paper discovered a "grocery" node within an image discriminative network [14]. This "grocery" node activated for various features pertaining to grocery stores, including rows of apples, rows of artichokes, and store fronts [14]. Because a node can encode multiple features, simple activation maximization techniques often results in images that inaccurately mix colors, object parts, and even different scales of images [14].

One approach to improving activation maximization is multifaceted feature visualization. With this technique, objects are evolved towards multiple features [14]. By evolving towards multiple images, multifaceted feature visualization is able to discover all the different features that make a given node fire instead of inappropriately mixing these features. Using multifaceted feature visualization, Nguyen and Yosinki were able to evolve images of the various features that set off the "grocery" node described above (see Figure 11).


**Fig 11.** Results of multifaceted feature visualization on a "grocery" node (left) and the corresponding training set images that activate the "grocery" node (right) [14]

# PROPOSAL

Deep learning has proven itself successful in applications of creative generation, particularly in the field of image generation. I would like to take these techniques and apply them to poetry generation. Poetry generation is a good starting place within the field of creative language generation because poetry is generally shorter than prose and has fewer constraints. My research proposal involves techniques from past research on deep learning applied to creative generation as well as past research on poetry generation.

I plan to use the VAE/GAN framework as the basis of my poetry generation system because combining the two techniques takes advantage of the benefits of each. The success of using VAE or GAN models separately for creative generation in other domains is supported extensively, and there is also some research supporting the combination of these frameworks [9, 11, 12, 13]. As training data, I intend to use online poetry databases. I will begin with single stanza poems of a simple form in modern English. By starting with these strict limitations, my initial VAE/GAN framework is more likely to successfully encode poetry.

After the VAE/GAN framework has been trained, I plan to implement conditioning of the GAN generator. Unlike images, poetry has a starting point, ending point, and a path between that takes time to follow. Because of this, it is necessary for a poem to have a sense of continuity of theme and form. This is similar to the task of music generation, in which musical bars must retain a sense of continuity throughout a piece. Therefore, I plan to use conditioning techniques similar to

those implemented in the MidiNet model to generate music [11]. Rather than conditioning musical bars on past musical bars, I plan to condition poetic stanzas on past poetic stanzas. By creating one stanza at a time, the VAE/GAN framework will only have to produce small chunks of text, increasing the likelihood of success. Using conditioning techniques, each stanza will make sense in combination with previously created stanzas despite being generated separately. The research on MidiNet also demonstrates the feasibility of conditioning on multiple condition objects with different proportions. I plan to condition a given stanza most heavily on the preceding stanza by connecting the previous stanza's condition object to the most network layers. I can also condition on older stanzas with a lower proportion by connecting their condition objects to fewer layers.

I would also like to research the technique of part completion used in the generation of 3D objects [9]. It's possible that starting the generation process with part of a poem already complete would help the VAE/GAN framework create a plausible poem. If part completion is successful starting with half of a real poem, I would like to try running a second round of part completion starting with the half of a poem created during the first round. In this way, the VAE/GAN framework will benefit in both rounds from having half of the poem already written, but will end up creating an entirely new poem.

Finally, if my poetry generator is able to create anything resembling poetry, I would like to investigate how the generator network has abstracted the concepts of what makes poetry. To do so, I will use various techniques of feature visualization. To examine the generator network, I will explore the latent space of the VAE using the techniques described in the latent space section. By manipulating the vector dimensions and qualitatively observing how the poetry changes, I will get an idea of what abstract ideas the various dimensions encode. Similarly, I can observe latent space arithmetic on poems and the interpolation between poems. To examine the discriminative network, I plan to use activation maximization with multifaceted feature visualization as described in the section on multifaceted feature visualization. Because the abstract features that make up poetry are complicated, it is likely that the nodes in the discriminative network will encode multiple features, so using multifaceted feature visualization will give a more robust description of what concepts the nodes encode.

Because I plan to take a deep learning approach to poetry generation, much of the past rule-based research in this field will not be applicable. However, one crucial takeaway from past research on poetry generation is the hard-coding of grammar constraints. Because it is necessary for the grammar of a poem to be completely correct and because grammar can be modeled as a set of rules, I plan to impose grammar in a rule-based approach. There is no reason to rely on a deep neural network to learn the abstract concepts of grammar; hard-coding grammar rules takes advantage of past research in natural language processing and avoids wasting computational resources implementing grammar through learned networks. The grammar rules will be imposed on the decoder of the VAE (which is also the generator of the GAN). This way, as the VAE/GAN framework is trained, it will always be required to maintain grammatical validity.

My research hopes to use deep learning techniques, which have largely been used within the image generation domain, for language generation. By combining aspects of natural language processing and poetry generation techniques with deep learning and creative generation techniques, I aim to demonstrate the feasibility of deep learning as a poetry generation tool.

# REFERENCES


[1] J. Ledbetter, “RACTER, the poetic computer.” *The New Republic*, vol. 195, no. 6, Aug 1986.

[2] W. Chamberlain and J. Hall, *The Policeman’s Beard is Half Constructed*, New York City, Country: Warner Books, Inc., 1984. https://ubutext.memoryoftheworld.org/racter/racter_policemansbeard.pdf

[3] P. Gervas. “An Expert System for the Composition of Formal Spanish Poetry.” *Knowledge Based Systems*, vol. 14, no. 3, pp. 181-188, 2001.

[4] H. Manurung, “An evolutionary algorithm approach to poetry generation.” *University of Edinburgh, College of Science and Engineering, School of Informatics,* Jul, 2007. https://www.era.lib.ed.ac.uk/handle/1842/314?show=full

[5] A. Nguyen et al, “Innovation Engines: Automated Creativity and Improved Stochastic Optimization via Deep Learning” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2015.

[6] J. Despois, *Latent space visualization - deep learning bits #2*, Feb, 2017. [Online] https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

[7] J. Wu et al, “Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling.” Oct 2016. https://papers.nips.cc/paper/6096-learning-a-probabilistic-latent-space-of-object-shapes via-3d-generative-adversarial-modeling.pdf

[8] C. Doersch, “Tutorial on Variational Autoencoders.” *Carnegie Mellon/UC Berkeley*, Aug 2016. https://arxiv.org/pdf/1606.05908.pdf

[9] C. Nash and C. K. I. Williams, “The shape variational autoencoder: A deep generative model of part-segmented 3D objects,” in *Eurographics Symposium on Geometry Processing*, vol. 36, no. 5, 2017. https://homepages.inf.ed.ac.uk/ckiw/postscript/sgp2017.pdf

[10] I. J. Goodfellow et al, “Generative Adversarial Nets.” *Department d’informatique et de recherche operationnelle*, Montreal, Canada, Jun 2014. https://arxiv.org/abs/1406.2661

[11] L. Yang et al, “MidiNet: A Convolutional Generative Adversarial Network for Symbolic domain Music Generation.” *Research Center for IT innovation,* Taipei, Taiwan, July 2017.
https://arxiv.org/pdf/1703.10847.pdf

[12] J. Gauthier, “Conditional generative adversarial nets for convolutional face generation.” *Symbolic Systems Program, Natural Language Processing Group*, Stanford University, 2015.
http://cs231n.stanford.edu/reports/2015/pdfs/jgauthie_final_report.pdf

[13] A. Larsen et al. “Autoencoding beyond pixels using a learned similarity metric.” Feb 2016.
https://arxiv.org/pdf/1512.09300.pdf

[14] A. Nguyen, J. Yosinki, and J. Clune, “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned by Each Neuron in Deep Neural Networks.” Feb 2016.
http://www.evolvingai.org/files/mfv_icml_workshop_16.pdf