Title
Conditional Divergence Triangle for Joint Training of Generator, Energy-based and Inference Models

Author
Zhu, Shuai

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Conditional Divergence Triangle for Joint Training of Generator, Energy-based and Inference Models

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

by

Shuai Zhu

2019

ABSTRACT OF THE THESIS

Conditional Divergence Triangle for Joint Training of
Generator, Energy-based and Inference Models

by

Shuai Zhu
Master of Science in Statistics
University of California, Los Angeles, 2019
Professor Yingnian Wu, Chair

This paper proposes a conditional version of Divergence Triangle [1] as a framework to train generator, energy-based and inference models jointly with the information of labels, where the learning of the above three models are integrated perfectly in a unified probabilistic formulation. Experiments demonstrate that, within this one framework, we are able to complete the following tasks together, (1) control the fine-grained categories to generate realistic images, (2) obtain the meaningful representation of observed data in the low dimensions, and also (3) conduct label classification on unobserved data. Additionally, I also discuss a possible extension on Conditional Divergence Triangle model at the end of this paper for future work.

The thesis of Shuai Zhu is approved.

Hongquan Xu

Jingyi Li

Yingnian Wu, Committee Chair

University of California, Los Angeles

2019

To my mother . . .

who always believes in me and

supports all the dreams I have

TABLE OF CONTENTS

1 Introduction **1**

2 Representing models with label information **4**

2.1 Generator model representation 4

2.2 Energy-based model representation 5

2.3 Inference model representation 6

2.4 Brief summary 7

3 Conditional Divergence Triangle Learning **8**

3.1 Generator model learning 8

3.2 Energy-Based model learning 9

3.3 Inference model learning 11

3.4 Objective function for joint learning three models 11

3.5 Training algorithm 13

4 Experiments **16**

4.1 Generating images in fine-grained categories 16

4.2 Obtaining meaningful lower-dimensional representation 23

4.3 Label classification on unobserved data 25

5 Conclusion and Future Work **27**

A Network Structures **28**

References **31**

LIST OF FIGURES

2.1 Modified generator model diagram 5
2.2 Modified energy-based model diagram 6
2.3 Modified Inference model diagram 6

3.1 Conditional Divergence Triangle learning diagram 13
3.2 Diagram for Training Algorithm 14

4.1 Training examples of MNIST 17
4.2 Generated results from Conditional GAN 18
4.3 Generated results from Conditional Divergence Triangle 18
4.4 Training examples from Fashion-MNIST 19
4.5 Generated Fashion-MNIST results from Conditional GAN 20
4.6 Generated Fashion-MNIST results from Conditional Divergence Triangle 20
4.7 Training examples from Cifar10 21
4.8 Synthesized conditional results from Conditional Divergent Triangle 22
4.9 Conditional Linear interpolation in z for MNIST 23
4.10 Linear interpolation in z for Fashion-MNIST 24
4.11 Linear interpolation in z for Cifar-10 24

LIST OF TABLES

4.1 Comparisons of classification accuracy between different algorithms 25

A.1 Network structure of Generator Model for MNIST & Fashion-MNIST 28

A.2 Network structure of Energy-based Model for MNIST & Fashion-MNIST 28

A.3 Network structure of Inference Model for MNIST & Fashion-MNIST 29

A.4 Network structure of Generator Model for Cifar 10 29

A.5 Network structure of Energy-based Model for Cifar 10 30

A.6 Network structure of Inference Model for Cifar 10 30

CHAPTER 1

Introduction

Learning generative models of images is an essential problem in computer vision, and the goal is to construct flexible models to fit complex data distributions as well as enable people to generate highly realistic samples from those distributions conveniently.

Energy-based models [2, 3, 4, 5, 6] and latent variable models [7, 8, 9, 10] are thought to be two major classes for the family of generative models; furthermore, the latent variable models can also be divided into the generator model for image generation and inference model to infer the latent vector from the image. Previous researches have demonstrated that learning the energy-based and generator models together will benefit each other [11, 12]. A metaphor might be used to describe this cooperative relationship between those two models: a company hires an entry-level analyst, and he finishes a report to let his supervisor revise. After that, they send this revised report to their manager, and the manager provides some feedback to the supervisor so that the supervisor know where to improve; then, the supervisor will let the analyst modify this report. Hence, the analyst will learn from his supervisor, while the supervisor learns from the manager. In this metaphor, the supervisor will provide guidance to the analyst, but most of the work is done by this analyst. Actually, the analyst and supervisor correspond to the generator model and energy-based model respectively. For the generator model, the image is treated as a transformation of the latent variable whose prior distribution is known, and we can generate an image by transforming a sampling latent vector into a higher dimension; for the energy-based model, an energy function can be defined as a mapping from an image to its energy value, where the energy value might be understood as a combination of various feature statistics of the image. Hence, if images come from the same distribution, they should have similar energy values; otherwise, their energy values suppose to be different. In this respect, the energy-based model is similar to a discriminator, which

can distinguish the generated images from the real ones as well as provide the feedback to the generator model.

Both the generator model and energy-based model can be parameterized by the deep neural networks. However, learning the two models through maximum likelihood estimation (MLE) will involve intractable integrals that have to use expensive Markov chain Monte Carlo (MCMC) to approximate. In order to get rid of the costly MCMC approximation, the original Divergence Triangle [1] is proposed, and it introduces an inference model which defines an explicit approximation to the posterior distribution of the latent variables. With this inference model, we are able to approximately maximize the likelihood of models through directly minimizing the upper bound of Kullback–Leibler divergences. Hence, without the need for expensive MCMC methods, the original Divergence Triangle can make the processes of image samplings, energy computation as well as feed-forward inference readily available by training the above three models jointly.

Although the original Divergence Triangle framework is able to generate highly realistic images without using MCMC, it cannot decide the mode of images being generated since the models involve no extra information y, such as class labels; hence, the motivation of Conditional Divergence Triangle framework is to control the categories of the generated results through conditioning models on the label information. Aiming for an easier understanding, we need to clarify the meaning of "conditional" in my approach. The term "conditional" means that both my generator model and energy-based model will be conditioned on the label y, and we can perform this conditioning by feeding y, as an additional input layer, into the models. The ideal result is when we input certain label information and the sampling latent vectors together into the well-learned generator model, only the images under that specific category will be generated, which cannot be achieved by the original Divergence Triangle. Thus, I believe the generation process can be directed by adding information of labels into the models for the Conditional Divergence Triangle framework.

As we have the label information, the classification task can also be completed using the well-learned inference model within the framework of Conditional Divergence Triangle, which is considered to be an expanded functionality. Besides, we also discover some meaningful relationships between sampling latent vectors and the generated images through experiments, indicating the inferred latent vectors in the low dimension reflect the patterns of images in the high dimension.

My major contributions include the following:

- Enable the original Divergence Triangle [1] to generate fine-grained images with specific label information (condition).
- Compelete conditional image generation, image classification and meaningful dimensionality reduction within in one unified probabilistic framework.

CHAPTER 2

Representing models with label information

The original Divergence Triangle is an unsupervised framework for joint training of energy-based model, generator model and inference model, where both generator and inference models belong to the latent variable family. In this chapter, let's go through the above three probabilistic models and also talk about how to represent them with label information.

2.1 Generator model representation

The generator model [2, 3, 4, 5, 6] can be formulated as below.

$$z \sim N(0, I_d), \quad x = g_\theta(z) + \epsilon \tag{2.1}$$

where z is a d-dimensional signal latent vector sampled from the normal distribution $N(0, I_d)$, and g_θ, a deep network parameterized by parameters θ, is a top-down mapping function to map z into D-dimensional signal x, where x usually represents an image; ϵ denotes the noise that follows the normal distribution $N(0, \sigma^2 I_D)$; hence, $x|z$ will follow $N(g_\theta(z), \sigma^2 I_D)$ which is a normal distribution as well. In general, the observed-data model is $p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p(z)dz$, where $p(z)$ is a prior distribution and $p_\theta(x|z)$ can be expressed by the deep network g_θ.

Supposing we introduce the label information y, we may modify the generator model as below:

$$z \sim p(z), \quad y \sim p(y), \quad x = g_\theta(z, y) + \epsilon \tag{2.2}$$

where z, a latent vector, follows prior distribution $p(z)$, ϵ is the noise and y, independent of z, follows the label distribution $p(y)$; in the meanwhile, g_θ is still a deep network except treating both z and y as its inputs. We can write the observed-data model as $p_\theta(x) = \int p_\theta(x, z)dz =$

$\int\int p_\theta(x,z,y)dydz = \int\int p_\theta(x|z,y)p(z)p(y)dydz$, where the generator g_θ denotes $p_\theta(x|z,y)$. See the diagram below:

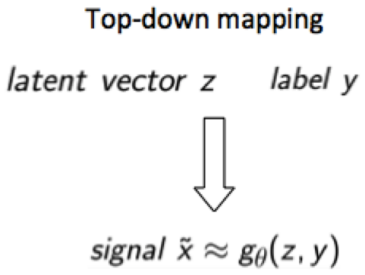

Figure 2.1: Modified generator model diagram

2.2 Energy-based model representation

For the energy-based model [7, 8, 9, 10], we can define the following probability model

$$\pi_\alpha(x) = \frac{1}{Z(\alpha)} \exp[f_\alpha(x)] \tag{2.3}$$

where f_α is the energy function parametrized by a bottom-up deep neural network with parameters α, $f_\alpha(x)$ represents the energy value (scalar) of image x, and $Z(\alpha)$ is a normalizing constant to make $\pi_\alpha(x)$ a probability. Therefore, $\pi_\alpha(x)$ can be used to model the probability of image x, the real images should have high energy $f_\alpha(x_{real})$ and fake images possess low energy $f(x_{fake})$; in this way, high probabilities can be assigned to those real images and low probabilities go to the fake ones.

If we introduce the label information y, we can modifiy the energy-based model as below:

$$\pi_\alpha(x,y) = \frac{1}{Z(\alpha)} \exp[f_\alpha(x,y)] \tag{2.4}$$

where we consider (x,y) as paired data, and $\pi_\alpha(x,y)$ should only assign high probability to those real images whose labels are exactly y, which means x and y have to be matched; otherwise, π_α should learn to assign low probabilities through energy function f_α. Although $Z(\alpha)$ is intractable, the energy-based model still defines an explicit likelihood via $f_\alpha(x,y)$. It is hard to sample from π_α due to this intractable issue; however, with help of the other models, we can avoid this sampling step in the energy-based model. See the diagram of energy-based model below:

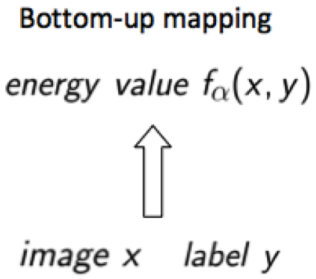

Figure 2.2: Modified energy-based model diagram

2.3 Inference model representation

The original inference model refers to the mapping function from image x to latent vector $\tilde{z}$. We can formulate this inference model as $\tilde{z} = I_\phi(x)$, where I_ϕ is a bottom-up mapping deep neural network parametered with parameters ϕ, and $\tilde{z}$ is an inferred latent vector. We want that the inferred $\tilde{z}$ also follows the prior distribution $p(z)$, which can be constrainted by the Kullback-Leibler divergence.

For the Conditional Divergence Triangle, we let the deep network I_ϕ use image x to infer both $\tilde{z}$ and $\tilde{y}$ at the same time ($\tilde{z}$ and $\tilde{y}$ share the body of the nerual network), where the inferred $\tilde{y}$ can be seen as the predicted labels. Thus, the inference model can be written as $p_\phi(z, y|x)$. Since $\tilde{y}$ and $\tilde{z}$ should be independent, we can also write $p_\phi(z, y|x) = p_\phi(z|x)p_\phi(y|x)$, where $p_\phi(y|x)$ represents a classifier. See the diagram of inference model below:

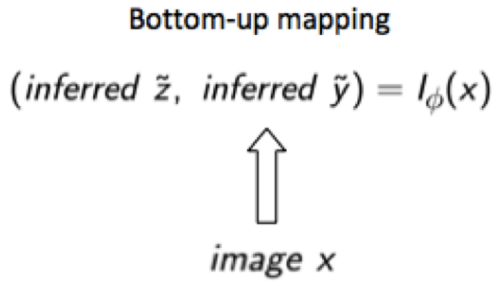

Figure 2.3: Modified Inference model diagram

2.4 Brief summary

With the label information y, we can interpret the generator, energy-based and inference models in the probabilistic language as:

- Generator model: $p_\theta(x|z, y)$.
- Energy-based model: $\pi_\alpha(x, y)$
- Inference model: $p_\phi(z, y|x) = p_\phi(z|x)p_\phi(y|x)$

With the probabilistic representations of the three models, we will discuss how to use the Conditional Divergence Triangle to jointly learn those three models in the next chapter.

CHAPTER 3

Conditional Divergence Triangle Learning

In this chapter, we will formally present the Conditional Divergence Triangle framework and focus on more details about the learning process.

3.1 Generator model learning

For the generator model, it's trivial to sample from latent distribution $p(z)$ and label distribution $p(y)$; we can define generative process as following: $z \sim p(z)$, $y \sim p(y)$ and $\tilde{x} \sim p_\theta(x|y,z)$. Since $p(y)$ and $p(z)$ are independent, the parameters θ can be learned by the maximum likelihood of $p_\theta(x)$, where $p_\theta(x) = \int\int p_\theta(x|z,y)p(y,z)dydz = \int\int p_\theta(x|y,z)p(y)p(z)dydz$.

Assume we observe paired training data $\{(x_i, y_i) \sim p_{data}(x,y)\}_{i=1}^n$, where $p_{data}(x,y)$ represents the real joint distribution of images and labels. Hence, we will have the following loglikelihood function,

$$l(\theta) = \frac{1}{n}\sum_{i=1}^{n} log p_\theta(x_i) \tag{3.1}$$

In order to maximize this $l(\theta)$, we calculate its derivative,

$$\begin{aligned}
l^{'}(\theta) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_\theta(x_i)}\frac{\partial}{\partial\theta}p_\theta(x_i) \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_\theta(x_i)}\frac{\partial}{\partial\theta}\int\int p_\theta(x_i|y_i,z_i)p(y_i)p(z_i)dy_i dz_i \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_\theta(x_i)}\int\int \frac{\partial}{\partial\theta}p_\theta(x_i,y_i,z_i)dy_i dz_i \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{p_\theta(x_i)}\int\int \frac{\partial}{\partial\theta}log(p_\theta(x_i,y_i,z_i))dy_i dz_i \times p_\theta(x_i,y_i,z_i)
\end{aligned}$$

$$= \frac{1}{n}\sum_{i=1}^{n} E_{p_\theta(y_i,z_i|x_i)}[\frac{\partial}{\partial\theta} log(p_\theta(x_i,y_i,z_i))] \tag{3.2}$$

If we want to continue the above updates, we have to compute the expectation corresponding to the posterior distribution $p_\theta(y,z|x)$, which is analytically intractable. One way to solve this problem is applying computational expensive MCMC, such as Langevin dynamics or HMC [13], to approximate. Another way is to use the inference model $p_\phi(y,z|x)$ to estimate $p_\theta(z,y|x)$ as in variational auto-encoder(VAE) [14]. The objective of VAE is to minimize $KL(p_{data}(x)||p_\theta(x)) + KL(p_\phi(z|x)||p_\theta(z|x))$, and in our case, we need modify θ and ϕ to minimize our generator objective as below:

$$KL(p_{data}(x)||p_\theta(x)) + KL(p_\phi(z,y|x)||p_\theta(z,y|x)) = KL(A_\phi||B_\theta) \tag{3.3}$$

where we define A-distribution as $A_\phi = p_{data}(x)p_\phi(z,y|x) = p_\phi(x,y,z)$ and B-distribution as $B_\theta = p_\theta(x)p_\theta(z,y|x) = p_\theta(x,y,z) = p(z)p(y)p_\theta(x|y,z)$.

Actually, maximizing the loglikelihood $l(\theta)$ is equivalent to minimizing $KL(p_{data}(x)||p_\theta(x))$ when n is large enough. Since KL divergence is non-negative, we know $KL(p_{data}(x)||p_\theta(x)) \leq KL(A_\phi||B_\theta)$, which is an upper bound. Similar to VAE, we may choose to minimize the upper bound $KL(A_\phi||B_\theta)$ instead of directly minimizing $KL(p_{data}(x)||p_\theta(x))$ to learn the generator model.

3.2 Energy-Based model learning

For the energy-based model, we have $\pi_\alpha(x,y) = \frac{1}{Z(\alpha)}\exp[f_\alpha(x,y)]$. We can use the paired sample $\{(x_i,y_i) \sim p_{data}(x,y)\}_{i=1}^n$ to learn the parameter α through MLE, where the loglikelihood $l(\alpha) = \frac{1}{n}\sum_{i=1}^n log\pi_\alpha(x_i,y_i)$ [Note: when n is large enough, maximizing $l(\alpha)$ is equivalent to minimizing

$KL(p_{data}(x,y)||\pi_\alpha(x,y))$]. Taking the derivative of $l(\alpha)$, we have,

$$\begin{aligned} l^{'}(\alpha) &= \frac{\partial}{\partial\alpha}\frac{1}{n}\sum_{i=1}^{n} log\pi_\alpha(x_i,y_i) \\ &= \frac{\partial}{\partial\alpha}\frac{1}{n}\sum_{i=1}^{n} log(\frac{1}{Z(\alpha)}exp[f_\alpha(x_i,y_i)]) \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\alpha}f_\alpha(x_i,y_i) - E_{\pi_\alpha(x,y)}[\frac{\partial}{\partial\alpha}f_\alpha(x,y)] \end{aligned} \tag{3.4}$$

However, due to the normalizing constant $Z(\alpha)$, $\pi_\alpha(x,y)$ is also analytically intractable. We consider to replace $\pi_\alpha(x,y)$ with $p_\theta(x,y) = \int p_\theta(x|y,z)p(y)p(z)dz$; then, $l^{'}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} f_\alpha(x_i,y_i) - E_{p_\theta(x,y)}[\frac{\partial}{\partial\alpha}f_\alpha(x,y)]$, which is equivalent to,

$$\min_\alpha KL(p_{data}(x,y)||\pi_\alpha(x,y)) - KL(p_\theta(x,y)||\pi_\alpha(x,y)) \tag{3.5}$$

For the first term $KL(p_{data}(x,y)||\pi_\alpha(x,y))$ in equation (3.5), we wish to multiply $p(z|x,y)$ on both sides to turn them into joint distributions (The value of KL divergence won't change, if we multiply the same value on both sides); The first idea is to make use of the inference model to find $p_\phi(z|x,y)$; however, our original inference model is $p_\phi(z,y|x) = p_\phi(z|x)p_\phi(y|x)$, which seems different. But since images x and its corresponding label y should be highly correlated, we may consider $p_\phi(z|x,y) \approx p_\phi(z|x)$, which happens to be a part of our original inference model. Therefore,

$$\begin{aligned} KL(p_{data}(x,y)||\pi_\alpha(x,y)) &= KL(p_{data}(x,y)p_\phi(z|x)||\pi_\alpha(x,y)p_\phi(z|x)) \\ &\approx KL(p_{data}(x,y)p_\phi(z|x,y)||\pi_\alpha(x,y)p_\phi(z|x,y)) \\ &= KL(A_\phi||C_{\alpha,\phi}) \end{aligned} \tag{3.6}$$

where $A_\phi = p_\phi(x,y,z)$ is same as in the generator learning, and $C_{\alpha,\phi} = \pi_\alpha(x,y)p_\phi(z|x,y) = p_{\alpha,\phi}(x,y,z)$; Thus, minimizing $KL(p_{data}(x,y)||\pi_\alpha(x,y))$ is able to be replaced by minimizing $KL(A_\phi||C_{\alpha,\phi})$.

For the second term $KL(p_\theta(x,y)||\pi_\alpha(x,y))$, which is related to both the learning of parameters θ (for generator) and α (for energy-based model), we also wish to change them into the form of KL divergence between two joint distributions. Consider the following equation:

$$KL(p_\theta(x,y)||\pi_\alpha(x,y)) + KL(p_\theta(z|x,y)||p_\phi(z|x,y)) = KL(B_\theta||C_{\alpha,\phi}) \tag{3.7}$$

where the definitions of both B_θ and $C_{\alpha,\phi}$ are the same as above.

For updating the parameters α, we may consider $KL(p_\theta(z|x,y)||p_\phi(z|x,y))$ as a constant, since it has nothing to do with α. When learning the energy-based model, we want to maximize $KL(p_\theta(x,y)||\pi_\alpha(x,y))$; hence, we just need to maximize $KL(p_\theta(x,y)||\pi_\alpha(x,y)) + Constant = KL(B_\theta||C_{\alpha,\phi})$, i.e.,

$$\max_\alpha KL(p_\theta(x,y)||\pi_\alpha(x,y)) \quad \Longleftrightarrow \quad \max_\alpha KL(B_\theta||C_{\alpha,\phi}) \tag{3.8}$$

For updating the parameters θ in the generator model, in addition to what we discussed in Section 3.1, we also need to minimize $KL(p_\theta(x,y)||\pi_\alpha(x,y))$. Since $KL(p_\theta(z|x,y)||p_\phi(z|x,y))$ is non-negative, $KL(p_\theta(x,y)||\pi_\alpha(x,y)) \leq KL(B_\theta||C_{\alpha,\phi})$ is an upper bound; therefore, minimizing the upper bound $KL(B_\theta||C_{\alpha,\phi})$ is equivalent to minimize $KL(p_\theta(x,y)||\pi_\alpha(x,y))$.

3.3 Inference model learning

Actually, the inference model is used as a "bridge" to help us avoid the computational expensive MCMC, and its parameters ϕ are also updated during the learning of generator and energy-based models.

3.4 Objective function for joint learning three models

Combining the above three sections, the objective function of Conditional Divergence Triangle should involve the three joint distributions on (x,y,z) as below:

- A_ϕ-distribution: $A(x,y,z) = p_\phi(x,y,z) = p_{data}(x,y)p_\phi(z|x,y) \approx p_{data}(x,y)p_\phi(z|x)$
- B_θ-distribution: $B(x,y,z) = p_\theta(x,y,z) = p_\theta(x|z,y)p(z)p(y)$
- $C_{\alpha,\phi}$-distribution: $C(x,y,z) = \pi_\alpha(x,y,z) = \pi_\alpha(x,y)p_\phi(z|x,y) \approx \pi_\alpha(x,y)p_\phi(z|x)$

The above three joint distributions over (x,y,z) are modeled from different perspectives. The four distributions $p_{data}(x,y)$, $p_{data}(x)$, $p(z)$ and $p(y)$ are all easy to know, and other distributions

above can also be obtained from the deep neural networks. Hence, we propose to learn the generator model, energy-based model and inference model through optimizing the following objective function $L(\alpha, \theta, \phi)$ for Conditional Divergence Triangle,

$$\begin{aligned} &\max_{\alpha} \min_{\theta} \min_{\phi} L(\alpha, \theta, \phi); \\ &L(\alpha, \theta, \phi) = KL(A_\phi || B_\theta) - KL(A_\phi || C_{\alpha,\phi}) + KL(B_\theta || C_{\alpha,\phi}) \end{aligned} \tag{3.9}$$

For the above objective function, we can also explain it from another perspective. Intuitively, we want the above three joint distributions A_ϕ, B_θ and $C_{\alpha,\phi}$ get closer after jointly training the three models.

First, we consider minimizing ϕ. We know $KL(A_\phi||C_{\alpha,\phi}) \approx KL(p_{data}(x, y)||\pi_\alpha(x, y))$ from equation (3.6), which has nothing to do with ϕ; hence, $\min_{\phi} L(\alpha, \theta, \phi)$ is approximately equivalent to:

$$\begin{aligned} &\min_{\phi} \left[KL(A_\phi||B_\theta) - KL(A_\phi||C_{\alpha,\phi}) + KL(B_\theta||C_{\alpha,\phi})\right] \\ \approx &\min_{\phi} \left[KL(A_\phi||B_\theta) + KL(B_\theta||C_{\alpha,\phi})\right] \end{aligned} \tag{3.10}$$

It means we update ϕ to push the joint distributions A_ϕ, $C_{\alpha,\phi}$ to get closer to the distribution B_θ.

Then, we consider minimizing θ, and which is equivalent to,

$$\min_{\theta} \left[KL(A_\phi||B_\theta) + KL(B_\theta||C_{\alpha,\phi})\right] \tag{3.11}$$

It indicates we update θ to push the joint distribution B_θ to get closer to both A_ϕ and $C_{\alpha,\phi}$ distributions.

Finally, we consider maximizing α over $L(\alpha, \theta, \phi)$, and it is equivalent to,

$$\max_{\alpha} \left[KL(B_\theta||C_{\alpha,\phi}) - KL(A_\phi||C_{\alpha,\phi})\right] \tag{3.12}$$

Actually, the equation (3.12) is the same as the equation (3.5) for learning the energy-based model, and equation (3.12) implies that the joint distribution $C_{\alpha,\phi}$ tends to chase the distribution A_ϕ (produced by the real data) and stay far away from the distribution B_θ (produced by the generator model), performing like a judge so that the well-learned energy based model could assign higher probability to the real paired data, and lower probability to the "fake" pairs.

In summary, the learning of Conditional Divergence Triangle is based on the three Kullback–Leibler divergences within the three joint distributions on (x, y, z). The updating process is as follows: A_ϕ and $C_{\alpha,\phi}$ try to get close to B_θ; B_θ also tends to chase A_ϕ and $C_{\alpha,\phi}$; $C_{\alpha,\phi}$ seeks to get close to A_ϕ and stay away from B_θ. Note that we update parameters α, ϕ, θ one by one, and other parameters are fixed during the update of one parameter. The figure below illustrates the learning process,

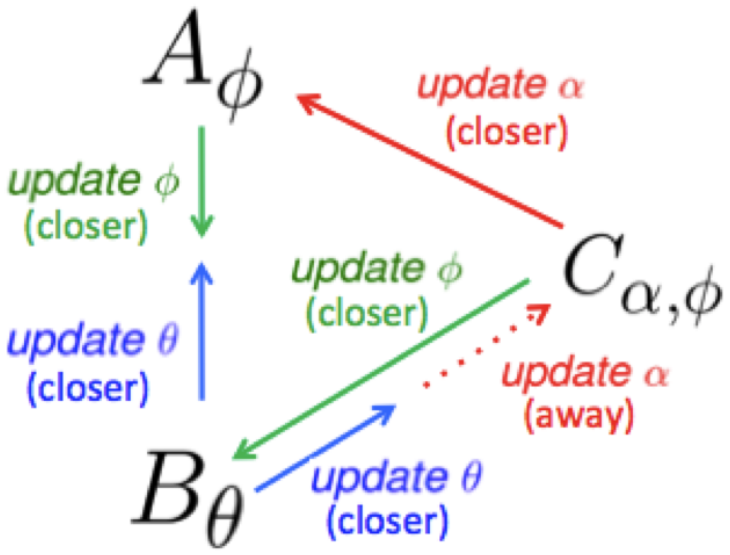

Figure 3.1: Conditional Divergence Triangle learning diagram

3.5 Training algorithm

The generator, energy-based and inference models are all parameterized by deep neural networks. We use stochastic gradient descent to jointly learn the three models within the framework of Conditional Divergence Triangle, and the expectations will be approximated by the sample averages. The figure below shows the procedure of our training algorithm.

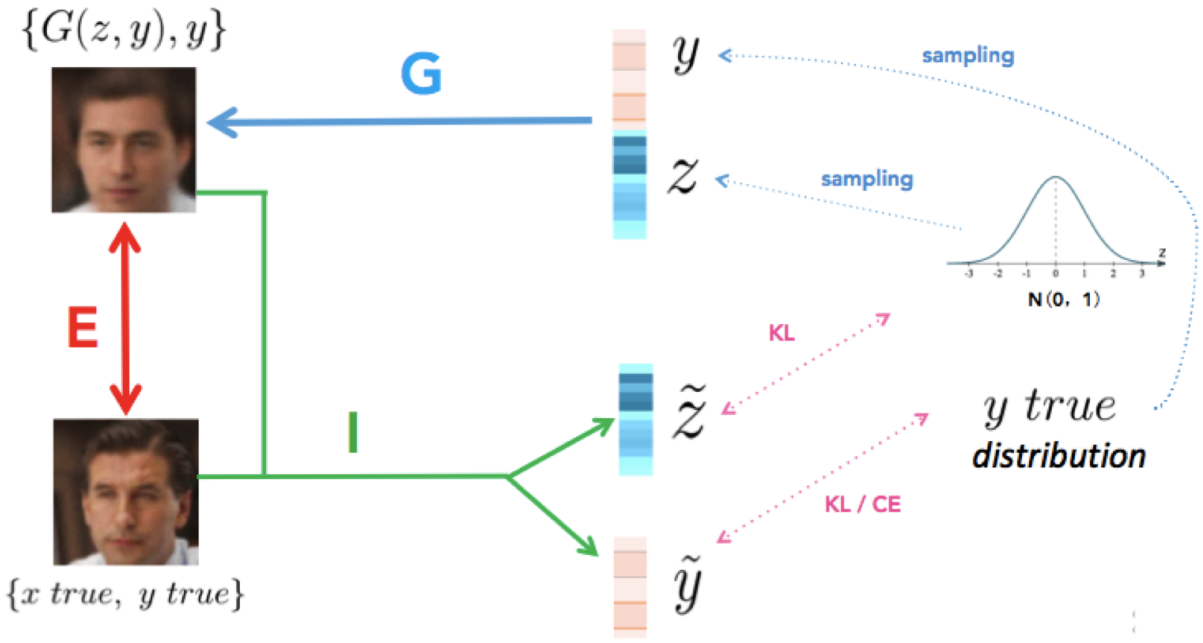

Figure 3.2: Diagram for Training Algorithm

We first sample z and y from their prior distributions, and then use the generator model G : $p_\theta(x|y,z)$ to generate the image $\tilde{x}$. Comparing $(\tilde{x},y)$ with true paired images and labels (x_{true}, y_{true}), we can learn the energy-based model E: $\pi_\alpha(x,y)$. After that, we mix $\tilde{x}$ and x_{true} to infer $\tilde{z}$ and $\tilde{y}$; then, compare $\tilde{z}$, $\tilde{y}$ with their prior distributions $p(z)$ and $p(y)$. In this cycle, we will learn all the parameters for these three networks.

The pseudocode for the training of Conditional Divergence Triangle is provided at below,

Algorithm 1 Joint Training for Conditional Divergence Triangle Model

Input:

1: training images and their corresponding labels $\{(x_i,y_i)\ i=1,...,n\}$

2: testing images $\{x_j$ j=1,...,m$\}$

3: number of training iterations T

4: α, ϕ, θ: initialized network parameters.

Let $t \leftarrow 0$ and define $\mathbf{L}$=$L(\alpha, \theta, \phi)$

repeat

$z_{(1)}, z_{(2)}, ..., z_{(\tilde{n})} \sim p(z)$

$y_{(1)}, y_{(2)}, ..., y_{(\tilde{n})} \sim p(y)$

$\tilde{x}_{(i)} \sim p_\theta(x|z_{(i)}, y_{(i)}), i = 1, ..., \tilde{n}$

$(x_1, y_1),\ (x_2, y_2)\ ..., (x_n, y_n) \sim p_{data}(x, y)$

$\tilde{z}_i, \tilde{y}_i \sim q_\phi(z, y|x_i), i = 1, ..., n$

α **update**: Having $\{(\tilde{x}_{(i)}, y_{(i)}), i = 1, ..., \tilde{n}\}$ and $\{(x_i, y_i), i = 1, ..., n\}$, update $\alpha \leftarrow \alpha - \eta_\alpha \frac{\partial}{\partial \alpha}\mathbf{L}$ using learning rate η_α.

ϕ **update**: Having $\{z_{(i)}, y_{(i)}, \tilde{x}_{(i)}, i = 1, ..., \tilde{n}\}$ and $\{\tilde{z}_i, \tilde{y}_i, x_i, i = 1, ..., n\}$, update $\phi \leftarrow \phi + \eta_\phi \frac{\partial}{\partial \phi}\mathbf{L}$ using learning rate η_ϕ.

θ **update**: Having $\{z_{(i)}, y_{(i)}, \tilde{x}_{(i)}, i = 1, ..., \tilde{n}\}$ and $\{\tilde{z}_i, \tilde{y}_i, x_i, i = 1, ..., n\}$, update $\theta \leftarrow \theta + \eta_\theta \frac{\partial}{\partial \theta}\mathbf{L}$ using learning rate η_θ.

Let $t \leftarrow t + 1$

until $t = T$

classification: Having $\{x_j$, j=1,...,m$\}$, obtain $\tilde{z}_j$, $\tilde{y}_j \sim q_\phi(z, y|x_j)$, and $\tilde{y}_j$ is used as the predicted result for x_j.

Output:

learned parameters α, θ and ϕ

synthesized samples $\{\tilde{x}_i, i = 1, ..., \tilde{n}\}$ given certain condition y_i

classification result $\tilde{y}_j$ on testing images x_j, where j=1,...,m

CHAPTER 4

Experiments

In this chapter, we use experiments to demonstrate that the Conditional Divergence Triangle model is not only able to generate highly realistic images in fine-grained categories, infer meaningful low-dimensional representations, but also conduct label classification on unobserved data.

The only pre-processings needed for the training images are resizing to 32×32 as well as scaling to [-1,1], which is the range of tanh activation function, and the network parameters are initialized with zero-mean normal distribution with standard deviation 0.02 and our optimizer is Adam[15] with decay rate 0.0005; besides, we also use batch normalization[16]. The specific network structures can be found in the Appendix A.

4.1 Generating images in fine-grained categories

In the image generation experiments, we use the well-trained generator model from the Conditional Divergent Triangle framework to generate highly realistic images with given label information. The obtained results from the generator $p_\theta(x|z, y = i)$ should be realistic and similiar to the visualized features of training images under i, where i represents the i^{th} class of the training images.

We first learn our model from MNIST [17] dataset of handwritten digits. The training images are grey-scale with the size of 28×28 pixels, and we resized them to 32×32. For the generator model (a top-down deep network), there are two inputs which are a 100-dimensional latent vector z and a 10-dimensional one-hot vector y. We first up sample both the two inputs to $4 \times 4 \times 256$; then, concatenate them together to make the shape become $4 \times 4 \times 512$. After that, using 4 layers of deconvolution by linear superposition with up-sampling, the number of filters at each layers are

512, 256, 128, 1 respectively, and our output shape is $32 \times 32 \times 1$. In addition, batch normalization, ReLU for non-linearity as well as the tanh activiation function at the bottom-layer are used to make sure signals in the output fall within the range [-1, 1]. For the energy-based model, we have image x $(shape : 32 \times 32 \times 1)$ and label y $(shape : 32 \times 32 \times 10)$, where the y^{th} channel are all ones; while, other channels are all zeros) as our inputs. We first apply 64 filters with stride 2 on each input to make them $16 \times 16 \times 64$; then, we contatenate them together and apply 4 convolutional layers , where the number of channels are 128, 256, 512, 1 from bottom to top. The output of the energy-based network should be a scalar, representing the energy value $f(x, y)$. For the inference model, the input is image x $(shape : 32 \times 32 \times 1)$, and we adopt a 5 layer bottom-up deep network, the channels for the first 4 layers are 128, 256, 512, 1, and the last layer is fully connected with three heads: two 100-dimensional heads respectively represent the mean and standard deviation of the inferred latent vector $\tilde{z}$ for applying reparametrization trick, and the other 10-dimensional head stands for the inferred label $\tilde{y}$.

At below, we show the training MNIST examples, results generated from Conditional GAN [18] and results from our approach in order.

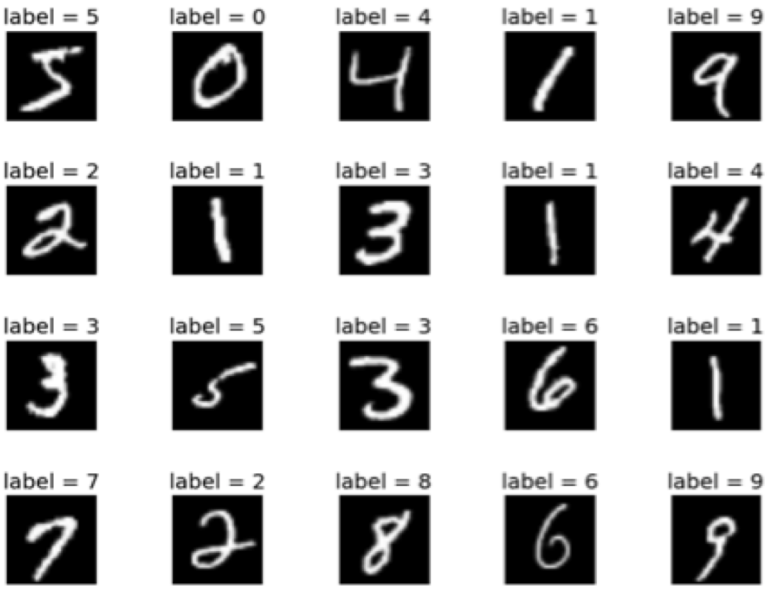

Figure 4.1: Training examples of MNIST

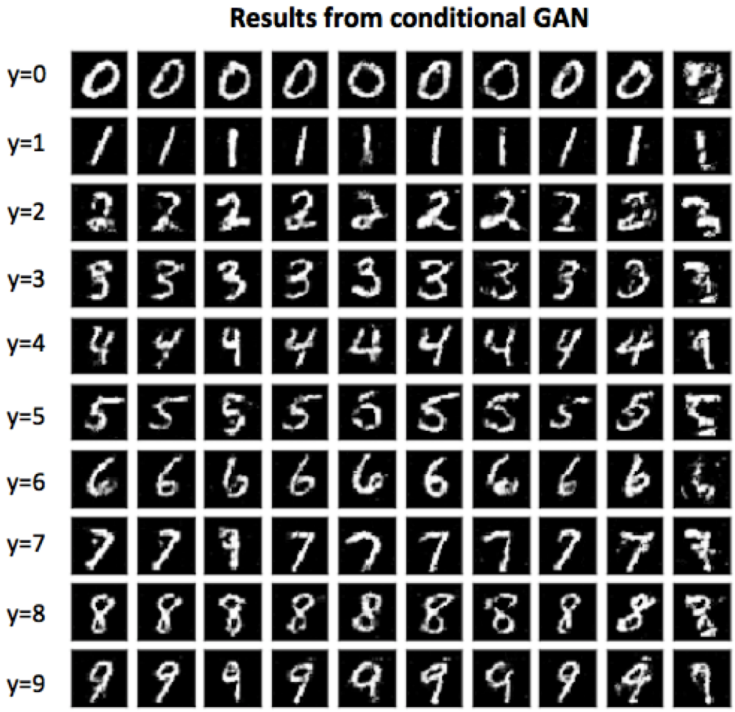

Figure 4.2: Generated results from Conditional GAN

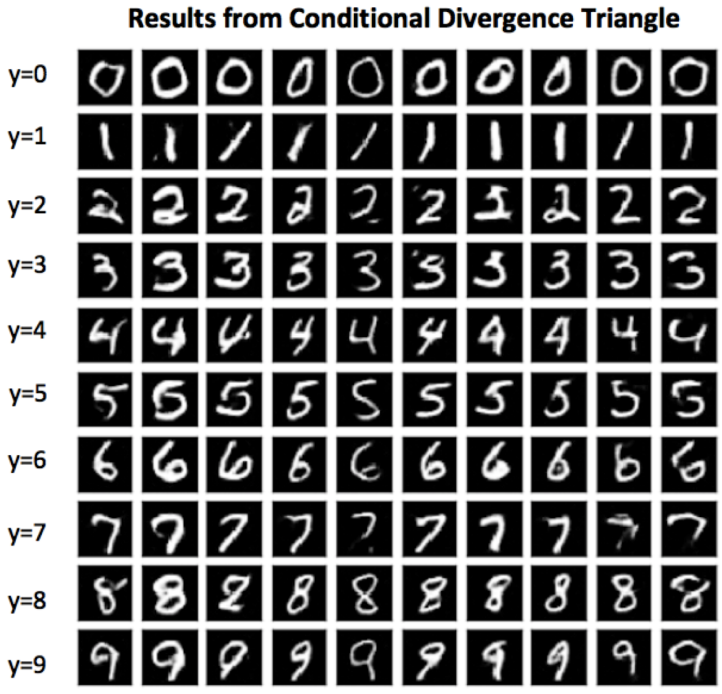

Figure 4.3: Generated results from Conditional Divergence Triangle

Obviously, the synthesized results from our approach (Figure 4.3) are more sharp and clear than the results from Condtional GAN (Figure 4.2). The generated images in each row share the same label vector y (given condition) and in each column share the same latent vector z; hence, we prove our approach is able to generate highly realistic images in fine-grained categories given certain conditions.

In addition to the MNIST, we also try learning from a relatively more complex dataset: Fashion-MNIST [19]. Fashion-MNIST contains 60,000 training examples and 10,000 testing examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes. Similarly, we resize those training images to 32×32 and feed them with labels into the three deep networks described above. After training 30 epoch, we get the synthesized images from our well-trained generator model.

At below, we also show the training Fashion-MNIST examples, results generated from Conditional GAN [18] and results from our approach in order.

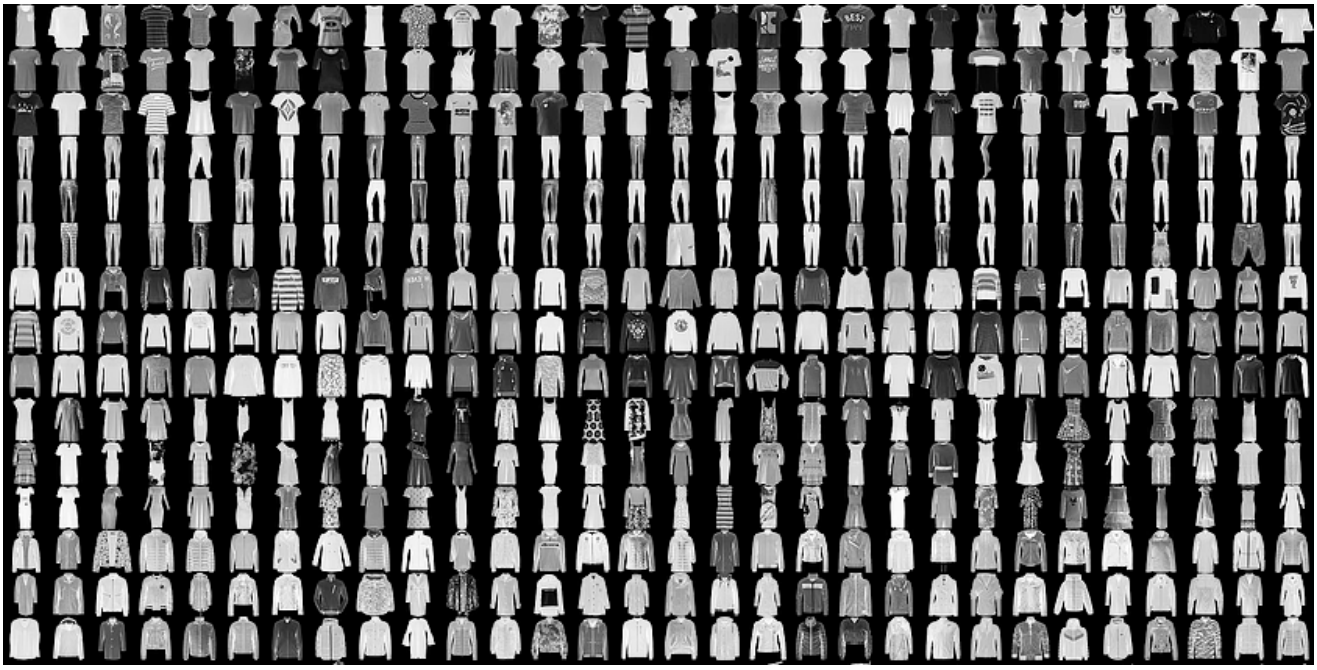

Figure 4.4: Training examples from Fashion-MNIST

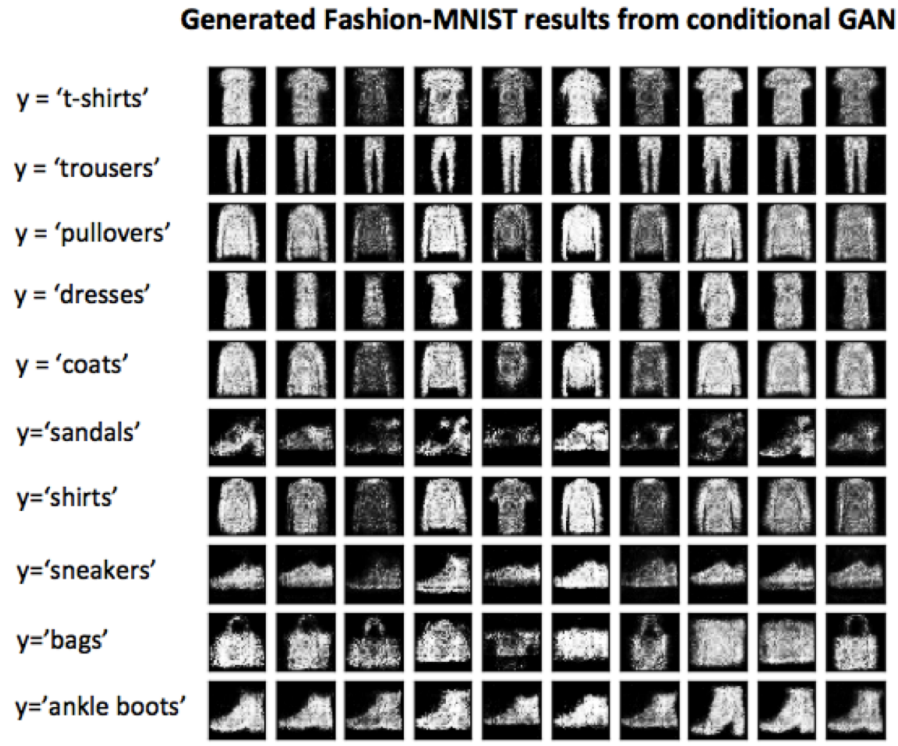

Figure 4.5: Generated Fashion-MNIST results from Conditional GAN

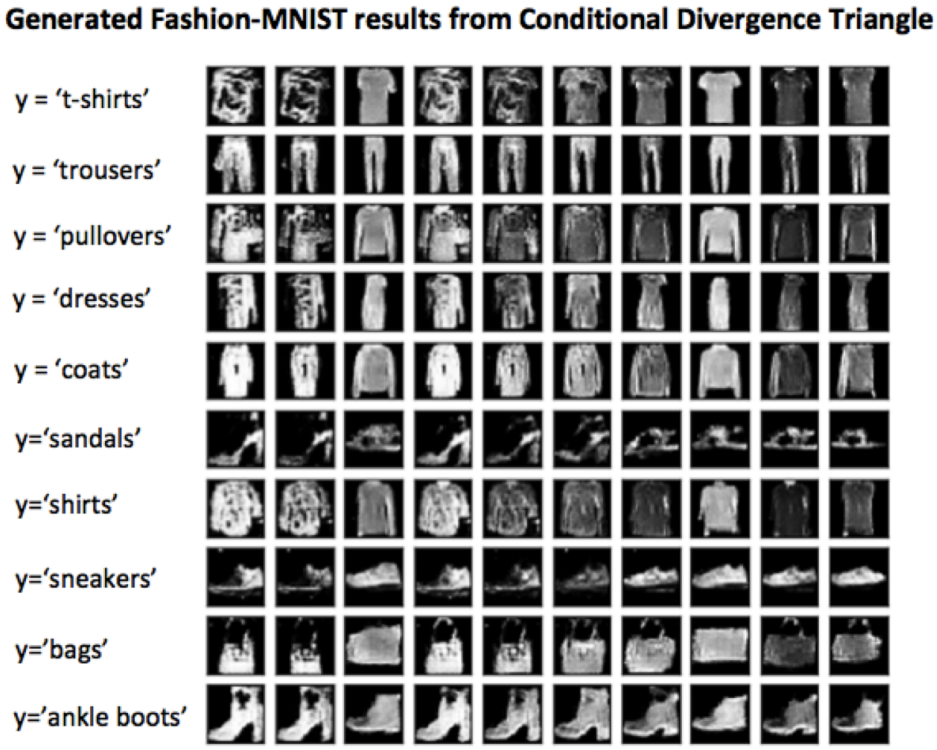

Figure 4.6: Generated Fashion-MNIST results from Conditional Divergence Triangle

Again, the synthesized results on Fashion-MNIST using Condition Divergence Triangle seems better than the results from Conditional GAN. The layout of above images is also same as above, each row shares the same label information and each column shares the same latent vector; hence, the generated results in each row belong to the same category, which proves our Conditional Divergence Triangle can generate highly realistic images with given conditons another time.

What's more, we also want to check the ability of Condtional Divergence Triangle on generating colorful images under certain conditions. Thus, we modify our framework to learn from the famous Cifar-10 objects [20] datasets. Cifar-10 has 60,000 training images of 32×32 pixels with RGB three channels, and those images belong to 10 classes. In order to adopt our framework on the colorful images, we modify both the shapes of our input and output from $32 \times 32 \times 1$ to $32 \times 32 \times 3$ and keep other structures as the same. We show the original Cifar-10 training examples and synthesized images in each category generated by Conditional Divergent Triangle at below:

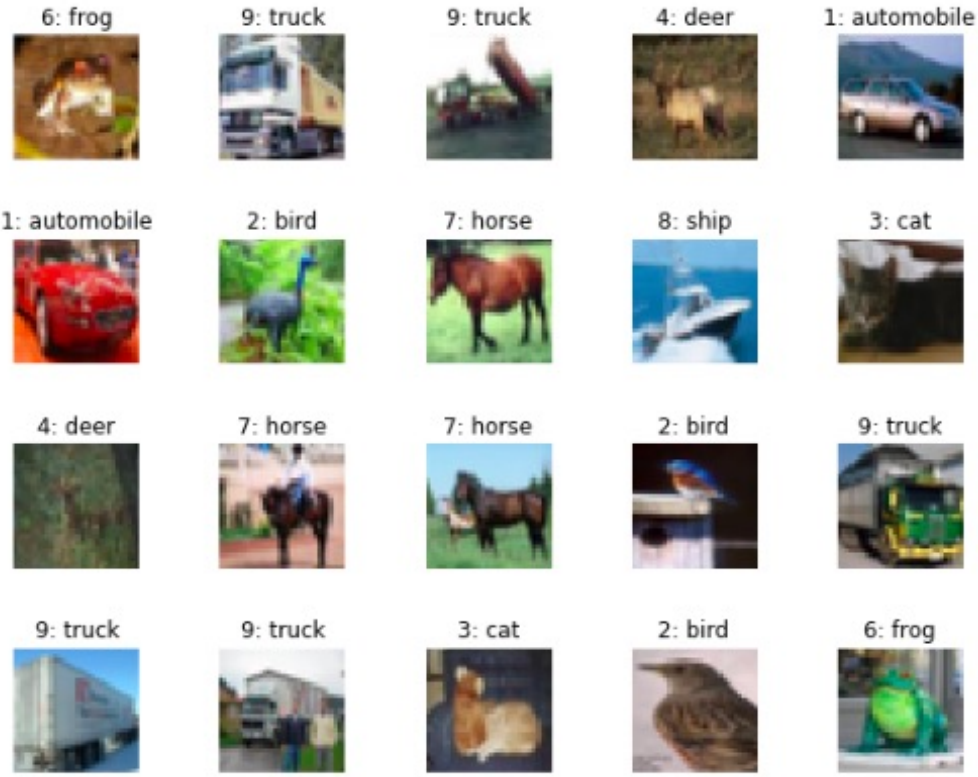

Figure 4.7: Training examples from Cifar10

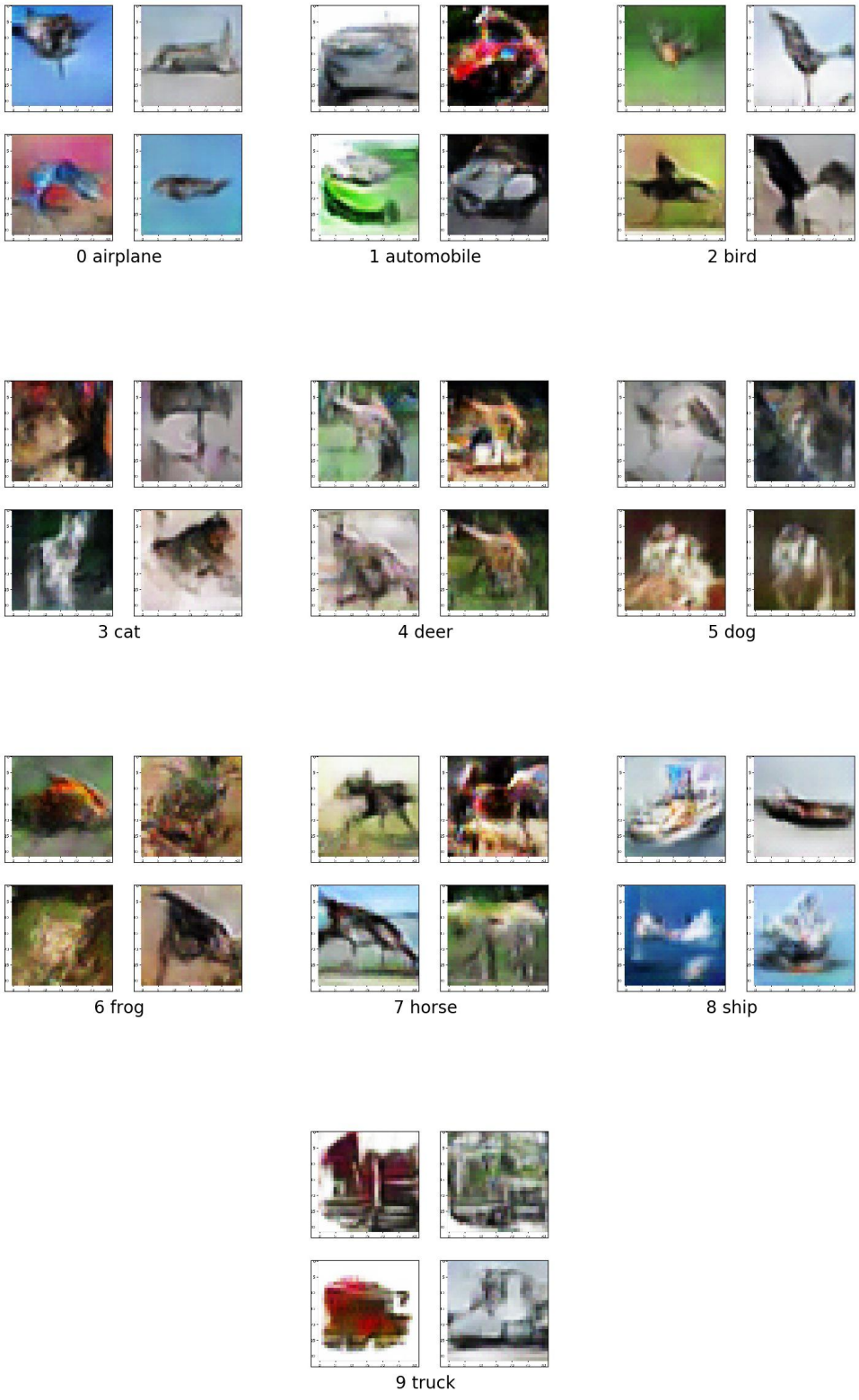

Figure 4.8: Synthesized conditional results from Conditional Divergent Triangle

The above results verify that the Conditional Divergent Triangle is also eligible to synthesize colorful images in fine-grained categories under certain conditions.

4.2 Obtaining meaningful lower-dimensional representation

Besides, we also implement experiments to show that our latent vector z is meaningful. In Figure 4.9, each column shares the same z value, and the value starts from -1 to 1 with step size 0.2 (skip z=0 for symmetry); the images in each row have the same condition y. From the result below, the digits seem thin and inclined when z is small; the digits seem fat and upright when z is relatively large, which means the latent vector z is meaningful. As we discover the styles of digits in each column are pretty similar, it indicates that z truely capture the visualized features of the images. Hence, we may use well-trained $p_\phi(z|x)$ in Conditional Divergence Triangle to infer those meaningful latent vector $\tilde{z}$ from given images, and some useful information about the images might be discovered by looking directly at those meaningful lower-dimensional representations.

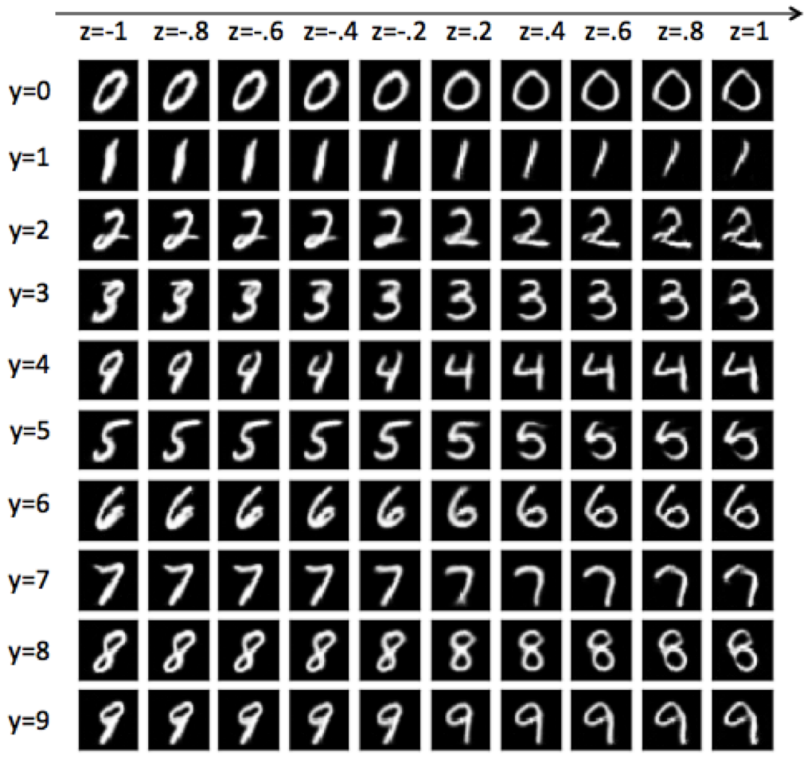

Figure 4.9: Conditional Linear interpolation in z for MNIST

We also perform similiar experiments on Fashion-MNIST as well as Cifar 10 datasets at below. The visualized features in images are gradually changed as z increases and each column maintains similar style, which further consolidate our conclusion above.

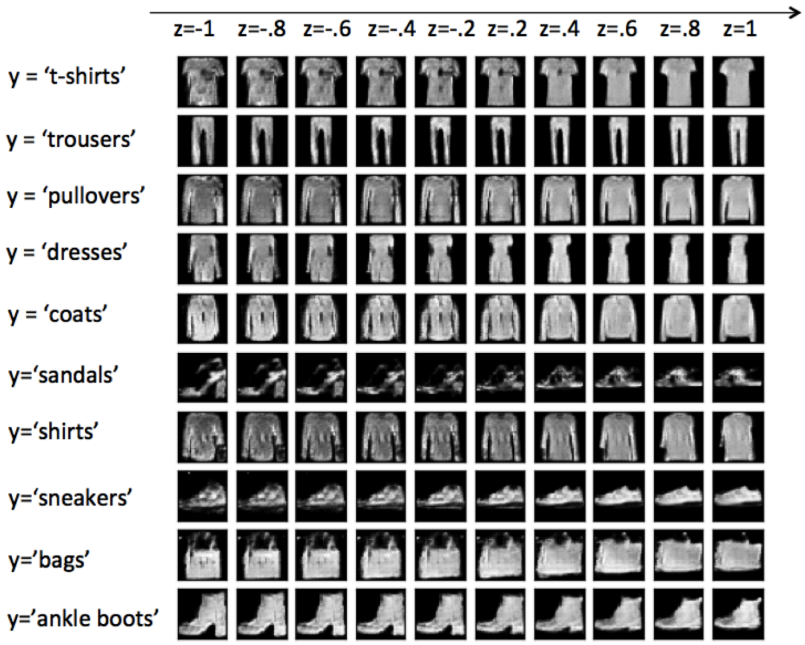

Figure 4.10: Linear interpolation in z for Fashion-MNIST

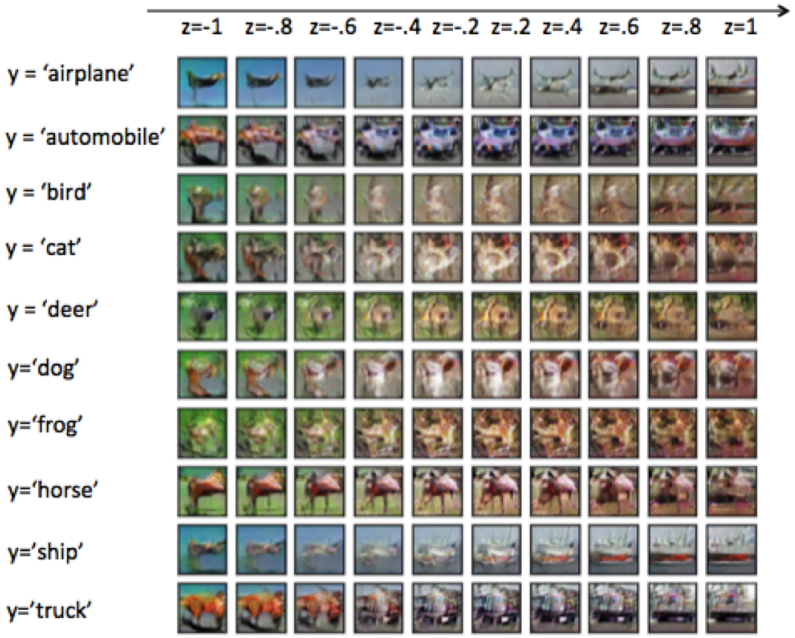

Figure 4.11: Linear interpolation in z for Cifar-10

4.3 Label classification on unobserved data

Finally, we can also make use of the well-trained $p_\phi(y|x)$, which is a part of the inference model, as a classifier to predict the labels for unobserved data. We implement the experiment on MNIST dataset, and the following table shows the accuracy comparsions between Conditional Divergence Triangle and other learning algorithms.

Accuracy on MNIST [21]	
Algorithm Name	Classification Accuracy
Logistic Regression	24.37%
Decision Tree (gini)	51.24%
KNN (10 neighbours)	76.40%
SVM (polynomial kernel)	87.11%
Plain DNN	90.72%
Conditional Divergence Triangle	**92.39%**
Conditional Divergence Triangle (double filters)	**93.52%**
ResNet-101	99.71%

Table 4.1: Comparisons of classification accuracy between different algorithms

From table 4.1, the classification accuracy of Conditional Divergence Triangle beats all other listed learning algorithms above except ResNet-101 [22]. I think there may exist two main reasons for the gap between our method and ResNet-101: (1) For the inference model of Conditional Divergence Triangle, we only use a simple four-layer convolutional nets structure to conduct the label classification; however, the ResNet-101 consists 100 convolutional layers and 1 fully connected layer, which is much more complex than the architecture of our approach. We try to double the number of filters in each layer for the inference model in our method, and find out the accuracy increase from 92.39% to 93.52%; hence, I believe if we have enough computational resources to increase the complexity of our inference model, our accuracy will also become higher. (2) Our inference model $p_\phi(y|x)$ is trained jointly, where the loss caused by both label y and latent vector

z; meanwhile, ResNet-101 only focuses on minimizing the Cross-entropy loss for label y, which has advantages on label classification.

The Conditional Divergence Triangle model is not just designed for classification, and our goal is to compelete multiple tasks within in one unified probabilistic framework through end-to-end training. In this respect, we believe the accuracy of our approach on label classification is acceptable.

CHAPTER 5

Conclusion and Future Work

In this study, we present the so-called Conditional Divergence Triangle model, which is an unified probabilistic framework for jointly learning generator, energy-based and inference models with the label information. Comparing to the original Divergence Triangle, the biggest improvements of our model are controlling the categories of synthesized images and expanding the ability of the framework to implement classification tasks on unobserved data.

Our extensive experiments above not only prove that Conditional Divergent Triangle model is able to synthesize high quality images under specific conditions, but also demonstrate that this framework can infer meaningful latent vectors as well as achieve relative high accuracy on the classification tasks.

In future work, we seek to extend the Conditional Divergent Triangle model from supervised learning to semi-supervised learning. Our initial idea is to use the limited paired data and augmentation techniques [23, 24, 25] to train a reasonable inference model as our classifer. After that, we use this classifer to predict labels for those unlabeled images and use them as new paired data to continue training the framework. In this way, we hope the Conditional Divergent Triangle model could still have good performance under limited training data.

Appendix A

Network Structures

Generator Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: z	1x1x100			input z
Input: y	1x1x10			input y
4x4 convT(256), ReLU	4x4x256	1	Yes	1st layer for z
4x4 convT(256), ReLU	4x4x256	1	Yes	1st layer for y
Concatenate	4x4x512			cat above 2 layers
4x4 convT(256), ReLU	8x8x256	2	Yes	2nd layer
4x4 convT(128), ReLU	16x16x128	2	Yes	3rd layer
4x4 convT(1), tanh	32x32x1	2	No	output

Table A.1: Network structure of Generator Model for MNIST & Fashion-MNIST

Energy-based Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: x	32x32x1			input x
Input: y	32x32x10			input y
4x4 conv(64), LReLU	16x16x64	2	Yes	1st layer for z
4x4 conv(64), LReLU	16x16x64	2	Yes	1st layer for y
Concatenate	16x16x128			cat above 2 layers
4x4 conv(256), LReLU	8x8x256	2	Yes	2nd layer
4x4 conv(512), LReLU	4x4x512	2	Yes	3rd layer
4x4 conv(1), Squeeze	Scalar	1	No	output

Table A.2: Network structure of Energy-based Model for MNIST & Fashion-MNIST

Inference Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: x	32x32x1			input x
4x4 conv(64), LReLU	16x16x64	2	Yes	1st layer
4x4 conv(128), LReLU	8x8x128	2	Yes	2nd layer
4x4 conv(256), LReLU	4x4x256	2	Yes	3rd layer
4x4 conv(100)	1x1x100	1	No	output: μ for $\tilde{z}$
4x4 conv(100)	1x1x100	1	No	output: σ for $\tilde{z}$
reshpae fc(10)	10		No	output:predicted $\tilde{y}$

Table A.3: Network structure of Inference Model for MNIST & Fashion-MNIST

Generator Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: z	1x1x100			input z
Input: y	1x1x10			input y
4x4 convT(256), ReLU	4x4x256	1	Yes	1st layer for z
4x4 convT(256), ReLU	4x4x256	1	Yes	1st layer for y
Concatenate	4x4x512			cat above 2 layers
4x4 convT(256), ReLU	8x8x256	2	Yes	2nd layer
4x4 convT(128), ReLU	16x16x128	2	Yes	3rd layer
4x4 convT(3), tanh	32x32x3	2	No	output

Table A.4: Network structure of Generator Model for Cifar 10

Energy-based Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: x	32x32x3			input x
Input: y	32x32x10			input y
4x4 conv(64), LReLU	16x16x64	2	Yes	1st layer for z
4x4 conv(64), LReLU	16x16x64	2	Yes	1st layer for y
Concatenate	16x16x128			cat above 2 layers
4x4 conv(256), LReLU	8x8x256	2	Yes	2nd layer
4x4 conv(512), LReLU	4x4x512	2	Yes	3rd layer
4x4 conv(1), Squeeze	Scalar	1	No	output

Table A.5: Network structure of Energy-based Model for Cifar 10

Inference Model				
Layers	In/Out Shape	Stride	BN	Layer name
Input: x	32x32x3			input x
4x4 conv(128), LReLU	16x16x128	2	Yes	1st layer
4x4 conv(256), LReLU	8x8x256	2	Yes	2nd layer
4x4 conv(512), LReLU	4x4x512	2	Yes	3rd layer
4x4 conv(100)	1x1x100	1	No	output: μ for $\tilde{z}$
4x4 conv(100)	1x1x100	1	No	output: σ for $\tilde{z}$
reshpae fc(10)	10		No	output:predicted $\tilde{y}$

Table A.6: Network structure of Inference Model for Cifar 10

REFERENCES

[1] T. Han, E. Nijkamp, X. Fang, M. Hill, S.C. Zhu and Y, N, Wu, "Divergence Triangle for Joint Training of Generator Model, Energy-based Model, and Inference Model", *arXiv preprint* arXiv:1812.10907, 2018.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", *arXiv preprint* arXiv:1312.6114, 2013.

[3] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning*, 2014, pp. 1278– 1286.

[4] A.MnihandK.Gregor, "Neural variational inferenceand learning in belief networks," in *International Conference on Machine Learning*, 2014, pp. 1791–1799.

[5] I.Goodfellow,J.Pouget-Abadie,M.Mirza,B.Xu,D.Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint* arXiv:1511.06434, 2015.

[7] Y. Lu, S.C. Zhu, and Y. N. Wu, "Learning FRAME models using CNN filters," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[8] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "A theory of generative convnet," in *International Conference on Machine Learning*, 2016, pp. 2635–2644.

[9] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng, "Learning deep energy models," in *International Conference on Machine Learning*, 2011, pp. 1105–1112.

[10] J. Dai, Y. Lu, and Y.-N. Wu, "Generative modeling of convolutional neural networks," *arXiv preprint* arXiv:1412.6296, 2014.

[11] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. "Cooperative training of descriptor and generator networks," *arXiv preprint* arXiv:1609.09408, 2016.

[12] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. "Cooperative learning of energy-based model and latent variable model via mcmc teaching," In *AAAI*, 2018.

[13] R. M. Neal, "Mcmc using hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, vol. 2, 2011.

[14] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint* arXiv:1609.03126, 2016.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint* arXiv:1412.6980, 2014. 1

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint* arXiv:1502.03167, 2015.

[17] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[18] Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets.*CoRR*, abs/1411.1784, 2014.

[19] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint* arXiv:1708.07747, 2017.

[20] N. Krizhevsky, H. Vinod, C. Geoffrey, M. Papadakis, and A. Ventresque, "The cifar-10 dataset," http://www.cs.toronto.edu/kriz/cifar.html, 2014.

[21] X. Chen, Z. Ye, Y. Zhang (2018). MNIST-baselines, Availabel at: https://github.com/cxy1997/MNIST-baselines [Last accessed 09 Feb 2019].

[22] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. *In: CVPR*, (2016).

[23] E. Jannik Bjerrum. SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules. *ArXiv e-prints*, Mar. 2017.

[24] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016.

[25] Y. Xu, R. Jia, L. Mou, G. Li, Y. Chen, Y. Lu, and Z. Jin. Improved relation classification by deep recurrent neural networks with data augmentation. *CoRR*, abs/1601.03651, 2016.