공학석사 학위논문

Understanding Visualization Idioms Through Deep Visualization

딥 비주얼라이제이션을 통한 시각화 이디엄의 이해

2018 년 2 월

서울대학교 대학원

컴퓨터공학부

김 원 재

Abstract

Understanding Visualization Idioms Through Deep Visualization

Wonjae Kim
Department of Computer Science and Engineering
The Graduate School
Seoul National University

Visualization (vis) idioms define the way visual representations are created and manipulated. Well-known vis idioms include familiar charts such as bar charts and pie charts. Vis research and applications require thorough understanding of vis idioms. As a medium to understand vis idioms, I suggest a novel approach that employs deep visualization. Deep visualization is a collective name of methods for visualizing the characteristics of neurons in deep neural networks by generating their preferred stimuli (images). In this paper, I present two neural networks, one classifying the type of vis idioms, and one generating images of given idiom type. By applying deep visualization on the neural classifier through the neural generator, I examine how deep visualization can help vis researchers understand diverse aspects of vis idioms in novel ways and potentially derive unexplored idioms.

Keywords: Information Visaulization, Visualization Idiom, Deep Learning, Deep Visualization, Generative Model
Student Number: 2016-21192

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Frequently used charts and diagrams, such as bar charts and pie charts, are called *Visualization Idioms* (vis idioms) [10, 27]. Understanding vis idioms are one of the core research topics in the area of information visualization. For decades, researchers have tried to establish what visual encodings should be used to make a visualization more effective (with lower cognition burden [13]). Through many empirical studies [24, 5, 12], visualization researcher and practitioners are given with several visualization design guidelines such as *"Position encoding is the most accurate in supporting quantitative data"* or *"Use hue to encode categorical/nominal data"*.

Vis idioms comprise of various visual encodings which encode data into graphical marks and their channels. Simple idioms use only a few visual encodings compared to complex idioms which require a lot. A thorough understanding of these encodings is necessary in order to fully understand the vis idioms and make an improvement. However, in complex vis idioms, understanding the encodings one by one is hard in practice. This is because the boundaries between the idioms are very unclear. Many people retain their own concepts of vis idioms: one may regard a figure of points of various sizes as a variation of scatterplot while another may regard it as a bubble chart. Such vagueness of definitions confuses vis researchers by blurring the distinction between the

explored and unexplored and causing controversy regarding in what dimension should a new vis idiom be defined. This hinders the development of a novel vis idiom. To deal with this problem, I employ a data-driven approach to draw a statistical border between different vis idioms, by letting the machine to make the boundaries clear. By training a neural classifier to classify the types of given images of vis idioms, the model learns clear probabilistic boundaries between the idioms.

Meanwhile, the neural classifier itself is merely a complex set of matrices. Therefore humans need other visualizations to understand why the model concludes an images as a bar chart or pie chart. Computer vision researchers have taken two major approaches: point-based visualizations and network-based visualizations [22]. Point-based visualizations try to understand the behavior of a neural network through data points, which is to search or generate the data points that maximally activate the target neuron or the target layer [21, 7, 44, 43, 25, 31, 29, 26, 28]. On the other hand, network-based visualizations focus on the connections between neurons (in most cases, the weights) and visualize the distribution of those values. Although these two approaches could be used simultaneously [21, 34], network-based visualizations often work only with specific types of networks, which makes them harder to be combined with other methods. Thus I focus on point-based visualizations to make the contributions more general.

To employ point-based visualizations, specifically the ones generating data points, we must decide which generator to use in generating the preferred data points (images). I chose AlexNet for my classifier and Plug and play generative networks (PPGNs) for my generator. Moreover, since the vis idiom corpus that I collected is about two hundred times smaller than ImageNet, which both the original AlexNet and PPGNs were trained upon, I replaced the ordinary ad-

versarial training of PPGNs with training of boundary-equilibrium generative adversarial networks (BEGAN) to achieve more stability in training. Thus, this paper has the following contributions: (1) A novel analysis of vis idioms using the state-of-the-art deep generative model, (2) An automatic exploration of innovative vis idioms using visual analogies in the latent space, (3) Improvements made upon plug and play generative networks using boundary-equilibrium generative adversarial networks, and (4) A stabilized sampling method based on mini-batching and optimization techniques.

Chapter 2

Related Work

In this chapter, I explain about the two pillars of this work: point-based visualizations and vis idioms. The first section will show how activation maximization and its sampling method evolved for the past few years. And the second section will list some studies about the components and the human perception of vis idioms.

2.1 Point-based Visualization

Attempts to understand a neural network using the preferred stimuli have been made since neural networks were proved useful [7]. The mainstream of point-based visualizations is often called *Activation maximization*, namely a set of methods trying to find the stimuli that maximize the activation value of the target neuron.

The activation maximization methods used in the early researches were simple enough to understand, which are simply taking ascending gradient steps on input stimuli with the target neuron's gradient of the activation value with respect to the input stimuli [Algorithm 1]. This sampling (hereafter, naive sampling) can yield a very high activation in only a few steps. However, considering the characteristics of CNNs such as shift invariance [39], numerous stimuli would result in very high activation values. Indeed, images sampled by naive sampling

Algorithm 1 `NaiveSampling`

Input : a pre-trained classifier $\mathcal{C}$

Input : a target neuron t in the $\mathcal{C}$

Output : an image X that is preferred by t

1: Sample $X \in \mathbb{R}^{c \times h \times w} \sim U(-1, 1)$

2: **while** the activation value $t(X)$ is under a certain value **do**

3: $\quad L_{activation} \leftarrow t(X)$

4: $\quad X \leftarrow X + \epsilon * \nabla_X L_{activation}$

5: **end while**

6: **return** X

are very hard to understand or recognize for humans [Figure 4.1].

In recent studies, researchers have tried to harness the gradient steps of naive sampling with natural image priors. At first times, the priors were simple heuristics of natural images, such as gaussian blur [Equation 2.1] [43] and total variation [Equation 2.2] [8, 26, 31]. By adding regularization steps in the sampling procedure [Algorithm 2], the sampler can yield data points that are smoother and more recognizable than those from a naive sampler [Figure 4.2].

With the growth of generative models, Nguyen et al. suggested DGN-AM [29]. DGN-AM employs a deep generator network as a natural image prior. This naive sampling procedure [Algorithm 3] with learned prior takes an ascending gradient step on the hidden vector rather than the input image. Hence, the only possible input images to the classifier would be the ones made by the generator. As the sampling space is constrained by a much articulated regularizer, DGN-AM was able to produce preferred stimuli of the best quality for the natural image classifier.

Lately, Nguyen et al. [28] improved the sampling procedure of DGN-AM

Algorithm 2 `NaiveSamplingWithPrior`

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

$$TV(X) = \sum_{x,y} \sqrt{\left(\frac{X_{x+1,y} - X_{x-1,y}}{2}\right)^2 + \left(\frac{X_{x,y+1} - X_{x,y-1}}{2}\right)^2} \tag{2.2}$$

Input : a pre-trained classifier $\mathcal{C}$

Input : a target neuron t in the $\mathcal{C}$

Input : a regularizer r_θ simulates natural image prior. r_θ could be either L2 decay ($r_\theta(x) = (1 - \theta_{decay}) \cdot x$), gaussian blur ($r_\theta(x) = GaussianBlur(x,\ \theta_{b\_width})$, where θ_{b_width} is width of gaussian blur kernel [equation 2.1]), or total variation ($r_\theta(x) = (1 - \lambda_{TV} \cdot TV(x)) \cdot x$, where $TV(x)$ is total variation of x [equation 2.2]).

Output : an image X that is preferred by t

1: Sample $X \in \mathbb{R}^{c \times h \times w} \sim U(-1, 1)$
2: **while** the activation value $t(X)$ is under a certain value **do**
3: $\quad L_{activation} \leftarrow t(X)$
4: $\quad X \leftarrow r_\theta(X + \epsilon \cdot \nabla_X L_{activation})$
5: **end while**
6: **return** X

with Metropolis-adjusted Langevin algorithm (MALA) [35, 36] [Algorithm 4]. Additionally, they showed that the generator is reusable as the classifier network can be replaced with another network, say, image captioning network. This family of methods are called plug and play generative networks (PPGNs). However, training a deep generator network that PPGN uses is slow and unstable due to GAN loss [9]. I observed that GAN training was more unstable when approximating the distribution of images of very low total variation such

Algorithm 3 `NaiveSamplingWithLearnedPrior`

Input : a pre-trained classifier $\mathcal{C}$

Input : a pre-trained generator $\mathcal{G}$

Input : a target neuron t in the $\mathcal{C}$

Output : a latent vector h that $\mathcal{G}(h)$ is preferred by t

Sample $h \in \mathbb{R}^h \sim U(-1, 1)$
while the activation value $t(\mathcal{G}(h))$ is under a certain value **do**
    $L_{activation} \leftarrow t(\mathcal{G}(h))$
    $h \leftarrow h + \epsilon \cdot \nabla_h L_{activation}$
end while
return h

Algorithm 4 `MetropolisAdjustedLangevinAlgorithm`

Input : a pre-trained classifier $\mathcal{C}$

Input : a pre-trained generator $\mathcal{G}$

Input : a pre-trained half-encoder $\mathcal{E}_1$

Input : a target neuron t in the $\mathcal{C}$

Output : a latent vector h that $\mathcal{G}(h)$ is preferred by t

Sample $h \in \mathbb{R}^h \sim U(-1, 1)$
while the activation value $t(\mathcal{G}(h))$ is under a certain value **do**
    $L_{prior} \leftarrow \mathcal{E}_1(\mathcal{G}(h)) - h$
    $L_{activation} \leftarrow t(\mathcal{G}(h))$
    $h \leftarrow h + \epsilon_1 \cdot \nabla_h L_{prior} + \epsilon_2 \cdot \nabla_h L_{activation} + N(0, \epsilon_3^2)$
end while
return h

as charts and diagrams. I will discuss this training issue further and suggest a modification to stabilize training in section 3.4.

2.2 Visualization Idioms

$$\begin{aligned}\text{Data-ink ratio} &= \frac{\text{Data-ink}}{\text{Total ink used to print the graphic}} \\ &= \text{propotion of a graphic's ink devoted to a} \\ &\quad \text{non-redundant display of data-information} \\ &= 1.0 - \text{proportion of a graphic that can be erased} \end{aligned} \tag{2.3}$$

Information visualization is the study of visual representations of abstract data that lowers the cognition burden of humans while exploring and analyzing the data. The performance of visualization idioms (or vis idioms) is traditionally evaluated by their data-ink ratio [42, 14, 16]. Data-ink ratio is calculated by dividing the amount of ink (in modern computer-aided visualization, the amount of rendered pixels) used in visual encodings by the amount of total ink [Equation 2.3]. Visual encodings consist of visual marks and channels: marks are geometric primitive objects such as points (0D), lines (1D), areas (2D), volumes (3D) that represent a data point, and channels are the properties of the marks such as hue, saturation, texture, etc.

The rank of expressiveness (meaning how well the marks and channels express the value of the data) and the rank of effectiveness (meaning how well they match the visual saliency to the importance of the data) are well studied empirically [27, 24, 5, 12]. Studies on the expressiveness/effectiveness ranks try to find out the most prominent visual encoding that produces maximal perceptual variation. Think of thousands of bar charts as an example. The difference between the charts can not be the bar (a rectangular line mark) itself; instead, the color, position, width, height and the numbers of the bars are what distinguish one chart from another. If the height was the most diverging channel in the perception of subjects, than the height of the bar is the most prominent

visual encoding for a bar chart. These ranks tend to be applied exclusively to simple vis idioms like bar charts and line charts, the ones that use one or only few visual encodings. More complex vis idioms such as sankey diagram use a larger number of visual encodings. In order to understand the data represented by complex vis idioms, the interactions between the encodings should be learned on top of the ranks.

This approach is clearly a posteriori, since the charts and diagrams (vis idioms) should be rendered in advance to be evaluated. However, these idioms are not designed upon the ranks of expressiveness and effectiveness, but were originally designed just based on human intuition. Thus these studies would be a kind of reverse engineering that find out which part of the human intuition led designers to come out with such idioms. In this paper, I capture the distribution of visual encodings that vis idioms have by employing a neural network not by human intelligence tasks (HITs) [12]. Note that excluding human subjects does not mean the approach is irrelevant in terms of human perception, since the image corpus (dataset of bitmap images of vis idioms) is still made upon human intuition. Plus, a mathematical approximation of the distribution of visual encodings can disentangle their interactions onto a latent space, thereby enabling automatic compositions of the encodings. Figure 4.8 shows an example of such composition; here, the addition of a treemap and a streamgraph results in a novel vis idiom, and the visual encodings included in this new idiom is the intersection of the ones in treemaps and streamgraphs. Humans are not capable of conducting such operations unless they fully understand the complex interactions of visual encodings in both visual idioms, which is in many cases highly unlikely at first sight.

Chapter 3

Architecture

3.1 Dataset

Visualization idiom	Search keyword	Before filtered	After filtered
Area Chart	area chart	300	251
Bar Chart	bar chart	300	268
Bubble Chart	bubble chart	300	267
Chord Diagram	chord diagram	300	95
Donut Chart	donut chart	300	249
Fan Chart	fan chart	300	172
Gantt Chart	gantt chart	300	219
Line Chart	line chart	300	202
Pie Chart	pie chart	300	292
Radar Chart	radar chart	300	253
Sankey Diagram	sankey diagram	300	258
Scatterplot	scatter plot	300	276
Streamgraph	streamgraph	300	300
Tree Diagram	tree diagram	300	230
Treemap	tree map chart	300	186
Venn Diagram	venn diagram	300	275

Windrose Chart	windrose chart	300	147
Total		5100	3940

Table 3.1: The number of collected and filtered bitmap chart images retrieved from Google Image Search. The filtering condition is same as [15]. Most of the chord diagrams are filtered out since the search results also returned images for guitar chord diagram (tabs).

I first listed 30+ vis idioms, each with its own name. Then I ran a Google Image Search with the names to gather bitmap images for those idioms. After excluding the idioms with less than 300 search results, 17 vis idioms were left. The images were filtered manually (the exact conditions are same as [15]). Total numbers of images for each chart type are shown in [table 3.1].

3.2 Classifier

Layer	Specification	
	Train	Test
Input Data	RGB images with various width and height	
Augmentation	resize and random crop to 224x224	resize to 256x256
	random horizontal flip with p=0.5	center crop to 224x224
Convolution (conv1)	kernel=11x11x3x64, stride=4, padding=2	
Batch Normalization	2D Batchnorm with channel size of 64	
Pooling (pool1)	2D Max-pool with kernel size of 3x3, stride=2	
Non-linearity	ReLU	

Convolution (conv2)	kernel=5x5x64x192, stride=1, padding=2	
Batch Normalization	2D Batchnorm with channel size of 192	
Pooling (pool2)	2D Max-pool with kernel size of 3x3, stride=2	
Non-linearity	ReLU	
Convolution (conv3)	kernel=3x3x192x384, stride=1, padding=1	
Batch Normalization	2D Batchnorm with channel size of 384	
Non-linearity	ReLU	
Convolution (conv4)	kernel=3x3x384x256, stride=1, padding=1	
Batch Normalization	2D Batchnorm with channel size of 256	
Non-linearity	ReLU	
Convolution (conv5)	kernel=3x3x256x256, stride=1, padding=1	
Batch Normalization	2D Batchnorm with channel size of 256	
Pooling (pool5)	2D Max-pool with kernel size of 3x3, stride=2	
Non-linearity	ReLU	
Flatten	flatten activations to size of 256x6x6	
Dropout	Dropout with p=0.5	No Dropout
Fully Connected (fc6)	(256x6x6) $\rightarrow$ (4096)	
Non-linearity	ReLU	
Dropout	Dropout with p=0.5	No Dropout
Fully Connected (fc7)	(4096) $\rightarrow$ (4096)	
Non-linearity	ReLU	
Fully Connected (fc8)	(4096) $\rightarrow$ (17)	

Table 3.2: Detailed description for modified AlexNet [18]

Figure 3.1: 100 sample test images randomly picked from chart image corpus. All images are resized to 256x256 and center cropped to 224x224 as depicted in the augmentation layer of Table 3.2

I chose AlexNet for my neural classifier and slightly modified it by adding a batch normalization in every convolutional layer [Table 3.2]. To train the classifier, I used the same data augmentation techniques used in the original paper [18]: scaled random crop to 224x224x3, random horizontal flip with probability

of 0.5, normalize pixel values into $(-1, 1)$ with RGB mean $[0.8252, 0.8261, 0.8108]$ and standard deviation $[0.2809, 0.2602, 0.2875]$. With five-fold cross validation, the classifier classifies 17 chart types with $0.853 \pm 0.007\%$ top-1 test accuracy, which is a fair amount compared to the reported 88.8% 10 class accuracy of AlexNet [15]. The training of the parameters were done by Adam optimizer [17] with the learning rate of 0.001. I dropped the learning rate by half every time when the validation accuracy showed no improvement for 20 epochs. A total number of epoch per CV split was 300, and the batch size was 32. The training of the classifier approximately took 2 hours using a single Nvidia Titan X.

3.3 Noiseless Joint PPGN-h

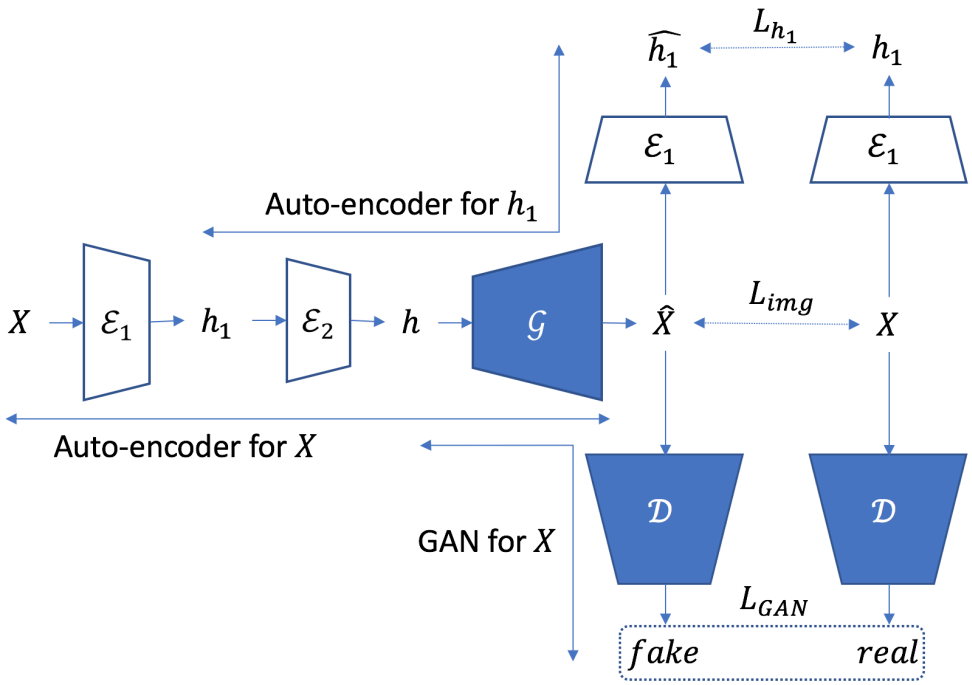

Figure 3.2: The original structure of noiseless joint PPGN-h from [28]

In this section, I will introduce the original architecture and the training procedure of the noiseless joint PPGN-h, the best model among the PPGNs reported in [28]. Since the architecture is quite complex, please refer the original paper [28] for full details. Noiseless joint PPGN-h consists of four neural networks: an encoder $\mathcal{E}$, a generator $\mathcal{G}$, a classifier $\mathcal{C}$, and a discriminator $\mathcal{D}$ [Figure 3.2]. The $\mathcal{E}$ and $\mathcal{C}$ are fixed networks that help training $\mathcal{G}$ and $\mathcal{D}$.

$$L_{DeepSIM} = \lambda_{h_1} L_{h_1} + \lambda_{img} L_{img} + \lambda_{GAN} L_{GAN} \tag{3.1}$$

$\mathcal{G}$ generates a fake image $\hat{X}$ with an input vector h, which is the activation of the neurons in the `fc6` layer (if $\mathcal{E}$ is AlexNet) of $\mathcal{E}$. $\mathcal{E}$ produces both h_1 (`conv5`) and h (`fc6`) with an input image X, which is batched from the dataset. To train $\mathcal{G}$, joint PPGN-h uses DeepSIM loss [6] [Equation 3.1], and a weighted sum of three losses: image reconstruction loss, perceptual loss, and GAN loss. To obtain the perceptual loss, $\mathcal{E}$ should produces $\hat{h_1}$ and $\hat{h}$ using $\hat{X}$. This round trip makes training of joint PPGN-h similar to an inverted autoencoder $(X \rightarrow)h \rightarrow \hat{X} \rightarrow \hat{h}$. The GAN loss follows Goodfellow's original paper [9]. $\mathcal{D}$ is trained along with $\mathcal{G}$ using the GAN loss.

3.4 PP-BEGAN

Layer	Specification
Input Data	latent vector with size of 4096
Fully Connected	(4096) $\rightarrow$ (4x4x128)
Upsample	2D upsample with scale factor of 2
Convolution (upconv1)	kernel=3x3x128x128, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU

Upsample	2D upsample with scale factor of 2
Convolution (upconv2)	kernel=3x3x128x128, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Upsample	2D upsample with scale factor of 2
Convolution (upconv3)	kernel=3x3x128x128, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Upsample	2D upsample with scale factor of 2
Convolution (upconv4)	kernel=3x3x128x128, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Upsample	2D upsample with scale factor of 2
Convolution (upconv5)	kernel=3x3x128x128, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Upsample	2D upsample with scale factor of 2
Convolution (upconv6)	kernel=3x3x128x3, stride=1, padding=1
Batch Normalization	2D Batchnorm with channel size of 3
Non-linearity	Tanh

Table 3.3: Detailed description for the generator and the decoder of the discriminator

Layer	Specification
Input Data	Image with size of 256x256

Convolution (conv1)	kernel=3x3x3x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Convolution (conv2)	kernel=3x3x128x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Convolution (conv3)	kernel=3x3x128x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Convolution (conv4)	kernel=3x3x128x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Convolution (conv5)	kernel=3x3x128x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Convolution (conv6)	kernel=3x3x128x128, stride=2, padding=1
Batch Normalization	2D Batchnorm with channel size of 128
Non-linearity	ELU
Fully Connected	(4x4x128) $\rightarrow$ (4096)

Table 3.4: Detailed description for the encoder of the discriminator

Most of the chart images have dense clusters of pixels of either very high or low values (255 or 0) in a given color channel [Figure 3.1]. In a traditional GAN training trying to directly match the data distribution itself, such property of chart images becomes an obstacle; when we train GANs with natural images

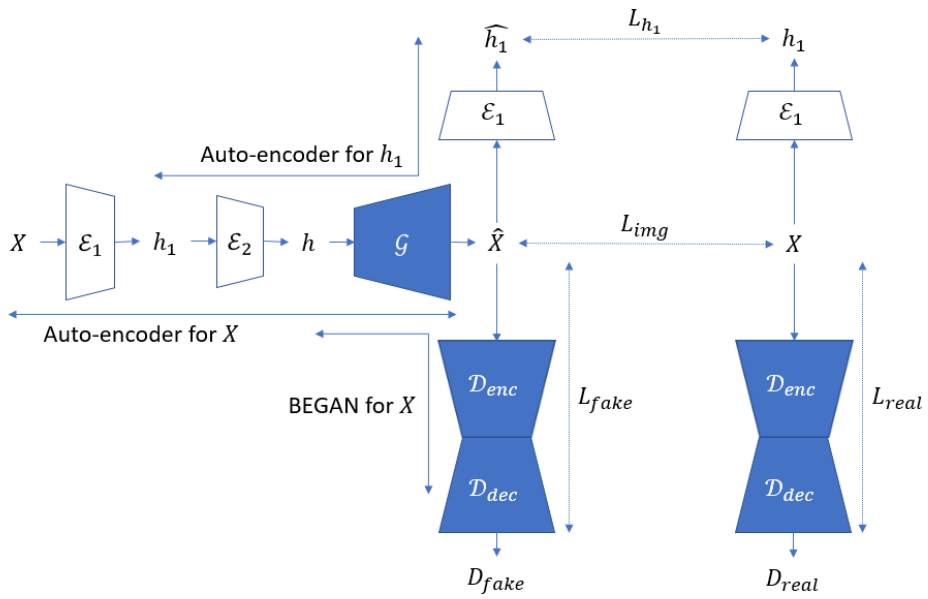

Figure 3.3: Structure of PP-BEGAN

(in the early stage of training) the generator tends to form noise images at first, then gradually improves using the gradients from the discriminator. This is possible because the noise made by the generator is not easily discernible to the discriminator when convolution layers are used [30]. However, when we use images with low total variation like chart images, the discriminator quickly learns the difference between fake and real images, and this ruins the GAN training. Moreover, the chart image corpus is very smaller than the ImageNet corpus (about a thousand times) [Table 3.1]. Noiseless joint learning of the noiseless joint PPGN-h by nature restricts the exploration of the latent space. The GAN training could easily fall into a mode collapse [9] if only a few thousand latent vectors are explored.

We need a much stable adversarial training to deal with this problem. As

a remedy, I replaced the ordinary GAN training of noiseless joint PPGN-h with a boundary-equilibrium GAN (BEGAN) [3]. The BEGAN uses an auto-encoder as its discriminator. By doing so, the architecture tries to match the auto-encoder loss between the real and fake images, not directly matching the data distribution itself. Unlike Jensen-Shannon divergence used in ordinary GAN training, Wasserstein distance [1] between auto-encoder losses can provide gradual gradients to the generator even with little overlap in the distributions of real images and fake ones. I named this modified network the plug and play boundary-equilibrium GAN (PP-BEGAN).

Figure 3.3 shows the overall structure of PP-BEGAN. The generator/decoder comprises one fully-connected layer followed by six up-convolutional layers, and the encoder comprises six convolutional layers followed by one fully-connected layer. Since the encoder/decoder used in $\mathcal{G}$ and $\mathcal{D}$ takes/generates an image of size 256x256x3 while $\mathcal{C}$ and $\mathcal{E}$ takes an image of size 224x224x3, all the images generated by $\mathcal{G}$ and $\mathcal{D}$ are center-cropped to 224x224x3 to calculate image reconstruction loss. Detailed descriptions for reproducing the layers are shown in [Table 3.3, 3.4].

$$\begin{cases} L_D = L_{real} - k_t \cdot L_{fake} \\ L_G = \lambda_{h_1} L_{h_1} + \lambda_{img} L_{img} + \lambda_{GAN} L_{fake} \\ k_{t+1} = k_t + \lambda_k (\gamma L_{real} - L_{fake}) \end{cases} \tag{3.2}$$

To fully exploit the learned classifier $\mathcal{C}$, objectives of the BEGAN are slightly modified. While two objectives remain the same as the original BEGAN, L_G employs L_{h_1} and L_{img} in order to employ DeepSIM loss [Equation 3.2].

When training with chart images, PP-BEGAN was significantly more stable than noiseless joint PPGN-h. Figure 3.4 shows direct comparison between the

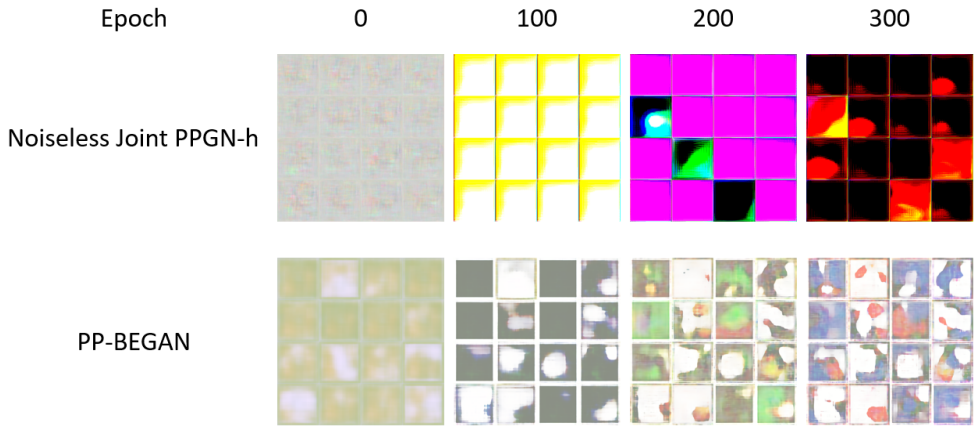

Figure 3.4: Samples generated by noiseless joint PPGN-h and PP-BEGAN. 16 fixed random latent vectors $h \in \mathbb{R}^{4096}$ were fed to each generator.

two generators at epoch $[0, 100, 200, 300]$. Sixteen fixed random latent vectors $h \in \mathbb{R}^{4096}$ were fed to each generator for fair comparison. The training of the generator from noiseless joint PPGN-h approximately took a single day for 300 epochs while the generator of PP-BEGAN took a half day more. Both training was done by a single Nvidia Titan X. Note that this result is of chart images; using natural images in training generators might lead to different results. I leave this to future work.

Chapter 4

Exploratory Study

In this chapter, I will explore the various facets of vis idioms preferred by the neural classifier [Table 3.2]. This section comprises of three parts. First, I exhibit the preferred stimuli those sampled with four different sampling method — naive sampling, naive sampling with total variation, MALA with noiseless joint PPGN-h, and MALA with PP-BEGAN — then discuss about the sampling methods. Second, with the samples obtained with the PP-BEGAN, I will report interesting findings of what the generator learned from the chart images. Third, I apply visual analogies and interpolations to the sampled hidden vectors (`fc6`) to verify that PP-BEGAN correctly captures the distribution.

4.1 Sampling Results

Figures 4.1, 4.2, 4.3, 4.4 show the 17 chart types sampled with the four different sampling methods. For naive sampling, I simply added the gradient of the target neuron's activation value to the input [Algorithm 1]. To add a total variation regularizer to naive sampling, I conducted a total variation regularization r_θ [Algorithm 2] with the total variation factor λ_{TV} of 0.5. The other two results are sampled with MALA [Algorithm 4], (ϵ_1, ϵ_2, ϵ_3) = (1e-5, 1, 1e-15). Note that the negative of the objectives ($L_{activation}$ and L_{prior}) are used as Adam tries to find the minimum at default. The samples from noiseless joint PPGN-h are

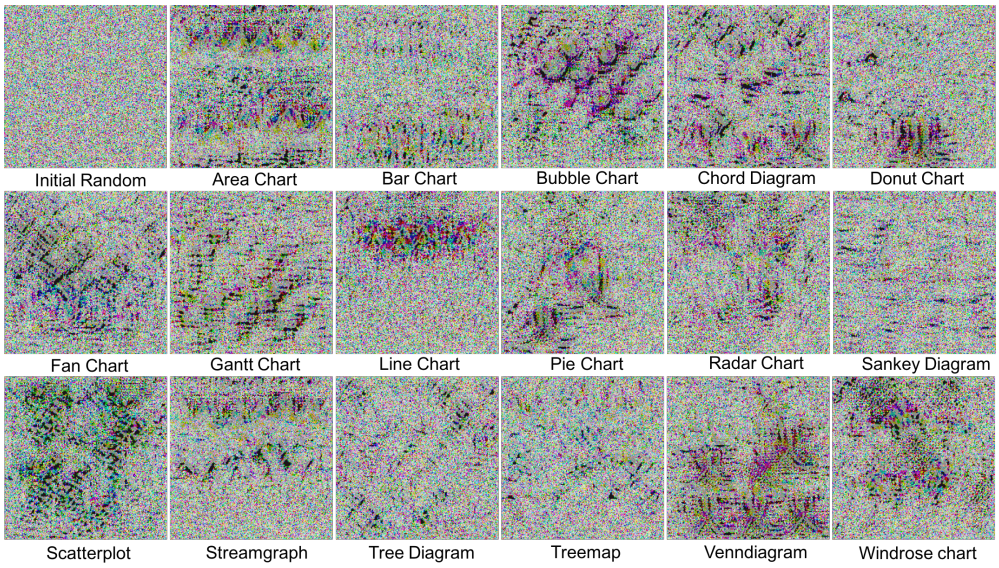

Figure 4.1: Preferred images of 17 chart types sampled with naive sampling with the given chart type classifier. Optimized via SGD with a learning rate of 1.

highly unstable and much harder to recognize than those from PP-BEGAN, as predicted in the previous section 3.4.

Various optimization methods exist for naive sampling and MALA. The original PPGNs paper [28] did a simple stochastic gradient descent (SGD) on the negative objectives with a learning rate of 1, without any momentum [38] or caching [41] to anneal the stimulus. In figures 4.1, 4.2, 4.3, 4.4, I used SGD with a learning rate of 1 for two the naive sampling methods and did Adam with learning rate of 0.01 for two MALA methods. To find out how the different optimization methods affect the sample results, I ran MALA on PP-BEGAN with Adam optimization at a learning rate of 1, 1e-1, 1e-2, and SGD at a learning rate of 1 [figure 4.5]. As a result, I found that Adam was significantly better than the SGD at increasing the activation value of the target neuron, and

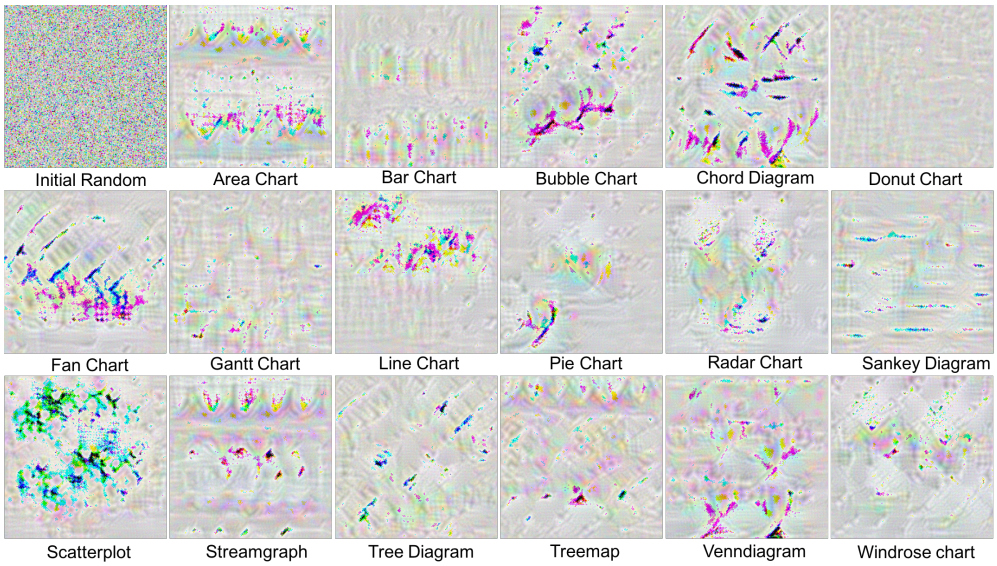

Figure 4.2: Preferred images of 17 chart types sampled with naive sampling with the given chart type classifier. Total variation regularizer was used (with λ_{TV} of 0.5). Optimized via SGD with a learning rate of 1.

also found that smaller learning rates work better. Besides optimization, as the networks of PP-BEGAN adopt batch normalization, I found that mini-batching the sampling process yields better samples.

4.2 Qualitative Analysis

Figure 4.6 shows 225 samples preferred by the neuron that classifies area chart. Note that PP-BEGAN was trained on approximately 160 images per chart. As mentioned in section 2.2, to grasp human intuition on these samples, I collected open feedbacks from 23 participants (1 female, age $= 24.3 \pm 3.22$, 7 had studied information visualization) by showing 1700 images (100 for each chart type) and asking *"What are the concepts shared by these images?"*.

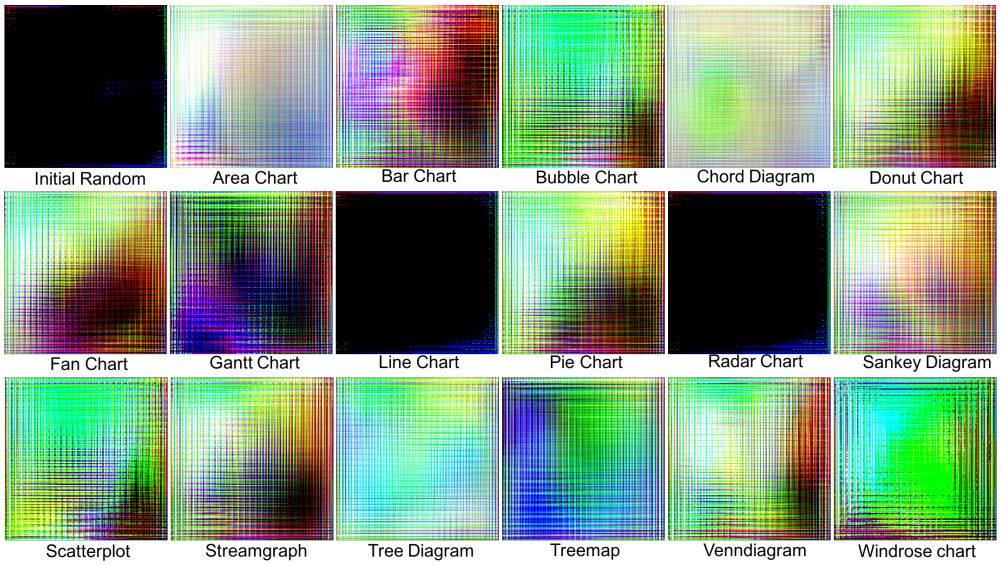

Figure 4.3: Preferred images of 17 chart types sampled with MALA with the given chart type classifier and the noiseless joint PPGN-h. (ϵ_1, ϵ_2, ϵ_3) = (1e-5, 1, 1e-15). Optimized via Adam with a learning rate of 0.01.

vis idiom	Responses (number)
Area Chart	Mountains, Curves
Bar Chart	Rectangles, Buildings
Bubble Chart	Circles, Paints
Chord Diagram	Caterpillar, Pepper
Donut Chart	Curvatures, Ring
Fan Chart	Unrecognizable
Gantt Chart	Stairway, Rectangles, Wafer
Line Chart	Line, Threads
Pie Chart	Circle, Clot
Radar Chart	Paints, Polygon

Sankey Diagram	Circuit, Tile, Mosaic
Scatterplot	Dots, Clusters
Streamgraph	Mountain and its reflection
Tree Diagram	Maze, Grid
Treemap	Partitions, Rectangles
Venn Diagram	Cells, Paints
Windrose Chart	Unrecognizable

Table 4.1: Most common responses from the participants when they were asked *"What are the concepts shared by these images?"*

Table 4.1 shows the most common responses from the participants. Most participants commented that these images are more like contemporary art than a chart or a diagram. Participants with expertise in information visualization were reluctant to use terminology such as *axis* to describe the images, because the images were very different from traditional vis idioms.

Although PP-BEGAN was not able to generate realistic images of vis idioms with clear visual encoding, the responses represent the morphological similarity between the generated samples and real images of vis idioms. Such similarity serves enough to understand the intuition behind the creation of vis idioms, such as *"Scatterplots use proximity encoding along the position encoding."* (as most scatterplot samples had clusters), or *"Donut charts require an arc, not a ring, to represent a single percentile value"* (as near half of donut chart samples were rendered in an arc shape rather than a ring shape).

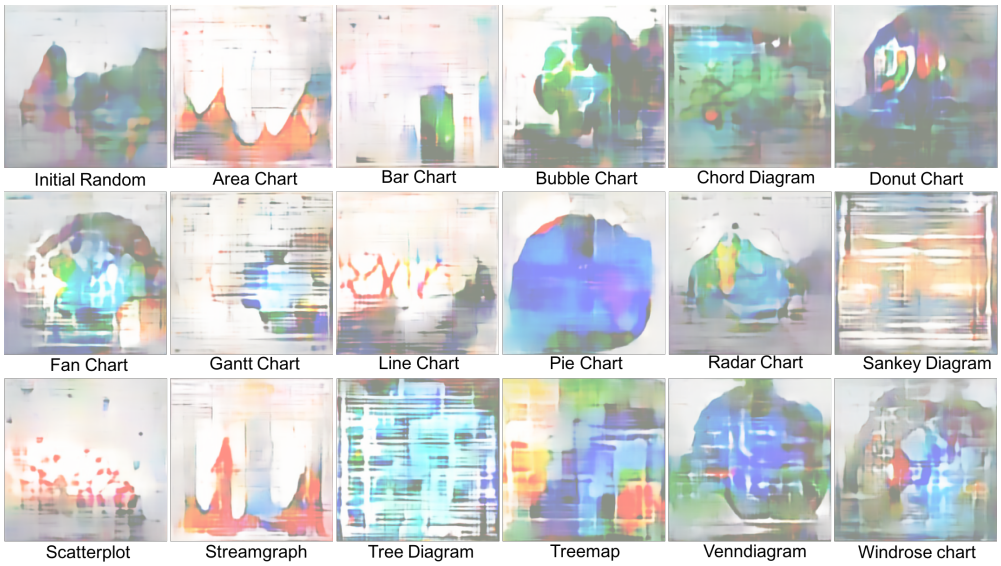

Figure 4.4: Preferred images of 17 chart types sampled with MALA with the given chart type classifier and the PP-BEGAN. (ϵ_1, ϵ_2, ϵ_3) = (1e-5, 1, 1e-15). Optimized via Adam with a learning rate of 0.01.

4.3 Walking in the latent space

Radford et al. [32] suggested an examination called *walking in the latent space*, which is to figure out whether the latent space accurately maps the data distribution. The principle is simple; if the generated sample changes smoothly along the interpolation of two latent vectors, then the latent space decently captures the data distribution. I apply this examination to PP-BEGAN and exhibit a 2D interpolation of four latent vectors for a bar, area, bubble chart and scatterplot [Figure 4.7]. As shown in the figure, the generator seems to perfectly capture the data distribution.

If a model is capable of doing linear interpolation of the latent vectors, as a corollary it is capable of making a visual analogy. Following previous researches

Figure 4.5: Preferred images of bubble chart those sampled via MALA with the given chart type classifier and the PP-BEGAN. (ϵ_1, ϵ_2, ϵ_3) = (1e-5, 1, 1e-15). Optimized with four different optimizers (Adam with learning rate of 1, 0.1, 0.01, SGD with learning rate of 1), and mini-batched with three different batch sizes (1, 4, and 9).

[33, 20], in this paper, the term *analogy* is defined as a process of adding and subtracting the relationship between the items to a new item. Figure 4.8 shows four additive analogies made upon a bar, bubble, line chart and treemap plus streamgraph. Visual analogies on visualization idioms would be able to assist human designers by presenting a rough blueprint of cognitive queries such as *"What would it be like to combine the characteristics of a treemap and a streamgraph?"*, of which the answer is presented in figure 4.8. Pairwise samples are shown in figure 4.9, note that pairwise results only show single aspect of in-

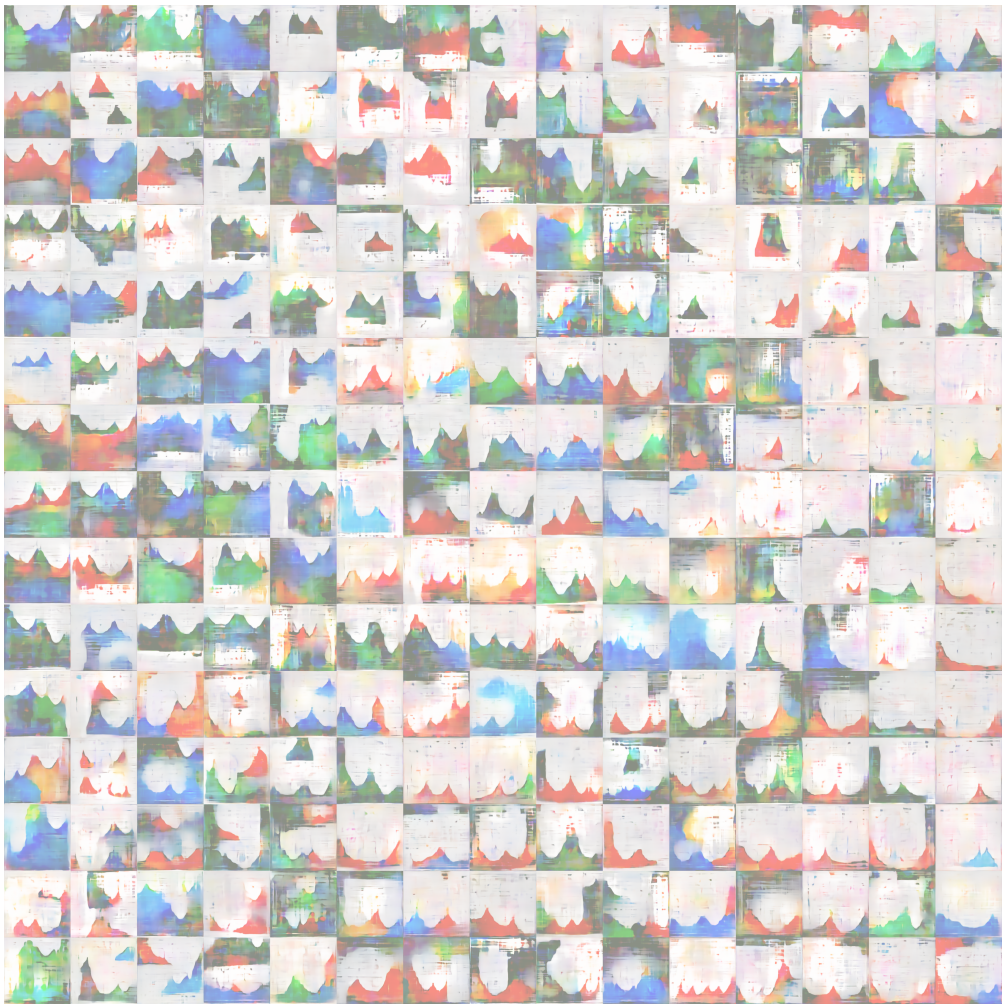

Figure 4.6: 225 area chart samples generated by PP-BEGAN. All were sampled with MALA within 200 iterations. This grid was made by the hungarian method [19] upon t-distributed stochastic neighbor embedding (t-SNE) [23] of `fc6` activations of each image.

terpolation, in practice, designer could explore various interpolated images as shown in 4.6.

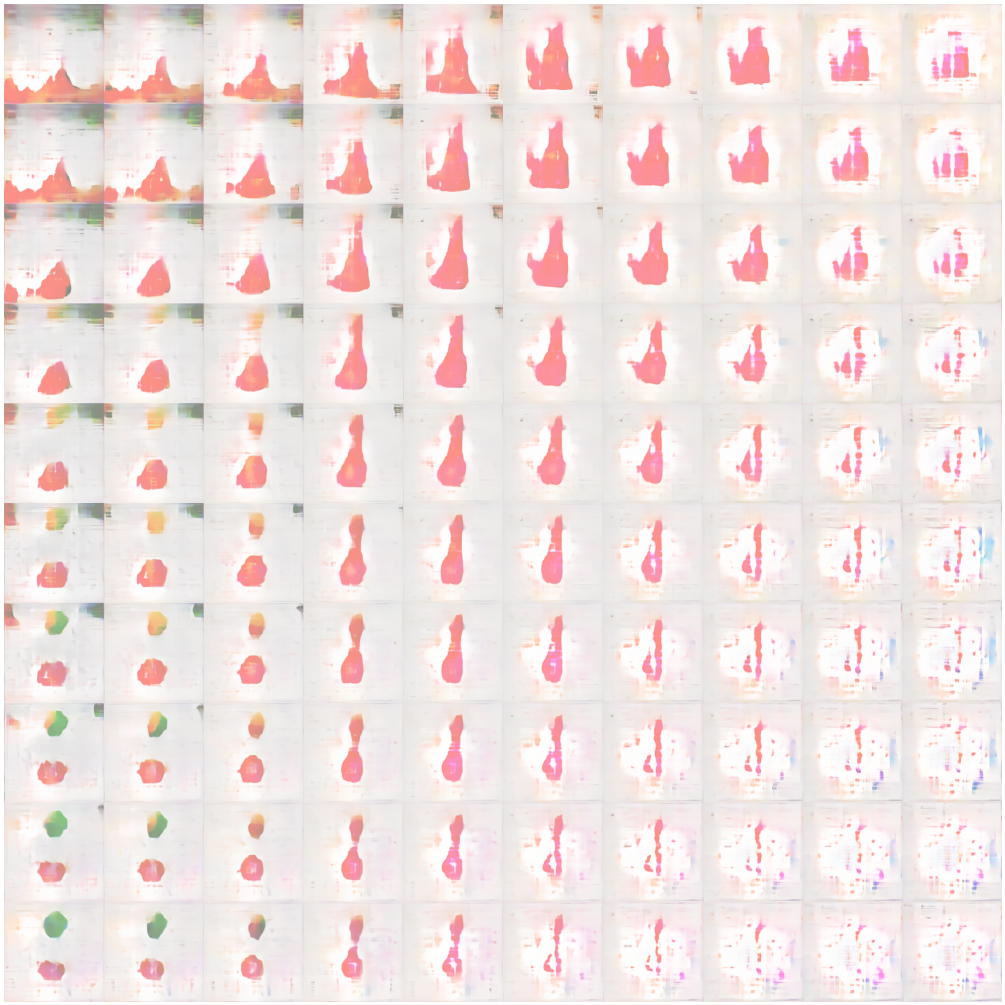

Figure 4.7: Interpolation of four latent vectors representing an area chart (upper left), bar chart (upper right), bubble chart (lower left), and scatterplot (lower right)

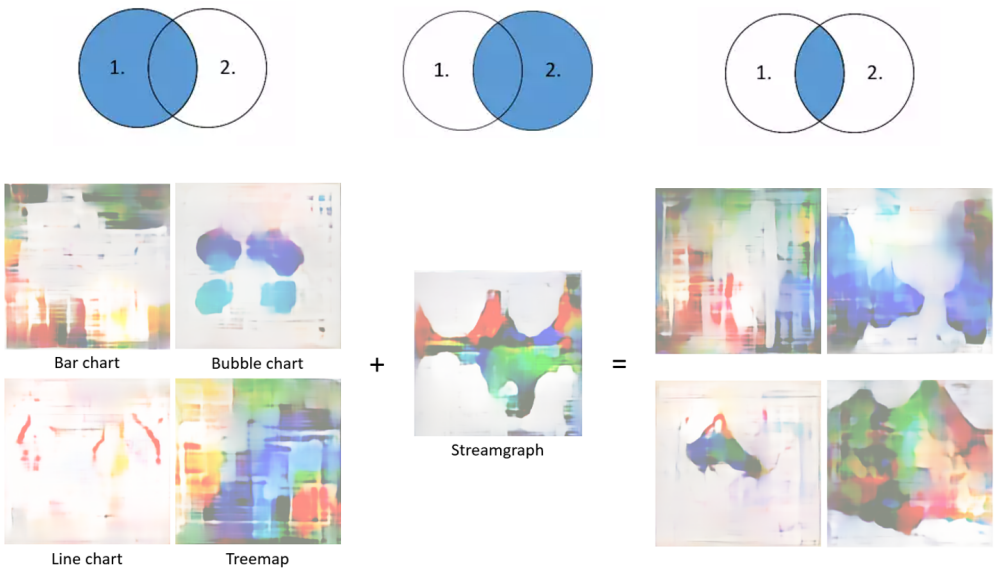

Figure 4.8: Analogy making of streamgraph plus [bar chart, bubble chart, line chart, treemap]

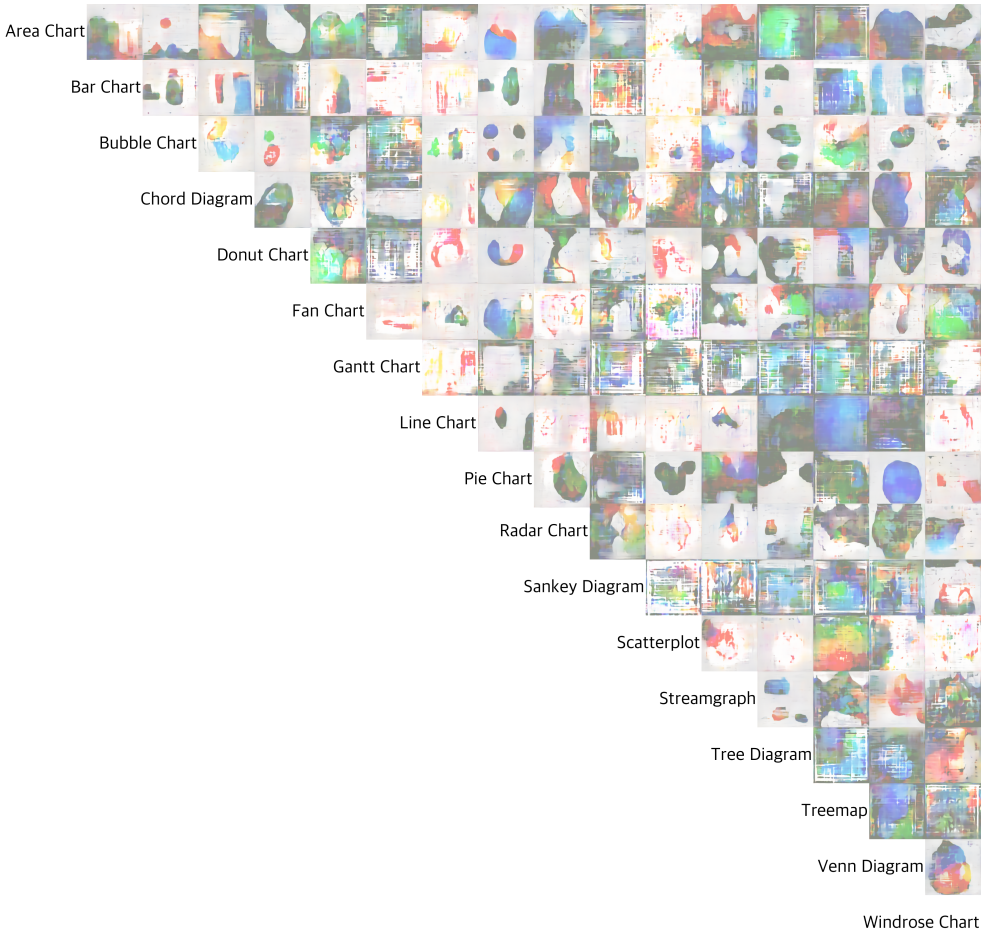

Figure 4.9: Pairwise analogy makings of all 17 vis idiom types.

Chapter 5

Discussion and Future Work

I now discuss the limitations of this paper and possible future work. First of all, the comparison of noiseless joint PPGN-h and PP-BEGAN was only made upon bitmap images of vis idioms. Further research should be made on natural images such as ImageNet and CelebA. Second, switching the classifier $\mathcal{C}$ and the encoder $\mathcal{E}$ to networks with a higher feature extraction capability than AlexNet — GoogleNet [40], VGGNet [37], and ResNet [11] for example — might improve the quality of generative model. Third, the values of the data points that visual marks and channels should represent are still entangled in the latent space of PP-BEGAN. I believe adding a variation information maximization term [2, 4] to PP-BEGAN would help the model to disentangle discrete and continuous information. Fourth, collecting more vis idiom images would greatly improve the quality of samples, since the current corpus is very small compared to ImageNet (about two hundred times smaller).

In this paper, PP-BEGAN is used as a tool for analyzing vis idioms. This architecture is encoder-agnostic as the original PPGNs are. Thus, PP-BEGAN is capable of what original PPGNs can do, such as impainting an image or generating an image from a descriptive sentence. Such applications should be investigated in future work. Moreover, since PP-BEGAN extends the application domain from natural images to synthetic images, applications on various

non-photo images including medical images would greatly help domain experts interpret their neural models.

Chapter 6

Conclusion

This paper delivers important contributions to both the information visualization and the machine learning community. For the Information visualization community, the model (both the classifier and the generator) gives a clear data-driven answer for what type the given vis idiom is, and statistically what visual encodings form the idiom. Such generalization of data is called *data extrapolation*. Moreover, the model *interpolate* the vis idioms, creating a novel sketch of vis idioms that correspond to the intersection of the given vis idioms. Extrapolation and interpolation could greatly help vis researchers clarify the vague design space of vis idioms and develop a novel vis idiom.

For the machine learning community, this paper suggests a modification for the state-of-the-art generative model plug and play generative networks by using boundary equilibrium generative adversarial networks. Although the comparison was made upon a in very small synthetic corpus, the model showed promises of stable training of deep generative networks for activation maximization. Moreover, I compared several different optimization techniques and batch sizes for sampling the preferred stimuli, and found that mini batch sampling yields more stable results.

Generating preferred stimuli is very important; it is not only a versatile conditional generative model, but also a cornerstone for interpretable machine

learning. In this paper, I used the model to discriminate vis idioms, but it could also be used for other visualization communities such as Visual Analytics Science and Technology (VAST) which recently adopted convolutional neural networks in their applications. Acquiring interpretability through deep visualization is crucial for areas that do need explanation. Thus, further work is needed to fully appreciate the capability of PP-BEGAN.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] David Barber and Felix Agakov. The im algorithm: a variational approach to information maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, pages 201–208. MIT Press, 2003.

[3] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[5] William S Cleveland and Robert McGill. An experiment in graphical perception. *International Journal of Man-Machine Studies*, 25(5):491–500, 1986.

[6] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341:3, 2009.

[8] Pascal Getreuer. Rudin-osher-fatemi total variation denoising using split bregman. *Image Processing On Line*, 2:74–95, 2012.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] Robert B Haber and David A McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *Visualization in scientific computing*, 74:93, 1990.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM, 2010.

[13] Weidong Huang, Peter Eades, and Seok-Hee Hong. Measuring effectiveness of graph visualizations: A cognitive load perspective. *Information Visualization*, 8(3):139–152, 2009.

[14] Ohad Inbar, Noam Tractinsky, and Joachim Meyer. Minimalism in information visualization: attitudes towards maximizing the data-ink ratio.

In *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!*, pages 185–188. ACM, 2007.

[15] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. Chartsense: Interactive data extraction from chart images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6706–6717. ACM, 2017.

[16] James D Kelly. The data-ink ratio and accuracy of newspaper graphs. *Journalism Quarterly*, 66(3):632–639, 1989.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[20] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[21] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

[22] Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56, 2017.

[23] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[24] Jock Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141, 1986.

[25] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[26] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[27] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[28] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[29] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[30] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[31] Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[33] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in neural information processing systems*, pages 1252–1260, 2015.

[34] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics*, 23(1):61–70, 2017.

[35] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[36] Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[39] H Suzuki, T Matsumoto, and Leon O Chua. A cnn handwritten character recognizer. *International journal of circuit theory and applications*, 20(5):601–612, 1992.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[41] Tijmen Tieleman and Geoffrey Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1033–1040. ACM, 2009.

[42] John W Tukey et al. Data-based graphics: visual display in the decades to come. *Statistical Science*, 5(3):327–339, 1990.

[43] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[44] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

국문초록

시각화 이디엄은 시각적인 표현이 어떻게 만들어지고 조작되어지는지에 대한 정의이다. 널리 알려진 시각화 이디엄에는 바 차트나 파이 차트같은 차트들이 있다. 시각화에 대한 연구와 응용을 하기 위해서는 이러한 시각화 이디엄에 대한 철저한 이해가 선행되어야 한다. 시각화 이디엄을 위한 도구로서, 저자는 딥 비주얼라이제이션을 기용한 새로운 접근법을 제시한다. 딥 비주얼라이제이션은 심층 신경망의 이해를 위해 그 신경망의 뉴런의 성질을 하나의 이미지로 시각화하는 방법들의 총칭이다. 본 논문에서, 저자는 시각화 이디엄의 타입을 분류하는 신경망과 주어진 타입에 따른 시각화 이디엄 이미지를 만드는 신경망, 두 가지 신경망을 제시한다. 생성 신경망을 통해 분류 신경망에 딥 비주얼라이제이션을 적용함으로서, 저자는 딥 비주얼라이제이션이 어떻게 시각화 연구자들이 새롭게 시각화 이디엄의 다양한 측면에 대해 이해하는 것을, 또 어떻게 아직 고려되지 않은 이디엄을 찾아내는 것을 도울 수 있는지에 대해 살펴본다.

주요어: 정보 시각화, 시각화 이디엄, 딥 러닝, 딥 비주얼라이제이션, 생성 모델
학번: 2016-21192