

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Preserving Semantic Structural Constraints within Neural Networks

Permalink

<https://escholarship.org/uc/item/192579jj>

Author

Le, Hubert Hoai

Publication Date

2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Preserving Semantic Structural Constraints within Neural Networks

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Hubert Hoai Le

Committee in charge:

Professor Zhuowen Tu, Chair
Professor Kamalika Chaudhuri
Professor Hao Su

2018

Copyright
Hubert Hoai Le, 2018
All rights reserved.

The thesis of Hubert Hoai Le is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2018

EPIGRAPH

*To deal with a fourteen-dimensional space, visualize a
three dimensional space and say fourteen.*

—Geoffrey Hinton

TABLE OF CONTENTS

Signature Page	iii
Epigraph	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Acknowledgements	ix
Abstract of the Thesis	x
Chapter 1 Introduction	1
Chapter 2 Background	3
2.1 Neural Networks	3
2.1.1 Activation Functions	4
2.1.2 Optimization	4
2.1.3 Generative Adversarial Networks	5
2.1.4 Wasserstein Generative Adversarial Networks	6
2.2 Learning Generative Models via Discriminative Approaches	6
2.2.1 Introspective Learning	8
2.3 Word Embeddings	10
2.3.1 Bag of Words	10
2.3.2 Term Frequency - Inverse Document Frequency	11
2.3.3 Word2Vec	12
Chapter 3 Related Work	13
3.1 Conditional Generative Adversarial Networks	13
3.1.1 Text to Image Synthesis	14
3.2 Two Branch Networks	15
3.2.1 Embedding Network	16
3.2.2 Similarity Network	17
Chapter 4 Methodology	19
4.1 Approaches	19
4.1.1 Introspective Image Synthesis	19
4.1.2 Wasserstein Introspective Image Synthesis	21
4.1.3 Affinity Network	21

Chapter 5	Experiments	24
Chapter 6	Conclusion	30
Bibliography	31

LIST OF FIGURES

Figure 2.1:	A typical feedforward neural network.	3
Figure 2.2:	Common activation functions	4
Figure 2.3:	Wasserstein training procedure	7
Figure 2.4:	Introspective Generative Modeling Process	9
Figure 2.5:	Introspective Generation Algorithm	10
Figure 3.1:	Conditional Generative Adversarial Network	14
Figure 3.2:	Text-Image Synthesis Network	15
Figure 3.3:	Embedding Network	17
Figure 3.4:	Similarity Network	18
Figure 4.1:	Discriminator training vs Generator training trainable parameters	20
Figure 4.2:	Affinity Network	22
Figure 5.1:	Text-Image Synthesis Network generates <i>a man</i>	24
Figure 5.2:	Two Branch Network generates CIFAR10	25
Figure 5.3:	Introspective 300 epochs	25
Figure 5.4:	Introspective 500 epochs	26
Figure 5.5:	Introspective 900 epochs	26
Figure 5.6:	Wasserstein Introspective 500 epochs	27
Figure 5.7:	Wasserstein Introspective 900 epochs	27
Figure 5.8:	Affinity Network generates <i>a dog</i>	28
Figure 5.9:	Affinity Network generates <i>a man</i>	28
Figure 5.10:	Conditional Generative Adversarial Network generates <i>a man</i>	28

LIST OF TABLES

Table 5.1: A list of inception scores for various models at 500 and 900 epochs.	27
---	----

ACKNOWLEDGEMENTS

Thanks to Professor Zhuowen Tu, whom, without his continued support, I would not have been able to work in this field. In addition, I would like to thank Justin Lazarow and Kwonjoon Lee, who gave me advice during my journey. This document would not have been possible without their vast expertise in Computer Vision and Deep Learning.

I would also like to thank Professor Hao Su and Professor Kamalika Chaudhuri for agreeing to be on the committee for this thesis.

ABSTRACT OF THE THESIS

Preserving Semantic Structural Constraints within Neural Networks

by

Hubert Hoai Le

Master of Science in Computer Science

University of California San Diego 2018

Professor Zhuowen Tu, Chair

Generative Adversarial Neural Networks are neural networks which participate in a zero-sum game, competing against each other to maximize an objective. One network, the generator, hopes to generate data that lies in a similar distribution to given data. The discriminator aims to separate data generated by the generator and ground truth data. This allows us to generate and replicate data given a dataset.

These networks have been increasingly popular in generating images from text. By leveraging the Introspective Learning framework, we are able to take image classification networks and synthesize images. We show that our results are competitive on many-to-many mappings against Conditional Generative Adversarial Neural Networks.

Chapter 1

Introduction

In the realm of machine learning, neural networks have become increasingly popular. With their extreme flexibility and large amount of tuneable parameters, they are adaptable to a wide variety of tasks.

Some common tasks include:

- Object Identification
- Re-identification
- Game AI
- Common Classification Tasks
- Image Segmentation
- Image Generation

These are only several common tasks. There are a great variety of uses not mentioned here. The task we address is image synthesis - given a data set of images, generate images from some sort of description. In order to do so, we utilize and compare several different approaches from several different neural network architectures. In particular, we discuss the importance

of our architecture and the relevance to the field of conditional generation, where an image is generated off of a condition, namely, text.

Ordinarily, images with similar image features are placed next to each other in the network's latent space. However, we would instead like to use a text input to place semantically similar images close to each other, i.e. images of a dog should be close to each other in the latent space, even if one may look visually similar to, say, a cat.

Chapter 2

Background

2.1 Neural Networks

Feed-forward neural networks are multiple-layer perceptrons that take some dimension of input and perform multiple linear transformations, followed by possibly nonlinear activation units, and outputs some points in dimension specified by the final layer. Figure 2.1 shows an example neural network.

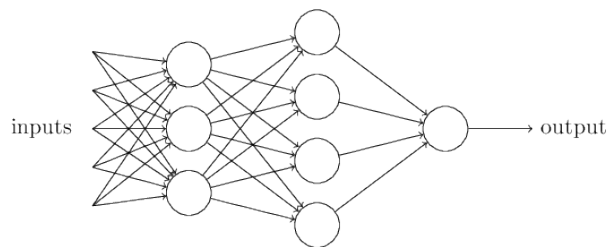


Figure 2.1: A typical feedforward neural network

For each of the perceptrons, they have an activation function of the form

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \quad (2.1)$$

2.1.1 Activation Functions

Generally, to force that the network is not simply a linear transformation of the input units, we utilize activation functions that will add nonlinearity to the network. This helps the network draw more complicated decision boundaries. Some of the more common activation functions are shown below.


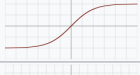



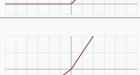
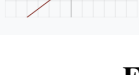
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ ^[1]	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Softsign ^{[7][8]}		$f(x) = \frac{x}{1 + x }$	$f'(x) = \frac{1}{(1 + x)^2}$
Inverse square root unit (ISRU) ^[9]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}} \right)^3$
Rectified linear unit (ReLU) ^[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU) ^[11]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Figure 2.2: Common activation functions

2.1.2 Optimization

Due to the sheer number of trainable parameters, neural networks more than several layers were deemed intractable in the early 2000's, until the arrival of the backpropagation [RHW86] algorithm. It allowed for gradients to be calculated quickly and easily, linear with respect to the number of the neurons in the network. The only requirements needed to train a network are then a loss function and an optimization method. Commonly used optimization techniques such as ADAM [KB14], stochastic gradient descent [RM51] as well as minibatch gradient descent are used to tune network parameters to maximize or minimize loss.

A clever choice of the loss function along with complex architectures allow for neural

networks to adapt to a variety of domains. While we will not go into the explanation here, layers such as the convolutional layer and deconvolutional layer have allowed us to process images more naturally to perform complex operations on them. We will see shortly that transformations in the image space are of great interest and can be actually enforced using semantic constraints.

2.1.3 Generative Adversarial Networks

Recent advances in neural networks have provided us with ways to frame neural networks as adversarial games. In particular, the Generative Adversarial Network [GPAM⁺14] is a type of system introduced in 2014 which is comprised of two neural networks participating in a zero-sum game. An analogy can be drawn between a counterfeiter and an art expert - the first network is a discriminator, which, in our analogy, is the art expert. It tries to tell real samples apart from fake samples generated by the generator, or counterfeiter. The generator learns the given input distribution and tries to generate images that the discriminator cannot differentiate between. As time goes on, the counterfeiter will have their samples marked as fake, and will learn to create better images to fool the art expert. Meanwhile, the art expert will have to learn to improve its ability to tell fake and real in order to keep up with the counterfeiter. The art that the counterfeiter will produce will become more like the input distribution, until at equilibrium, the art expert cannot differentiate real and counterfeit.

The loss function of the network is shown below:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(x)))] \quad (2.2)$$

Minimizing this loss is equivalent to the Jensen-Shannon distance between the generated and real distribution of inputs. However, the gradient of this loss function is often not smooth over most intervals and thus gradient descent has a difficult time.

2.1.4 Wasserstein Generative Adversarial Networks

In early 2017, the idea of using the Earth-Mover distance as the objective function [ACB17] was proposed, under the pretense that it provided smooth, nonzero gradients. The Wasserstein objective function is shown below:

$$W(P_r, P_\theta) = \inf_{\gamma \in \Pi} E_{x, y \sim \gamma} \|x - y\| \quad (2.3)$$

However, this distance is intractable in practice. Instead, we opt to use the Kantorovich-Rubinstein duality to solve an alternate equation, which was shown to be not only tractable, but provide better gradients and faster convergence.

$$W(P_r, P_\theta) = \sup_{\|f\|_{L \leq 1}} E_{x \sim P_r} f(x) - E_{x \sim P_\theta} f(x). \quad (2.4)$$

Due to the fact that the output of the network is now no longer a well-defined probability, but rather a distance, the training process is a bit different. Rather than have the generator fool a discriminator, there are instead critics. In the analogy, imagine that there existed critics that rated counterfeit and real works; rather than differentiating between real and fake, it allows us to tell if we're getting better and better. Taking the gradient of this function, we get a slightly modified training procedure. Figure 2.3 denotes this.

2.2 Learning Generative Models via Discriminative Approaches

Given a generative model, the discriminative approach to calculating $p(y = +1|x)$ is as follows:

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α , the learning rate. c , the clipping parameter. m , the batch size. n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while

```

Figure 2.3: Wasserstein training procedure

$$p(y = +1|x) = \frac{p(x|y = +1)p(y = +1)}{\sum_{y=-1,+1} p(x|y)p(y)} \quad (2.5)$$

Rearranging the terms, and assuming $p(y = +1) = p(y = -1)$ gives the following:

$$p(x|y = +1) = \frac{p(y = +1|x)}{p(y = -1|x)} p(x|y = -1) \quad (2.6)$$

Therefore, by learning a discriminative model, we can provide a generative approach to learning a given distribution [Tu07]. In order to learn $p(x|y = -1)$, denoted now as $p_t^-(x)$, we use the following learning process:

$$p_t^-(x) \rightarrow^{t=\infty} p(x|y = +1) \quad (2.7)$$

We are essentially sampling such that an $x \sim p_t^-$ becomes indistinguishable from our reference distribution. Denoted as pseudo-negatives, we continually sample from the distribution samples that our model believes are positive, then we label negative.

We have shown that discriminative approaches can be used to learn a generative model. Next, we show an application of this to deep learning.

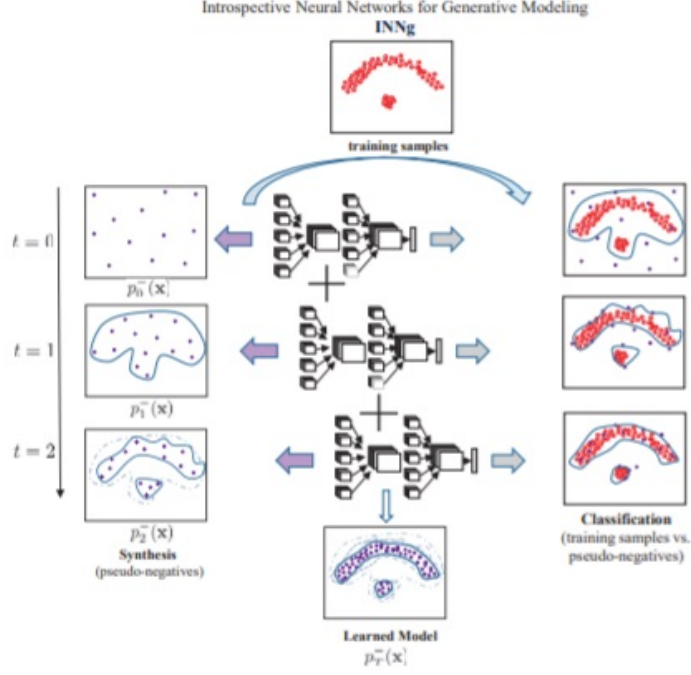
2.2.1 Introspective Learning

While Generative Adversarial Networks are able to generate images from a distribution, they do not have classification ability. Introspective Generative Modeling [LJT17] is a paradigm where a classifier learns to generate images based on a classifier network, then creates images via backpropagation instead of a forward pass. The general idea is that the input is an image, and the output is a scalar denoting real or not real. To generate images, the weights of the network are frozen, then backpropagation will push the gradients onto the original image itself. To generate an image, a trained classifier network simply has to input a noise image, then backpropagate on it until it is classified as real.

We draw a parallel in that generated images represent $p(x|y = +1)$, then relabeling them as negative to promote an adversarial-like task. In this case, however, the adversary is the classifier- it is simply learning what positive samples should look like and continually improving upon itself. This task is called Introspective Learning, and we will later utilize this framework in some of our approaches.

One might ask if this approach will converge. It can be shown that at each step n , the difference between the Kullback-Leibler divergence between our current estimate $D[p^+(x)||p_n^r(x)]$ and our next estimate $D[p^+(x)||p_{n+1}^r(x)]$ is positive, and thus, the distance decreases at each step. Z_n is simply a normalizing factor. We will also denote $q(x)$ to denote a learned model approximation.

$$D[p^+(x)||p_n^r(x)] - D[p^+(x)||p_{n+1}^r(x)]$$



$$\begin{aligned}
&= \int p^+(x) \log\left(\frac{1}{Z_n} \frac{q(y=+1|x)}{q(y=-1|x)} p_n^r(x)\right) dx - \int p^+(x) \log[p_n^r(x)] dx \\
&= \int p^+(x) \log\left(\frac{1}{Z_n}\right) dx + \int p^+(x) \log\left[\frac{q(y=+1|x)}{q(y=-1|x)}\right] dx \\
&= \log \frac{1}{Z_n} + \int p^+(x) \log\left(\frac{q(y=+1|x)}{q(y=-1|x)}\right) dx \geq 0
\end{aligned}$$

Since it must be the case that the divergence is decreasing at each step, we show convergence.

This method is extremely simple because it only requires a classifier to sample from. In essence, it enables us to turn any classification model into a generative model, which is extremely powerful. In addition, our introspective framework has the advantage of only requiring a single model instead of the two models required in the adversarial framework.

Our training algorithm is very simple, we just perform the exact calculations via sampling and update our model in a similar fashion as described.

Algorithm 1 Outline of the INNing algorithm.

Input: Given a set of training data $S_+ = \{(\mathbf{x}_i, y_i = +1), i = 1..n\}$ with $\mathbf{x} \in \mathfrak{R}^m$.

Initialization: obtain an initial distribution e.g. Gaussian for the pseudo-negative samples: $p_0^-(\mathbf{x}) = U(\mathbf{x})$. Create $S_-^0 = \{(\mathbf{x}_i, -1), i = 1, \dots, l\}$ with $\mathbf{x}_i \sim p_0^-(\mathbf{x})$

For $t=1..T$

1. Classification-step: Train CNN classifier C^t on $S_+ \cup S_-^{t-1}$, resulting in $q_t(y = +1|\mathbf{x})$.

2. Update the model: $p_t^-(\mathbf{x}) = \frac{1}{Z_t} \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x})$.

3. Synthesis-step: sample l pseudo-negative samples $\mathbf{x}_i \sim p_t^-(\mathbf{x}), i = 1, \dots, l$ from the current model $p_t^-(\mathbf{x})$ using a SGD-based sampling procedure (backpropagation on the input) to obtain $S_-^t = \{(\mathbf{x}_i, -1), i = 1, \dots, l\}$.

4. $t \leftarrow t + 1$ and go back to step 1 until convergence (e.g. indistinguishable to the given training samples).

End

Figure 2.5: Introspective Generation Algorithm

2.3 Word Embeddings

Unfortunately, neural networks don't have the ability to perform computations on text the same way that they do on numbers. We need to translate text into something that is machine intelligible. We opt to convert them to word vector form. There are several popular methods for this. We discuss them briefly.

2.3.1 Bag of Words

'*I watched a movie*'. Consider this body of text, which consists of 4 words. Often, we realize that there are words in the body of text that we can discard; namely, words such as *a* and *the* don't contribute much to the meaning of the sentence. We denote such words as stopwords, and remove them from the list. In addition to that, notice the meaning of the sentence is not warped much by removing the order of the words. In summary, what this means is that given a vocabulary, we can represent a sentence or corpus with just the relevant words in a set.

This representation is called 'bag of words'. The relative ordering of words is not used from the original corpus- instead, we opt to derive meaning of a sentence from the content of the sentence. In practice, this approach often works pretty well.

One way we vectorize this approach is to create an encoding of the given words in a corpus by a certain vocabulary. We can create a vector of the counts of the words of a given vocabulary occurring in a corpus to represent it. In this way, we retain only the word count and words present, which hopefully capture most of the meaning.

2.3.2 Term Frequency - Inverse Document Frequency

However, it may not be the case that the vocabulary words that appear often are the most indicative of the meaning of a sentence. For example, let's assume an article about a review for Star Wars. *Star Wars* itself may only appear once or twice in the entire article, and yet, is central to the meaning of the the article. In comparison, *space ship* might appear often, yet does not describe the article as well as *Star Wars*. Why is this the case?

Well, relative to other articles that we read, *Star Wars* appears more times in this article than in others. In other words, if we factor in that other articles don't contain these words, then the meaning of this article comes into light. Given a term and document, we can formulate this as such:

$$tfidf(t, d) = \frac{f(t, d)}{\sum_i f(t, d_i)} \log \frac{N}{n_t} \quad (2.8)$$

Where N represents the number of documents, n_t represents the number of documents which contain the term t . The first term is simply a term frequency count, multiplied by the relative importance of the word over all documents. Doing this calculation for every word and vectorizing it gives very good empirical results.

2.3.3 Word2Vec

There are drawbacks to using the beforementioned word vector methods. They lack the ability to do zero-shot learning. That is, if a word has not been seen before, then we have no information on it and thus, we cannot glean any information off of it. One of the most common solutions is to learn an extremely large vocabulary, and apply that to smaller use cases. In practice, if such a vocabulary is learned, even if we have not seen certain phrases before, we can use our previously learned knowledge to generalize.

We will not go into the approach of creating them in detail for now. Word2Vec [MCCD13] will define an n -dimensional vector for a phrase or sentence. For a given word2vec model, all generated vectors should be the same length. Similar words will appear in similar places in word2vec space, for example, calculating the norm of the distance between *man* and *woman* may be very small, in comparison to something like *man* and *car*. This approach provides us with ways to compare text using common loss functions such as L^2 loss.

Chapter 3

Related Work

Below, we discuss some popular approaches to combining text and image into the same space. They will serve as several different approaches to our current task.

3.1 Conditional Generative Adversarial Networks

Conditional Generative Adversarial Networks [MO14] are very similar to their predecessor, the Generative Adversarial Network, except that input is a concatenated reference distribution sample, along with a conditional y . This conditional acts as a signal - it provides information to the network on what to generate by using it as a prior. This conditional is often a class label or background information used to provide a more concise distribution for the output.

The formulation of these types of networks is very simple- only a slight change is necessary to the original loss function.

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x|y)] + E_{z \sim p_z(z)} [\log(1 - D(G(x|y)))] \quad (3.1)$$

Conditioned on some variable y , we can learn to associate certain parts of a reference

distribution with certain input variables.

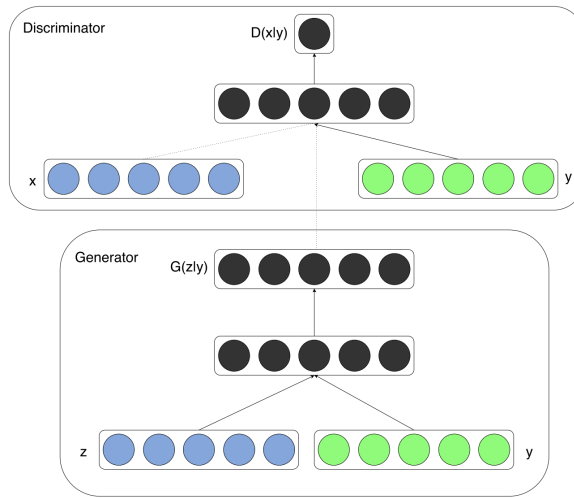


Figure 3.1: Conditional Generative Adversarial Network

This technique has achieved impressive empirical results. Due to the great flexibility of what the conditional is, there have been many applications regarding creative uses of the conditional to apply this model to various domains. In particular, the text-image domain has found great success in these approaches. We will discuss some of them here, as they are considered the state of the art models with regards to image synthesis in the joint image-text domain.

3.1.1 Text to Image Synthesis

In 2016, Reed created one of the most popular models today for synthesizing images from the text domain. The approach is a conventional discriminator generator architecture, but with a text conditional in the discriminator network [RAY⁺16]. The generator network takes in a text $\phi(t)$ and noise sampled from a gaussian distribution z to encourage variety in images generated, and outputs a generated image \hat{x} .

The discriminator network takes an image input, concatenated with text, and outputs a probability value representing the probability the image matches with the text. Training the

network is similar to training a conditional image network, but with modifications to ensure robustness.

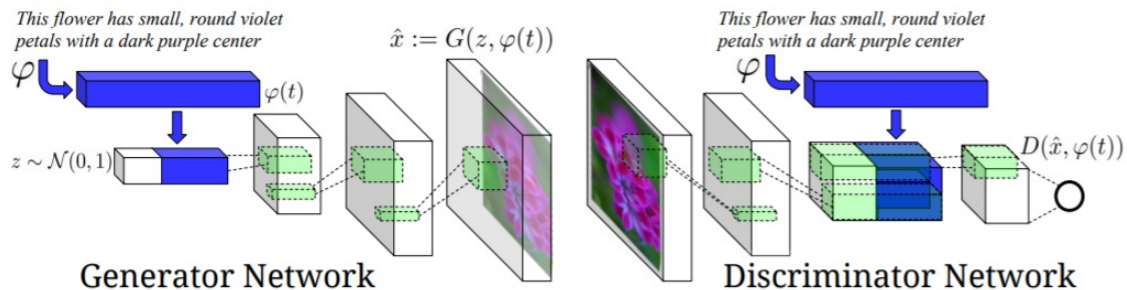


Figure 3.2: Text-Image Synthesis Network

The loss consists of three parts - s_r , which is a (real image, right text) pair, s_w , which is a (real image, wrong text) pair, and s_f , which is a (fake image, right text) pair. It is necessary to include all three parts, as not only must the network learn what a real image looks like, it must also learn what text to associate with that image.

While this network does extremely well on phrases with 'complete' information, it cannot generalize very well when the text is lacking. Consider *A blue bird sits on a branch*. The network has no issues generating an image such as this, because it has information on what color the bird should be, and the posture of the bird. However, when presented with text that does not have such descriptions, such as *A bird*, this network has a much harder time generating a realistic looking image if the variance of the input dataset is high, such as the case with the Microsoft Common Objects in Context [LMB⁺14] dataset, or if there exists a many to many mapping between phrases and images, as with the Flickr 30k entities dataset [PWC⁺15]. More on this will be discussed in the experiments section.

3.2 Two Branch Networks

In 2015, Wang [WLL17] created Two-Branch Neural networks for image-text preservation. The premise is simple- join a given image x and a text y , $f(x)$ and $f(y)$ can represent

transformations into a joined latent space. This embedding space is where both the text and image are pushed into similar dimensions and operations can be performed.

The notion of a class label has been relatively predominant with conditional generative adversarial networks so far, but text is a tricky domain because it isn't necessarily class based. With more complicated text phrases, the notion of a class breaks down. Therefore, we have to rely on the inherent semantic relationships between different phrases.

There are two networks of interest to us utilizing this two branch structure, and we will discuss them in further detail.

3.2.1 Embedding Network

Suppose we have an image patch x and a matching text y , along with a non-matching text y' . We want to enforce some sort of semantic constraint on how these two will map in the latent space- that is, the cosine similarity of these two should be high if they match.

In other words, the distance between the vector representation of x, y denoted by $d(x, y)$ should be smaller than the distance between $d(x, y')$. In this way, if we enforce that matching image-text pairs are closer to each other in the latent space, we are able to map a relationship between image and text.

Since this latent space is explicitly defined as the final layers of the two branches, imposing constraints works very well on the vector outputs of the network. There are three constraints that cater towards shaping the latent space to match commonsense expectations. Firstly, the distance between two similar images should be smaller than two nonmatching images; that is, $d(x_i, x_j) + m < d(x_i, x_k)$, where x_i, x_j match and x_i, x_k do not match. m represents a margin or neighborhood for which the matching pairs must be within. The other two constraints are also common sense: $d(y_i, y_j) + m < d(y_i, y_k)$, the text equivalent of the previous constraint, and $d(x_i, y_i) + m < d(x_i, y_k)$, the constraint that matching image-text pairs should be closer than nonmatching ones.

Embedding Network

$$d(\text{img}_1, \text{"a fire pit"}) + m < d(\text{img}_2, \text{"a fire pit"})$$
$$d(\text{img}_1, \text{"a fire pit"}) + m < d(\text{img}_3, \text{"campers"})$$

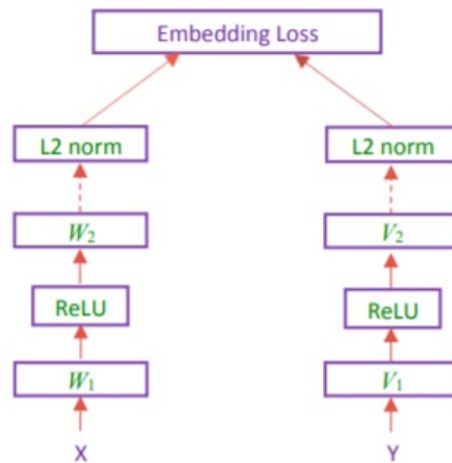


Figure 3.3: Embedding Network

3.2.2 Similarity Network

A much simpler task that does not involve us enforcing any sort of constraints is the similarity network. We merge the vectors into an element-wise product, and simply treat this as a classification task. We output a sigmoid probability that tends towards 1 if the image and text match. As it turns out, this simple approach, which is noticeably less complicated, achieves similar results in phrase localization tasks as the embedding network.

However, it is not guaranteed that semantic constraints are held in the latent space anymore. That is, we are not guaranteed the same neighboring constraints as in the embedding network. We will show later that this is the reason for which certain tasks don't perform as well on the similarity network.

Similarity Network

 , "a fire pit": +1

 , "a fire pit": -1

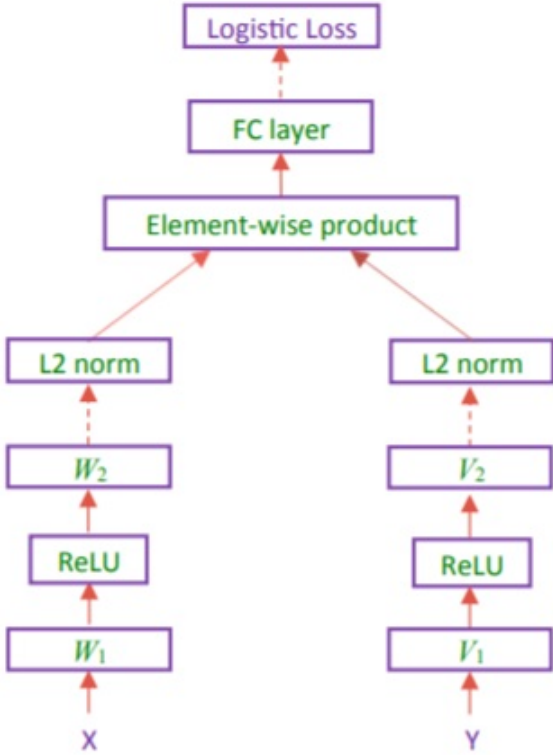


Figure 3.4: Similarity Network

Chapter 4

Methodology

Most of the previously shown approaches use various methods to combine text and image into a representation useable with classification tasks. For example, the two-branch networks allow us to reframe the text-image space into several different classification problems.

We also saw, through the introspective learning framework, that we can frame generative approaches through learning a classification task. Therefore, given the previously discussed classification approaches, by using a learned prior, we can reframe these classification tasks as generative problems.

In this way, we can transform each of the networks described to perform image synthesis. That is, $P(x|y = +1)$ can be approximated with $P(y = +1|x)$ via the equations previously specified.

4.1 Approaches

4.1.1 Introspective Image Synthesis

Using the two-branch network architectures described before, we reframe the classification task- given a classification network which outputs the similarity metric between a given phrase

and image, generate a given input distribution.

However, this is not so simple, as the triplet sampling embedding network approach is difficult to model. Given we want an image that is some fixed distance m from a given text, how would that image look? It is not quite well-defined what m we should pick, nor is it easy to estimate this via empirical tests. This distance is more or less arbitrary, as the space that we create can have different margin widths.

With the similarity network, we know that we would want a image which has a similarity of 1 to a given text. In other words, it is clear that $P(y = +1|x)$ must be 1. By setting the posterior probability to 1, we can generate an approximation for $P(x|y = +1)$, and refine it through the introspective learning framework.

Application is simple- we just freeze the weights during generator training in the network, and make the input image a trainable parameter. In that way, we have set the posterior to be 1, and backpropagation will change the input to match the supplied posterior with an estimate based on our current model.

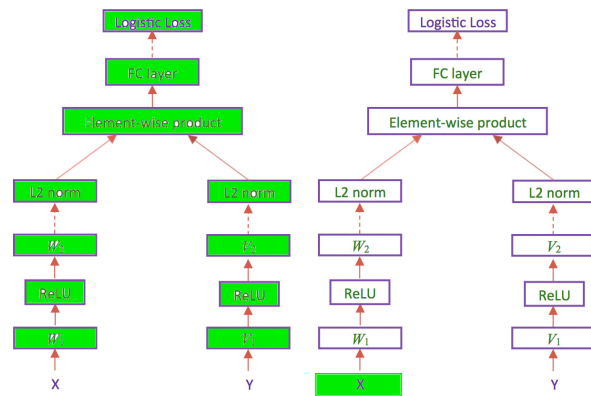


Figure 4.1: Discriminator vs Generator trainable parameters

To encourage variety in images, the Conditional Generative Adversarial Network framework concatenates noise with text to generate images- in our case, since we start our images from sampled gaussian noise and generate with backpropagation, this is not necessary.

The formulation for the loss of the network is a logistic regression loss.

$$L(x, y, z) = \sum_{i,j} \log(1 + \exp(-z_{ij}f(x_{ij}, y_{ij}))) \quad (4.1)$$

Where z_{ij} is 1 if image x_{ij} and text y_{ij} match, and -1 otherwise.

4.1.2 Wasserstein Introspective Image Synthesis

We found that, in the case of these networks, that leveraging the Wasserstein metric was possible in our introspective learning framework. Let us denote x_i, x_j to be real and fake images. Then, we change the interpretation of the output of the network- instead of a scalar that represents the probability that a given text matches an image, it becomes a distance scalar. We wish to maximize instead $d(x_i, y) - d(x_j, y)$, aiming to learn, instead of matching and non-matching images, the real image distribution and fake image distribution given a specific label.

The formulation for the loss function is the same as the previously shown Wasserstein Network loss. We are interested in separating the distribution of real images given a conditional text, but no other components of the loss need be changed.

In both this and the previously mentioned approach, we use the similarity network architecture. However, as mentioned previously, there are no explicit semantic constraints in the joined latent space- therefore, there are no guarantees that in our network’s latent space that images which are semantically similar are placed next to each other in this space. Rather, it is more likely that visually similar images will be placed close to each other.

4.1.3 Affinity Network

The previous approaches use the two-branch network architectures to create the latent space. However, there were no structural constraints on the latent space in order to group similar images and similar texts together. The embedding network requires that in order to use our

introspective framework, the backpropagation technique need be used with a specific margin and vector distance in mind, but this is difficult to obtain, because we do not know the true distance between a given matching image and text in the latent space.

Rather than doing that, we opt to instead frame the synthesis problem as a regression task: given an image, perform regression to obtain the text description for that image. In order to generate an image with this network using our introspective framework, we simply set the text embedding to a target embedding that we want, and perform backpropagation, making the input image the trainable parameter.

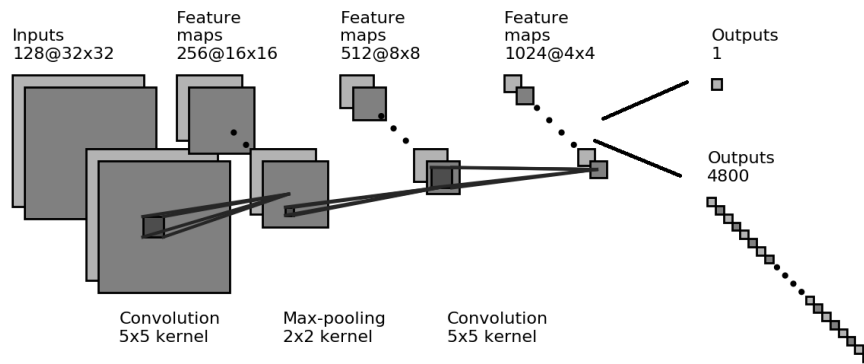


Figure 4.2: Affinity Network

In this way, we are able to embed semantic constraints in the latent space simply by training the network- the final layer's image features should mirror the word2vec space. To ensure

that the image that is produced is not only faithful to the semantic constraints, but also realistic looking, we reshape the image features to a scalar and represent the scalar as the Wasserstein distance. Now, the network must not only satisfy the semantic constraints, but also separate the real image distribution from the fake image distribution.

Chapter 5

Experiments

We first show image synthesis results on the Flickr 30k Entities dataset. This dataset has the benefit of being continuous, free-form phrases. While other text-image synthesis approaches have used datasets where the sentence structure is similar and very specific. As mentioned before, these networks have a difficult time generalizing when certain image aspects are not described, as the network learns to generate images given text. Images will be shown from the experiments first, and Inception scores will be given later. The images are shown in batches with randomized text used as descriptions. We use minibatch gradient descent with a batch size of 100 to speed up the training process.

This, in essence, means that zero-shot learning doesn't work very well for unique sentence structures. On the text-image synthesis network, which works very well for the caltech birds and flowers dataset with context-rich phrases, we attempt to generate images using simpler phrases. After training on the Flickr 30k Entities dataset, we generate images using *a man* as the conditional, but the images don't look very good.



Figure 5.1: Affinity Network

We add text labels to cifar10 and attempt to do text-image synthesis with our two-branch

similarity network. The results look very decent, considering the amount of variance intraclass in cifar.



Figure 5.2: Two Branch Network generates CIFAR10

The text does not play a large role in this generation, since there are no phrases nor complicated grammar. We have shown this network generalizes to the class label case, but we run experiments to see if it also generalizes to more complicated phrases such as *a man running on the Flickr 30k Entities*. The following images are after 300, 500, and 900 epochs.

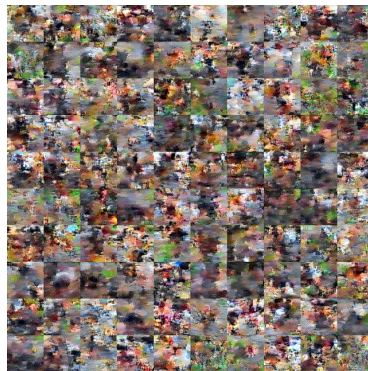


Figure 5.3: Introspective 300 epochs

While the network seems to be learning to generate smoother images, it cannot quite create realistic looking images. Due to the many modes within the many text classes, we assumed mode collapse might have been the problem. The solution, as suggested, was to try Wasserstein distance instead of a logistic probability as the final layer. This gave us better empirical results, and we show the 500 and 900 epoch results below.



Figure 5.4: Introspective 500 epochs



Figure 5.5: Introspective 900 epochs

While the batches' images look a bit more well-defined, it is difficult to discern what type of object the batch was trying to generate. In these experiments, we narrow down the objects to men and dogs, to reduce the number of modes. However, it's not clear that the network has learned what a man and a dog look like.

Using the affinity network, we attempt to generate a batch consisting of *a man* and a batch consisting of *a dog*.

With dogs, the network learns to associate green backgrounds with *dog*. However, the large amount of variance in backgrounds and shapes in *man* makes it quite difficult for the network to make a realistic image of *a man*.

We show some Inception scores on the original cropped Flickr 30K entities dataset, as well as the generated images below. While word2vec embeddings were used for these experiments,

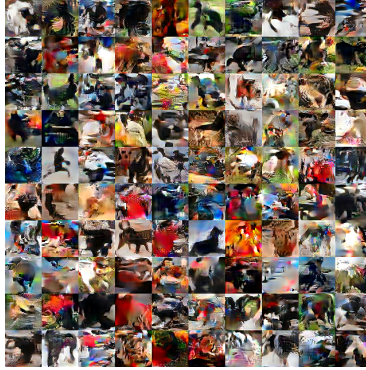


Figure 5.6: Wasserstein 500 epochs

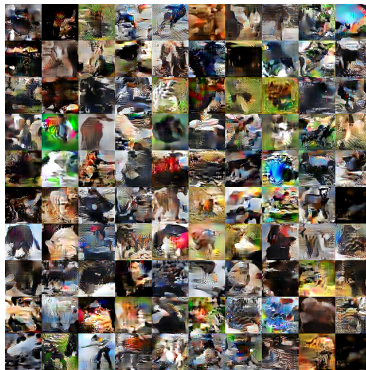


Figure 5.7: Wasserstein 900 epochs

we also experimented with term frequency-inverse document frequency embeddings, but found that their performance was roughly the same as word2vec, so those results were omitted. They are also a less interesting case due to the lack of zero-shot learning capability.

For the Wasserstein networks, we found that 10 critics worked the best. For the affinity network, a penalty of 0.5 was applied to the word2vec loss.

Table 5.1: A list of inception scores for various models at 500 and 900 epochs.

Models	500 epochs	900 epochs
Dataset	10.041	10.041
Introspective	2.099	2.202
Wasserstein Introspective	3.830	4.021
Affinity	4.060	4.050



Figure 5.8: Affinity Network generates *a dog*



Figure 5.9: Affinity Network generates *a man*

One thing to notice is that the affinity network doesn't see improvement after 500 epochs- this indicates that the full effect of the word2vec component of the loss has already taken effect.

We also ran the conditional generative adversarial network approach on the flickr30k entities dataset. Keeping in mind that an image can map to multiple phrases, and an phrase maps many images, this many to many mapping, along with the limited context in the text, makes it very difficult for the conditional network to generate images.



Figure 5.10: Conditional Generative Adversarial Network generates *a man*

The network does not have enough context to generate a man. This is to be expected, as this network is not meant to deal with our task, where a network must learn 'concepts' rather than

learn a small correspondence with each phrase.

Chapter 6

Conclusion

We have shown that the introspective modeling approach can reframe classification problems as generative problems, and that we can improve conditional image generation results by placing semantically similar images close to each other in the latent space. We encourage others to try applying our introspective approach to more complicated tasks and classification models.

Bibliography

- [ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.
- [GPAM⁺14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.
- [KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [LJT17] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [LMB⁺14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
- [MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.
- [PWC⁺15] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *CoRR*, abs/1505.04870, 2015.

- [RAY⁺16] Scott E. Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.
- [RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.
- [Tu07] Zhuowen Tu. Learning generative models via discriminative approaches. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [WLL17] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks. *CoRR*, abs/1704.03470, 2017.