Equivariance in GAN Critics

A Thesis
submitted to the faculty of the
University of Minnesota
by
Yash Upadhyay

in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of
Computer Science

Adviser: Dr. Paul Schrater
May 2019

I dedicated this to Mie and Pa.

Acknowledgments

First of all, I would like to thank my advisor, Prof. Paul Schrater. Without his continuous support of my research, guidance, and advice, this thesis would not be possible. His unwavering belief and encouragement even when the things weren't the brightest helped me push through the hurdles.

I would also like to thank the members of my committee, Prof. Daniel Kersten and Prof. Hyoo Su Park for their help and support.

Working through the rough times wouldn't have been possible without the help of my friends - Ayush, Debarati, Manish and Vaybhav. Their constant support and presence not only helped me survive the graduate student's life, but also made it fun.

None of this would have been possible without the support of my nearest and dearest. My parents - Manisha and Ghanshyam Upadhyay have been the two strongest pillars in my life. It is only through their support and encouragement that I was able to deal through the worst and shine through the best of my days. Their encouragement and freedom to let me choose what I want to do led me to work on what I really enjoy the most.

Lastly, without my best friend - Vaishnavi, the struggles through the past decade would have been way more difficult to overcome.

Thesis advisor: Paul Schrater Yash Upadhyay

Equivariance in GAN Critics

Abstract

Equivariance allows learning a representation that disentangles an entity or a feature from it's meta-properties. Spatially-equivariant representations lead to more detailed representations that can capture greater information from the image space in comparison to spatially-invariant representations. Convolutional Neural Networks, the current work-horses for image based analysis are built with baked-in spatial-invariance which helps in tasks like object detection. However, tasks like image synthesis that require learning an accurate manifold in order to generate visually accurate and diverse images would suffer due to the incorporated invariance. Equivariant architectures like Capsule Networks prove to be better critics for Generative Adversarial Networks as they learn disentangled representations of the meta-properties of the entities they represent. This helps the GANs to learn the data manifold much faster and therefore, synthesize visually accurate images in significantly lesser number of training samples and training epochs in comparison to GAN variants that use CNNs. Apart from proposing architectures that incorporate Capsule Networks into GANs, the thesis also assesses the effects of varying amounts of invariance over the quality and diversity of the images generated.

Contents

Listing of figures

List of Tables

"What I cannot create, I do not understand"

Richard Feynman

1 Introduction

1.1 Motivation

With recent advances in deep learning, machine learning algorithms have evolved to such an extent that they can compete and even defeat humans in some tasks, such as image classification on ImageNet [6] and playing Go [36]. However, it is hard to conclude that those algorithms have true "intelligence", since knowing how to do something does not necessarily mean understanding something, and it is critical for a truly intelligent agent to understand its tasks. To put this quote at the header of the chapter, in the case of machine learning - for machines to understand their input data, they need to learn to create the data. The most promising approach is to use generative models that learn to discover the essence of data and find a best distribution to represent it.

The task of image synthesis involves generating photo-realistic images from a given description from a given dataset. This description could be just random variables, some text or selective features inherent to the manifold of images from which the new image is being synthesized from. This task requires the use of generative models that can learn rich representations of the data and use the learned mapping in order to generate new images. There are many variants of algorithms like boltzmann machines [14], variational auto-encoders [19] and generative adversarial networks (GAN) [9] that have shown promising results in this domain. However, GANs have found significant popularity because of the ability to be trained completely via back-propagation, as well as impressive results despite the challenges in training them.

While there are many variants of the basic GAN architecture, the core ideas have crystallized and recent innovations revolve around specializing GANs to aid in very specific applications. Deep Convolutional Neural Networks [24] have been the work-horses for the task of image synthesis for a while. DCGANS, [31] use CNNs as discriminators and Deconvolutional Neural Networks [44] as generators. CNNs capture localized features through the layers over varying granularity and use max-pooling to incorporate positional invariance of these features captured in an image. Despite providing the required variance in positions of the local features for learning global representations (in a limited manner), pooling leads to a form of lossy compression of the image features. Also, [32] shows that the use of pooling helps CNNs only in the earlier epochs of the training and the CNNs that have been trained over greater number of epochs learn smoother filters that achieve the same performance as the CNNs with pooling layers that have been trained over similar number of epochs. Thus, rendering the employment of pooling layers unnecessary. CNN filters run convolutions over localized areas and through the successive layers try to greater translational invariance.

Invariance is a special case of equivariance, which is a mathematical property

of a representation which encodes information of transformations applied to the entity in the representation. This is achieved by disentangling the information regarding the original entity and the transformation applied to it. Invariance simply does not disentangle this information and doesn't encode any meta-features about the entity, essentially losing out information that might be useful for further analysis. Extending this idea, CNNs progressively learn spatially-invariant representation of the input images and end up learning limited spatial-relations between the features present in the image [15, 33].

In a GAN, the discriminator is the module that drives the loss functions for both, itself and the generator. Especially, the critic in WGAN [2, 34], judges the quality of the images generated critiquing them with a score that corresponds to the distance of the synthesized images to the real image manifold. Thus, it is essential in the critic reaches optimality so that it can provide better gradients for the generator. In a scenario where the critic itself is occluded from the true manifold due to lossy invariant properties inherent in its architecture, the driving gradients will not be optimal for the generator. If the critic can view the real manifold in a more informative manner, then it can make better judgments. Disentangling the meta-information regarding an entity would aid in such a scenario as such a representation would be better for showing the presence/absence of an entity and the information it carries with itself.

Capsule Networks[33] incorporate spatially-equivariant representation which can aid in mitigating the problem of learning equivariant and disentangled representations of the data. Capsules are a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity, such as an object, and the length of this activity vector represents the probability of the existence of the entity that the vector represents. Capsule Networks incorporate positional-equivariance as opposed to positional-invariance between the features of an image by using Dynamic Routing between Capsules. The attention-like Routing algorithm between layers allows Capsules from a given layer to learn the

Epochs	MNIST		Fashion-MNIST	
	CNN	CapsNet	CNN	CapsNet
1	91.09	98.51	48.72	84.25
2	93.53	92.22	73.99	86.53
3	95.04	99.41	75.36	87.91
4	96.30	99.58	78.64	88.97
5	97.17	99.63	81.02	90.05

Table 1.1.1: Capsule Network v/s Convolutional Neural Network test accuracy comparison(%)

contributions from the relevant Capsules from the previous layer. This leads to Capsule Networks learning a richer representation of the features present in the images along with the relations between them on a more global scale.

A simple experiment on the speed of learning of Capsule Networks in comparison to CNNs of similar number of trainable parameters and learning rate was an initial motivation for this thesis. Referring Table 1.1.1, we can see a significant improvement in performance in terms of accuracy, as well as the learning speed. This does show to a certain degree that the representations that the Capsules are learning are much better for classification.

Therefore, we explore Capsule Network, a spatially-equivariant network as a critic in a WGAN. A more powerful critic that can model the manifold better and reach optimality faster can provide better gradients for the generator to learn. Thus, helping the generator synthesize images of greater visual fidelity while seeing significantly lesser number of samples in comparison to WGANs that use a CNN critic.

Following are the contributions of this paper:

1. GAN architecture with an equivariant critic that can learn disentangled

representations

2. A Split-Auxiliary critic architecture for using Capsule Networks for conditional image synthesis

3. Quantitative analysis of the images synthesized by our proposed architectures and comparison with CNN based architectures

4. Analysis of how equivariance and invariance in the critic of the GAN can affect the synthesis of new images

1.2 Key ideas

1.2.1 Equivariance is better than Invariance

Spatial-equivariance helps the networks learn better representations as they are able to disentangle the information about the entities they encode and their meta-properties. This is in comparison to spatially-invariant representations which don't store relationships and meta-features about features/entities they represent.

Capsule Networks are based on the principal of positional equivariance whereas, CNNs are based on the principal of positional invariance of the features. Convolutional Neural Networks lose the spatial relationships between features through successive layers especially if using max-pooling as it brings greater positional-invariance and proves as a lossy form of feature compression. Capsule networks are able to learn the features and the relations between them better than Convolutional Neural Networks because the Primary Capsule layer looks at all the the features of the input image and the routing process determines the set of global features that contribute to a capsule in the Secondary Capsule layer. Also, the fact that each Capsule encodes the properties of the entity/feature it represents, enables Capsule Networks to model the distribution in greater granularity, making them more robust to small affine transformations [33].

1.2.2 Better the critic, better the generator

The concept of using a Wasserstein-1 distance as a loss metric [34] requires replacing a discriminator, that outputs the probability of an image being real or fake with a critic that assigns a high score to a real image. The critic is trained to maximize the Wasserstein-1 distance between the scores assigned to the real and fake images whereas the generator is trained to increase the score that the critic churns out for the images it synthesized. The true gradients being more meaningful, guide the generator to synthesize more realistic images consistently. The better the gradients, the better the generator learns but for the gradients to be of healthy, the critic must reach optimality. Therefore, the faster the critic reaches the optimality, the faster the generator learns. The quality of the gradients also depends on the capability of the critic to learn the image manifold since, a poorly learned distribution wont give accurate scores to the images. Therefore, the better the critic learns the manifold, the better the generator gets. Apart from better visual fidelity, it promotes diversity in the images that are generated as equivariance leads to much better mapping of images over the manifold.

Referring to the previous section, we see that the principal of feature equivariance helps Capsule Networks perform better than the CNNs modelled after feature invariance. The improvement in performance comes in the form of learning the distribution faster and better. Therefore, replacing the CNN critic with a Capsule Network should present us with improvements in the the quality and reduction in the amount of data required to synthesize images with visual fidelity.

1.2.3 Critiquing and classification are supplementary

Critiquing requires the critic to learn the distribution of the real images and generate high scores to the images that belong to them. Whereas, classification requires that the classifier learn the key features occurring in the dataset of images and use them to classify the the images according to the features present in them. As described in [29], the using class information for training a GAN aids the

GAN to model the structure better. Being able to learn features that play a key role in discriminative tasks can help the critic learn more about the distribution. Capsule Networks have achieved state-of-the-art performance for classification. This points into a direction where Capsule Networks can be used to extract discriminative features that can help in critiquing. Instead of of using an ensemble of a generator and a classifier, we use a split-auxiliary architecture, where the network remains same for the critique as well as the classifier up till the last layer and the penultimate layer feeds its features to two different layers with different purposes. This helps the Primary Capsule layer learn the features necessary for discrimination while also building structure in features to help the critic score the samples.

2

Background

2.1 Discriminative and Generative Models

Majorly there are two types of machine learning models: Generative and Discriminative [28]. Discriminative models learn the conditional probability of the class of a given data point. Let X and Y be the observable and target variables respectively, then, it can be said that Discriminative model is a statistical model of the conditional distribution, $P(Y|X)$. In contrast to this, Generative models learn the joint probability distribution of X and Y, given by $P(X, Y)$. For a given instance x, Generative models classify it to a target class that maximizes the likelihood of $P(Y, X = x)$. Alternatively, according to [1], a generative model is a model of the conditional probability of the observable X, given a target y, symbolically, $P(X|Y = y)$.

Discriminative models, as opposed to generative models, do not allow one to generate samples from the joint distribution of observed and target variables. However, for tasks such as classification and regression that do not require the joint distribution, discriminative models can yield superior performance (in part because they have fewer variables to compute) [22, 28, 38]. On the other hand, generative models are typically more flexible than discriminative models in expressing dependencies in complex learning tasks. In addition, most discriminative models are inherently supervised and cannot easily support unsupervised learning. Application-specific details ultimately dictate the suitability of selecting a discriminative versus generative model.

Some examples of discriminative models are Linear Regression, Logistic Regression, support vector machines whereas, Gaussian Mixture Models, Naive Bayes Classifier and Autoencoders are Generative Models.

2.2 Equivariance and Invariance

Two key mathematical properties of image representations are - equivariance and invariance. Equivariance looks at how the representation changes upon transformations of the input image. Let the function φ transform an image $x \in \chi$ into a vector $\varphi(x) \in \mathbb{R}^d$. A representation φ is equivariant with a transformation g of the input image if the transformation can be transferred to the representation output. Formally, equivariance with g is obtained when there exists a map $M_g : \mathbb{R}^d \to \mathbb{R}^d$ such that:

$$x \in \chi : \ \varphi(g(x)) = M_g \varphi(x) \tag{2.1}$$

The structure of the mapping M_g should be simple, for example a linear function. Also, by requiring the same mapping M_g to work for any input image, intrinsic geometric properties of the representations are captured.

Invariance is a special case of equivariance obtained when M_g acts as the identity map. Invariance is often regarded as a key property of representations since one of the goals of computer vision is to establish invariant properties of images. For example, the presence of objects contained in an image is invariant to viewpoint changes.

The problem of designing invariant or equivariant representations has been a popular task in computer vision. Deep CNNs and related state-of-the-art architectures, are deemed to build an increasing amount of invariance layer after layer. This is even more explicit in the scattering transform [35]. There are other many architectures that try to learn equivariant representations [5, 20, 41]. Capsule Networks [15, 33] try to extract invariant local descriptors and add equivariant transformations over them.

2.3 CNNs

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. Convolutional networks were inspired by biological processes[7, 8, 16, 25] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer i.e. activation function, pooling layers, fully connected layers and normalization layers.[17]

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution

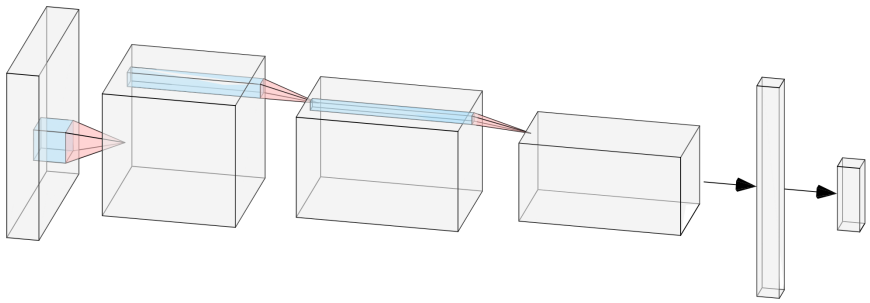

Figure 2.3.1: Typical architecture of a Convolution Neural Network which consists of Convolutional layers, Pooling layers and/or Fully Connected layers. Convolutional Kernels provide localized feature activation whereas Pooling layers reduces computation and also provides greater invariance properties to the network.

(although cross-correlation is a related operation). This only has significance for the indices in the matrix, and thus which weights are placed at which index. Each convolutional neuron processes data only for its receptive field, thereby reducing the number of parameters in a layer significantly in comparison to that required to process an image by a fully connected layer.

Apart from convolution layers, another key operation is that of Pooling. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters whereas, Global pooling acts on all the neurons of the convolutional layer. [21] Pooling operations may either consist of Max, Min or Average operations. [27] These operations are performed over the values looked by pooling receptive fields in case of local pooling, which also stride in a manner similar to that of the kernels in a Convolutional layer. Pooling aid in significant reduction of computation as it is cheaper than convolution operations and also reduces the size of the feature maps. Apart from computational benefits, it also leads to increased invariance, which may or may not be a good thing depending

upon the task.

Despite popular use of Pooling layers for many tasks, there are literature that state that pooling layers aren't necessary to achieve similar effects/performances. [32, 46]. Increasing the stride of the kernels in the Convolution layer is another option that one may opt in place of pooling.

Most deep learning architectures that work with images or videos utilize a variant of the Convolutional Neural Network, utilizing a combination of the Convolutional, Pooling and Dense layers. VGGNet [37] promoted the usage of smaller kernel sizes, which would lead to deeper networks. These worked better in comparison to the traditional networks for image recognition as the deeper networks added greater non-linearity which helped learn more abstract representations. The concept of repeating modules became popular after the VGGNet. However, despite its impressive performance at the time, VGGNet was computationally expensive due to the large number filters and layers. GoogLeNet (Inception-Net) [39] introduced the concept of 1×1 convolutions to reduce the computations along with the idea to stack feature maps generated by kernels of various sizes in order to extract features of different scale from a layer. Despite seeing an increase in performance in the task of image recognition, the problem of vanishing gradients became a prominent impediment in further increas of performance due to increasing depth of networks. This was overcome with the help of Residual blocks introduced in ResNets [12]. Residual blocks, as inspired by VGGNets used a series of repeating blocks to deepen the network, however, with the introduction of skip connections to map identity functions in order to help a healthier flow of gradients through deeper networks.

2.4 Capsule Networks

Sabour et al. [33] developed Capsule Networks as parse trees carved out from a single multi-layer neural network where each layer is divided into many small

groups of neurons called as Capsules, corresponding to each node in the parse tree. Each Capsule vector represents the meta-properties of the feature/entity the Capsule represents and the overall length of the Capsule represents the probability of the presence of the entity the Capsule represents. Each active Capsule chooses its parents from the layer above it using an iterative attention-like routing process. This dynamic routing process replaces the max-pooling step from CNNs allowing for better feature globalization and smarter feature compression.

Since the length of each capsule represents the probability of the presence of the entity it represents, a non-linear "squashing" function is used to shrink the length of the vector s_j between 0 and 1, which is denoted by the vector v_j, the output of the capsule j.

During the routing process from capsule i of a given layer to capsule j of the next layer, the output of capsule i, u_i is first multiplied by the matrix W_{ij} to give u_{ji}. s_j is then calculated as the weighted sum over all u_{ji} coming from the previous layer to capsule j, weighted over the coupling coefficient, c_{ij}. The coupling coefficient c_{ij} is calculated as the softmax over all b_{ij}, which is the summation of the agreements, a_{ij} between the individual input capsules and output of capsule j over all the iterations. The agreement, a_{ij}, is calculated as the dot product of the output, v_j and the incoming vector, u_{ji}.

Since the length of each capsule represents the probability of the presence of the entity it represents, a non-linear "squashing" function is used to shrink the length of this vector between 0 and 1. The output of this non-linearity from capsule j, v_j, is given as follows,

$$v_j = \frac{||s||^2}{1 + ||s||^2} \frac{s_j}{||s_j||} \tag{2.2}$$

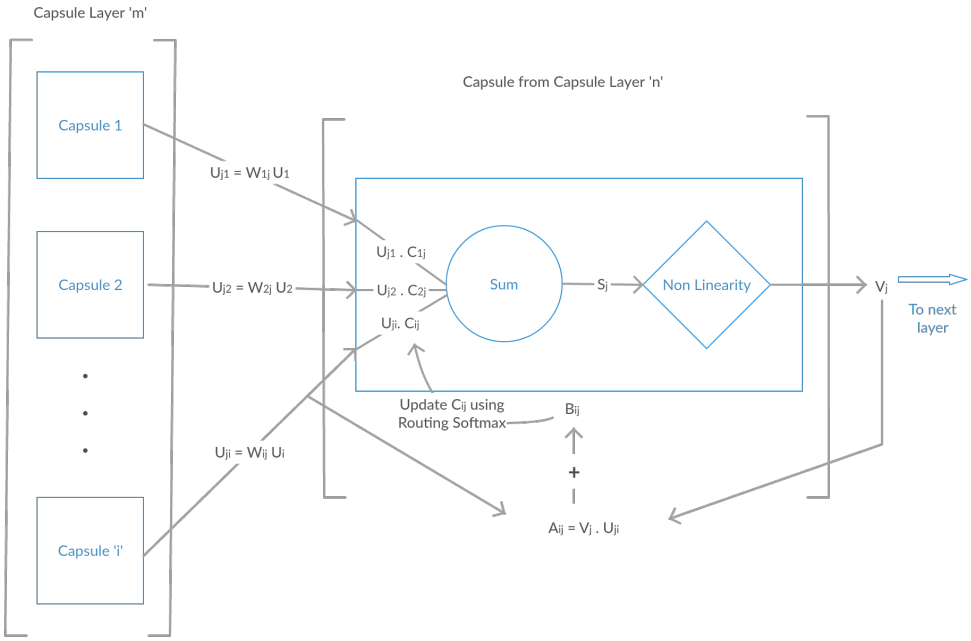

Figure 2.4.1: Dynamic Routing between Capsule Network Layers: The figure shows the attention like routing mechanism between capsule layers that allow a capsule to choose its parents via an iterative deterministic process.

where s_j is the total input.

During the routing process from capsule i of a given layer to capsule j of the next layer, the output of capsule i, u_i is first multiplied by the matrix W_{ij} to give u_{ji}. s_j is then calculated as the weighted sum over all u_{ji} coming from the previous layer to capsule j, weighted over the coupling coefficient, c_{ij}.

$$u_{ji} = W_{ij} u_i \tag{2.3}$$

$$s_j = \sum_i c_{ij} u_{ji} \tag{2.4}$$

The coupling coefficient c_{ij} is calculated as the softmax over all b_{ij}, which is the

summation of the agreements, a_{ij} between the individual input capsules and output of capsule j over all the iterations.

$$b_{ij} = \sum_{t} a_{ij_t} \tag{2.5}$$

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{2.6}$$

The Capsule Network introduced by Sabour et al. [33] used a marginal loss over the capsules in the final layer for optimization along with a reconstruction loss which helps in regularization. However, the architectures proposed in this paper do not incorporate the reconstruction loss. The marginal loss, L_k is defined as following,

$$L_k = T_k(max(0, m^+ - ||v_k||^2))^2 + \lambda(1 - T_k)(max(0, ||v||^2 - m^-))^2 \tag{2.7}$$

where $m^+ = 0.9$, $m^- = 0.1$, $T_k = 1$ iff the entity of class k is present else, $T_k = 0$. $\lambda = 0.5$ is used as a a down-weighting factor to prevent shrinking of activity vectors in the early stages of training.

By incorporating the process of Dynamic Routing over Capsule vectors derived from spatial features, Capsule Networks exploit the fact that while viewpoint changes have nonlinear effects at the pixel level, they have linear effects at the part/object level. Thus, incorporating equivariant properties in relation to small spatial transformations. This allows Capsule Networks to encode information relevant to an object in a much more efficient manner, which would get lost in a a network with deeper levels of invariance, especially with pooling. The disentanglement of an object's properties from its presence is observed in the meta-information stored by the Capsule vectors, whose length corresponds to the existence of the entity. The autoencoder-like reconstruction experiments in [33] show that the information stored by the vectors corresponds to the actual manifold of the objects, thus, also bringing in interpretability. This helps Capsule Networks learn the representations in a much faster and efficient manner.

Experiments in Section 4 bolster this and the architectures proposed in this thesis also benefit from the same.

An incremental effort by [15] on the original Capsule network introduced the concept of Expectation Maximization Routing (EM-Routing) in place of Dynamic Routing between Capsules. EM-Routing treats lower Capsules as data points that belong to the distribution represented by the higher Capsules, essentially clustering groups of lower level Capsules to indicate activation of higher Capsules. Also, the transformation of vector based Capsules into matrix-based Capsules helped in parameter reduction. However, several rounds of EM for Capsule clustering proves to be computationally very expensive and therefore, this thesis doesn't cover the applications of Matrix Capsules in the proposed architectures.

To summarize, the spatially equivariant properties inherent in Capsule Networks lend the following benefits over traditional Convolutional Neural Networks:

- Viewpoint invariance: the use of pose matrices allows capsule networks to recognize objects regardless of the perspective from which they are viewed.
- Better performance in comparison to networks of similar parameter size
- Better generalization to novel viewpoints
- Demonstrated adversarial robustness [15]

2.5 Generative Adversarial Networks

Generative Adversarial Networks [9] are finding popular applications as generative models in diverse scenarios. One of the biggest advantages of GANs is that they can be trained completely using back-propagation. GANs utilize two

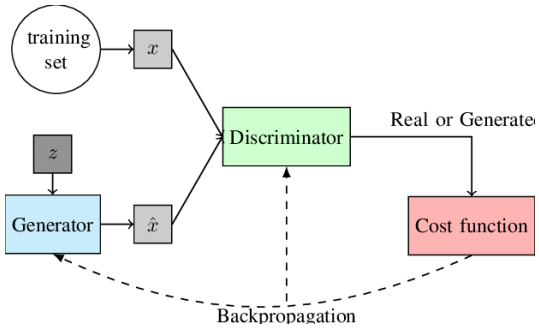

Figure 2.5.1: A general GAN architecture where the Discriminator looks at both - the real and fake images generated by the Generator. The loss function calculated by the Discriminator's assessment drives the training of the Discriminator itself as well as the Generator.

adversary multi-perceptron networks (generator and discriminator) that play a minimax game where the generator tries to learn the probability distribution p_g over a dataset x. Noise variables $p_z(z)$ serve as an input to the mapping function , the generator, $G(z, \theta_g)$ with parameters θ_g. $D(x, \theta_d)$ represents the discriminator with parameters θ_d and $D(x)$ represents the probability that the input x came from the dataset rather than p_g. Following equation describes the minimax game being played by the adversaries,

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[log(D(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{2.8}$$

[2] discussed how KL-divergence, as the proposed loss function in [9], can lead to uninformative gradients over low-dimensional manifolds where the intersection between the real and generated data can be very small. Therefore, they introduced Wasserstein GAN (WGAN) that used Earth-Mover's (Wasserstein-1) distance as a loss metric, which is continuous and with near-linear gradients provides a healthy convergence of the generator. WGANs used gradient clipping as a naive way to hold 1-Lipschitz continuity, which was later on improved by [10] with the use of gradient penalty. The use of Wasserstein Distance gave the freedom to use a powerful critic, as a critic that could reach optimality faster would be able to provide much healthier gradients to the

generator which would ultimately lead the the generator to learn much better. However, [10] demonstrated how the use of a gradient clipping used to enforce 1-Lipschitz continuity was a naive approach and therefore led to very strong regularization of the critic, which ultimately led to underfitting of the model. Therefore, they introduced Gradient Penalty, $\lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$ to the original Wasserstein critic loss which helps bypass the regularization via gradient clipping. The following shows the new loss function for the critic to minimize,

$$L = \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] + \lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{2.9}$$

where, $\mathbb{P}_{\tilde{x}}$ is defined by sampling uniformly across straight lines between pairs of points sampled from $\mathbb{P}_r$ and $\mathbb{P}_g$.

Apart from innovation on the front of loss functions, the newer variants bring innovation in the form of structure for specific tasks. For example, conditional GANs [26] use extra label information and result in better quality images and are able to control how generated images will look based on the input class information. A structural variant of this is Auxiliary-Conditional GAN [30] which uses a separate cross-entropy loss from a classifier in conjunction to the GAN loss for getting class-label structured images. Stack GANs [45] use textual input to generate a low resolution image based on the description and then a second super-resolution task conditioned on the low-resolution image and the input text. Disco GANs [18] learn relationships between different domains and then use these relationships for style-transfer from one domain to the other while preserving certain characteristics. However, an important point to note is that despite the multitude of variations in GAN architectures, tasks related to image synthesis use variants of CNN-based discriminators.

2.6 Generated Image Evaluation

Evaluating the images synthesized by generative model is a non-trivial task. One intuitive way to judge the images qualitatively is by the use of human annotators

but it is a highly subjective process and the results vary greatly, even the ones coming from single person. Therefore, Inception Score, a quantitative measure, was introduced by [34], which is given by,

$$IS = e^{\mathbb{E}_x KL(p(y|x)||p(y)} \tag{2.10}$$

Inception score is found to be correlated with the human judgment of image quality and therefore, is one of the most widely used metric for evaluating image generative systems.

However, as many have pointed out [3, 4, 13], Inception Score does not take diversity of the images within a class into consideration. The generator can get away with a high score even if it replicated one image per class. Therefore, [11] tried to introduce the modified-Inception Score, given by

$$mIS = e^{\mathbb{E}_{x_i}[\mathbb{E}_{x_j}[KL(p(y|x_i)||p(y|x_j))]]} \tag{2.11}$$

This rewards the high entropy of class-conditional probabilities of images within a class. But this doesn't necessarily mean all the images in the given will be of good quality. This score would fail in a scenario where the generator has synthesized diverse, yet perfect images within a class and for all the classes. To address problems arising out of evaluating generated images over void, [13] introduced the Frechet Inception Distance Score that uses the Wasserstein-2 distance between the activations for the real and generated images. It assumes that the activations from the two datasets follow a Gaussian distribution given by (μ_r, C_r) and (μ_f, C_f) and the lesser the distance between them, the better are the samples generated. It is given as follows,

$$||\mu_r - \mu_f||_2^2 + Tr(C_r + C_f - 2(C_r C_f)^{1/2} \tag{2.12}$$

However, there is no metric that disentangles the quality of images synthesized from the diversity of the generated images in a manner where diversity is achieved

in a semantic sense. Quantification of diversityin image space on a semantic scale is difficult as it will be inherently dependent on the actor's biases. [40] tries to convert the problem of diversity into that of distances as it calls images "further away" from each other to be more diverse than images "closer" to each other. However, the distance metric that can be appropriate for such a task is non-trivial.

3 Architectures

This thesis proposes the conditional and non-conditional architectures of Capsule GANs which incorporate equivariance as a part of the discriminator. The architectures allow for various forms of equivariances to be incorporated as a part of the architecture while allowing the network to remain relatively shallower in comparison to networks with similar properties and comparable performance. This section deals with the exact details for the proposed architectures.

3.1 Capsule GAN

This architecture uses Capsule Networks as a discriminator in place of a Convolutional Neural Network used in DCGANs. It can be seen in Fig. [3.1.1], our Capsule Network uses two Capsule layers: Primary and Secondary Capsule layers, in which, there is no routing between the Convolutional layer and the

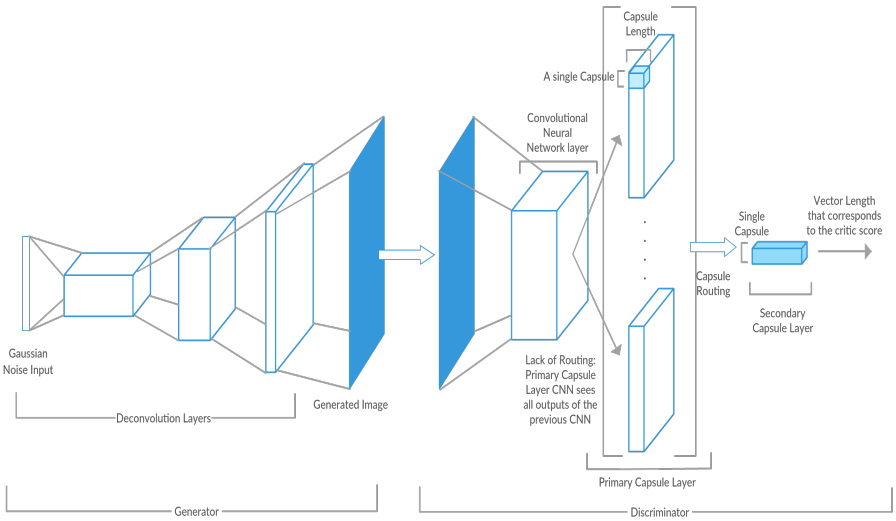

Figure 3.1.1: Capsule GAN Architecture

Primary Capsules. Routing exists only between the Primary and Secondary Capsules. The Secondary Capsule layer consists of only one Capsule and the "squashing" non-linearity used in the Primary Capsule layer is not applied here. The length of the activity vectors of this Capsule represents the score of the critic. We use the critic loss function described in Eqn. [3.1], where the critic D returns the length of the output capsule.

$$L_D = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2] \tag{3.1}$$

Whereas, the generator loss function is described as follows:

$$L_G = -\mathbb{E}_{x\sim P(z)} - D(G_\theta(x)) \tag{3.2}$$

where $P(z)$ represents the prior distribution serving as the input to the generator G_θ.

3.2 Conditional Split-Auxiliary Capsule GAN

For conditional generation, apart from receiving random variables generated from a Gaussian distribution, the generator also receives the class from which it must synthesize the image. The two vectors are then concatenated and utilized by the generator as a latent space representation of the image to be synthesized.

The discriminator uses a variation of the auxiliary-conditional architecture described by [29]. The discriminator consists of a similar architecture up to the Primary Capsule layer as described in Section 3.1. The Primary Capsules then serve as an input for two different Secondary Capsule layers: Primary Critic and Secondary Classifier. The Primary Critic scores the input images as being fake or real whereas, the Secondary Classifier classifies the image into the class label it belongs to. The Wasserstein Loss from the Primary Critic and the Marginal Loss from the Secondary Classifier (Eqn. [3.3]) are then coupled together.

$$L_k = T_k(max(0, m^+ - ||v_k||^2))^2 + \lambda(1 - T_k)(max(0, ||v||^2 - m^-))^2 \quad (3.3)$$

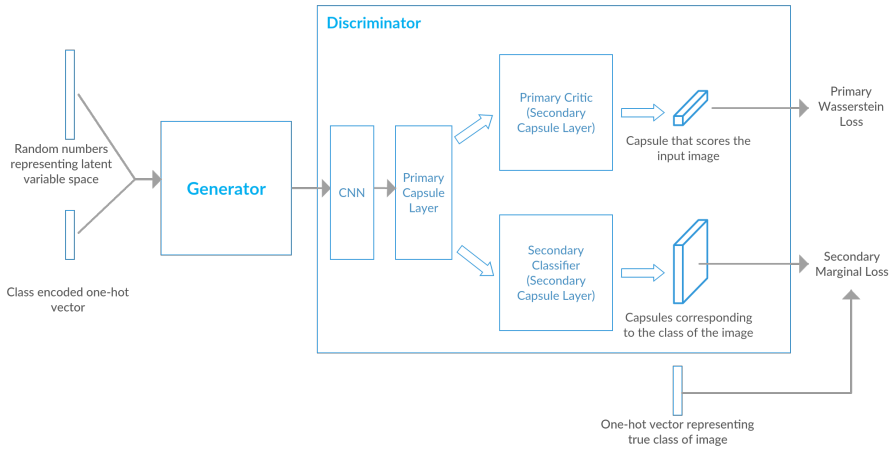

Figure 3.2.1: Architecture of Conditional Split-Auxiliary Capsule GAN

Let the Primary Critic Loss be L_P (Eqn. [3.1]), Secondary Marginal Loss be L_{S_c} (Eqn. [3.3]) and the Generator Loss from Eqn. [3.2] be denoted by L_{G_W}. The losses for the Discriminator(L_D) and the Generator(L_G) is given as follows:

$$L_D = L_P + L_{S_{x \in P(r)|y=c}}(x) + L_{S_{x \in P(z)|y=c}}(G_\theta(x)) \tag{3.4}$$

$$L_G = -L_{G_W} + L_{S_{x \in P(z)|y=c}}(G_\theta(x)) \tag{3.5}$$

where G_θ is the generator with parameters θ, $P(r)$ corresponds to the probability distribution of the dataset, $P(z)$ corresponds to the probability distribution of the prior to the generator, $k \in \{P(r), P(z)\}$, y is the class label of the image, and c corresponds to the intended class of the image. The Secondary Marginal Losses force the discriminator to the learn the representation of an image conditionally over a label. The split architecture allows for the Primary and Secondary Capsule Classifiers to borrow from the same set of extracted features in the Primary Capsules for 2 tasks - critiquing the validity of the image and classifying the class of the image. Apart from helping the Primary Capsules learn features for the class of an object, the split-architecture also helps reduce computational overheads due to a completely autonomous second classifier.

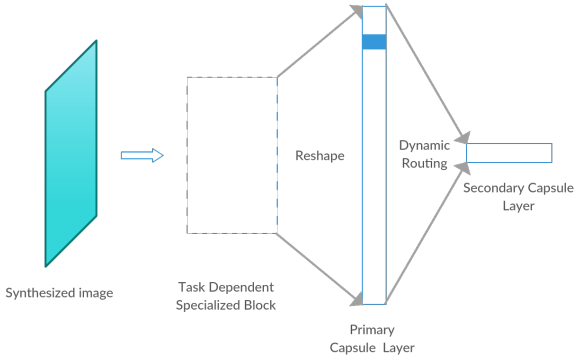

Figure 3.3.1: Generalized Capsule GAN for incorporating task specific equivariance or invariance

3.3 Capsule GAN Architecture for Task-Specific Equivariance

The structure of Capsule Network allows integration of equivariance or invariance as required by the task. The initial convolution layer can be replaced by a layer that incorporates the required properties of task dependent equivariance (such as rotational equivariance) into the discriminator, without significant change in the overall structure. Referencing Fig. 3.3.1, the feature extraction step of the Capsule Network can be altered in order to extract features that correspond to the required properties. It is in this layer that equivariance and invariance in the features extracted can be adjusted.

Addition of pooling layers such as max-pool or average-pool after a convolution layer on the input image allows for greater invariance between the extracted features. This is used to evaluate the direct impact of the addiction of invariance to the network.

This block can also be used to add equivariance for specific tasks. In order to

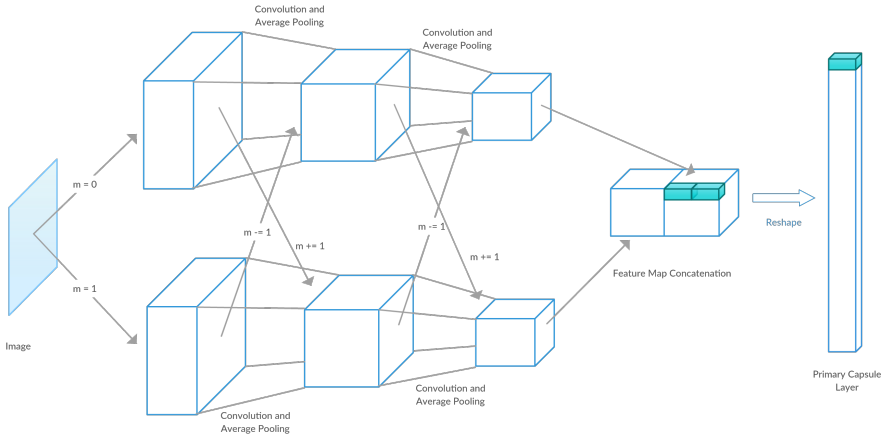

Figure 3.3.2: Harmonic Capsule GAN Architecture

incorporate rotational equivariance into the discriminator, harmonic convolutional layers can be added. Referencing Fig 3.3.2, the synthesized image is the input to the harmonic convolutional layer. This is succeeded by further 2 harmonic convolution layers with 2 streams of rotational frequency of $m\varepsilon\{0, 1\}$. The output from these streams is then concatenated and reshaped into the Primary Capsules. Note that the convolution layer in the Primary Capsule layer will have to be omitted as it will distort the entities extracted by the harmonic layers.

4

Experiments

4.1 Capsule GAN Evaluation

As a part of this section, we compare the results the images generated by the traditional Convolutional Neural Network based WGAN models with the proposed Capsule Network based discriminative GANs for both - conditional and non-conditional generation.

As a part of the assessment, we have trained our architectures on multiple datasets. We have compared the results from our proposed architectures with the images generated by Improved Wasserstein GAN, that utilizes a CNN critic with a similar number of backpropagation trainable parameters. We can see in Fig.

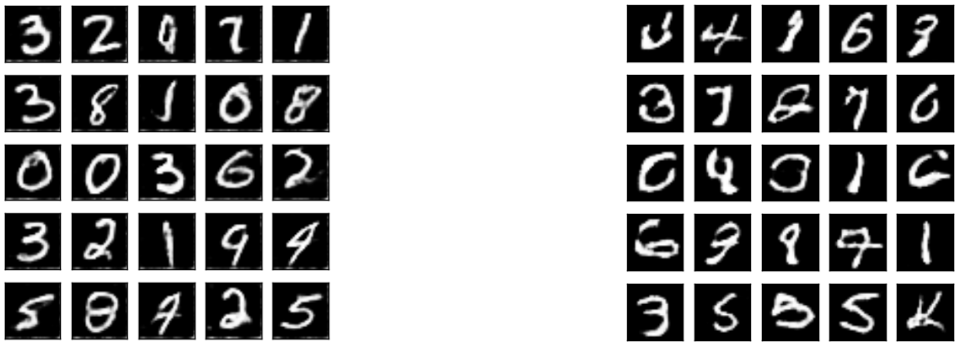

Figure 4.1.1: Images on the left are generated by our architecture whereas, the ones on the right are generated by Improved Wasserstein GAN, both trained over 5 epochs.

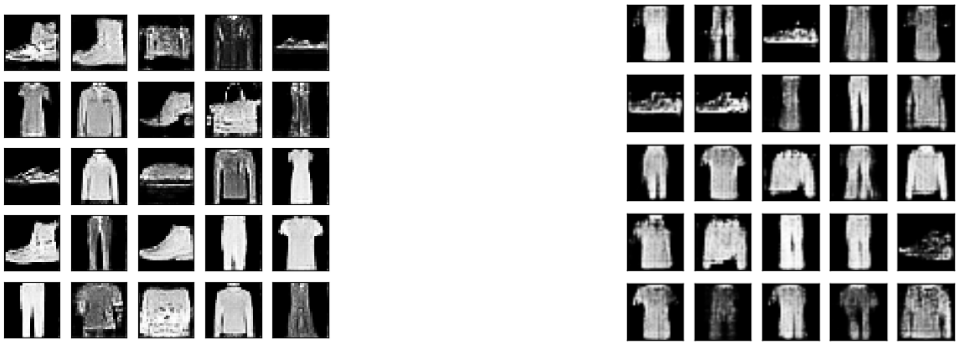

Figure 4.1.2: Images on the left are generated by our architecture whereas, the ones on the right are generated by Improved Wasserstein GAN, both trained over 5 epochs.

Figure 4.1.3: Images on the left are generated by Auxiliary-Conditional Improved Wasserstein DCGAN trained over MNIST for 100 epochs Whereas, the images on the right are generated by Split-auxiliary Conditional Capsule GAN trained over MNIST for 5 epochs. The digits being compared are - 3, 5, 6, 7, 9. We can can observe that the images generated by our model are visually much better than the images generated by DCGAN despite being trained over significantly lesser number of samples.

[4.1.1] that the proposed unconditional architecture, trained on MNIST [23] and Fashion-MNIST [42] datasets, synthesizes images with high visual fidelity even in the earlier epochs. We have also trained our model on CelebA [43] to synthesis images with a resolution of 64x64. The results in Fig. [4.1.4] have been generated in 50 epochs.

We also synthesized rotated MNIST images by training Conditional Improved Wasserstein GAN and Conditional Split-Auxiliary Capsule GAN on an MNIST dataset which has images with rotations of: 0, 90, 180, 270 degrees. We can see in Fig. [4.1.3], that despite our architecture being trained for only 5 epochs, the quality of the images generated by it surpasses the quality of the images generated by Conditional Improved Wasserstein GAN which was trained over 100 epochs over the same dataset. One can clearly see that the our model has been able to pick up really strong features even in the earlier epochs to distinguish between a '6' and a rotated '9'. It is easy to see that our model has learned to distinguish between the two by the curvature of the tail in the two digits. These results bolster our key idea of having a split-classifier that optimizes the class conditionals and the process of critiquing simultaneously in one network.

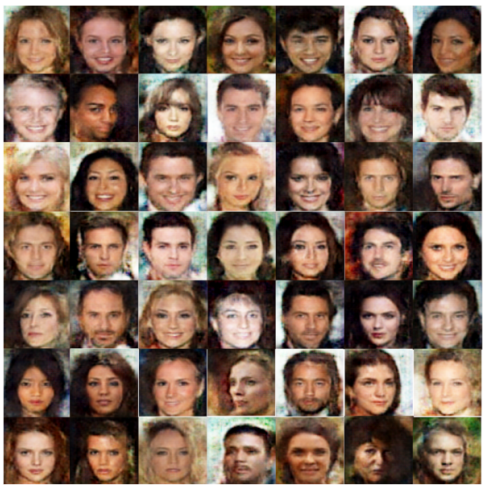

Figure 4.1.4: 64x64 resolution images generated from CelebA in 50 epochs by Capsule GAN

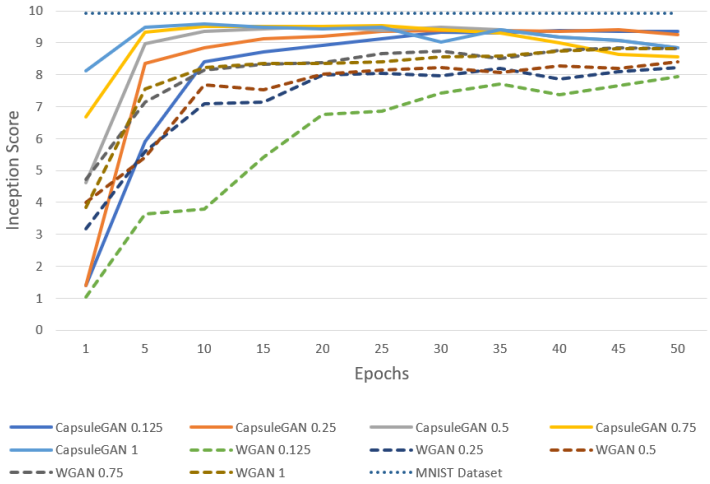

Figure 4.1.5: Inception Scores of Capsule GAN and IWGAN being trained over a fraction of the MNIST dataset

The images synthesized by our architectures achieve state-of-the-art results for MNIST dataset and that too with significantly lesser amount of training data as well as training epochs. Referencing Fig. [4.1.5], we can see that the Capsule GAN architectures achieve very high Inception Scores from the early training epochs itself. The ability to achieve such high scores by looking at just 1/8*th* of the samples is a testament of the ability of Capsule Networks to encode a dataset's distribution in a much better manner. The top Inception Score achieved by our architecture is 9.58, whereas the test partition from MNIST achieved a score of 9.92. The conditional architecture were also able to achieve state-of-the-art performance with a score of 9.99, achieving almost the theoretical limit for equal distribution of perfect samples from each class.

We calculate the FID Score for the generated images using a trained Capsule Network classifier. We use a Capsule Network because it achieves state of the art performance on the datasets with faster training time. With Inception Network, we were able to achieve only 98.3% accuracy on MNIST, whereas we were able to get 99.72% accuracy with Capsule Networks. Since Capsule Network can capture better features than the Inception Network, it will be a better judge of the features present in the synthesized images.

For MNIST, where use the activations of the Secondary Capsule whose length is the maximum. Whereas, for CelebA, there can be multiple classes present in an image and therefore, we consider a class to be active if the vector length corresponding to its Secondary Capsule is greater than 0.5. The Secondary Capsules of active classes are then stacked class-wise for calculating the FID. We show the class-wise FID scores for all the classes of MNIST and the first 10 classes for CelebA. Referencing Table [4.1.1], we can see that our model achieves significantly better FID scores on MNIST as well as CelebA.

4.2 Effects of Equivariance and Invariance

While optimizing the GAN loss function, one would want that the GAN generator is able to capture the entire manifold of the training images, thus, leading to visual fidelity and diversity. In Fig [4.2.1], there are 3 different manifolds - training image manifold, manifold captured by the generator and the complete image manifold. It is possible that the training images themselves may not be able to capture the entire manifold of the possible images, therefore, in this case, the training image manifold is a smaller subset of the possible manifold. Elaborating on Fig. [4.2.1], *A* represents the region of the training images covered by the generator, whereas *B* represents the region of training images missed by the generator which corresponds to lack of coverage. Region *C* represents the manifold covered neither by the training images, nor the generator whereas, *E* represents the region of true novelty, where the generator is producing images that do not belong to the manifold of the training samples but are still visually correct and region *D* represents the region of incorrect samples being generated by the generator. The ideal scenario would be the maximization of regions *A* and *E* and minimization of regions *B* and *D*. Since region *C* is intractable, it might be difficult to to calculate the region *E*. Therefore, in the following discussions we will be looking out for only maximizing the region *A* over *B* and minimizing the region *D*.

Upon visualizing the Capsule representations of the MNIST images in Fig. [4.2.2], we can see that the images generated by Capsule GAN has a greater coverage over the secondary principal axis in comparison to that of the images generated by the IWGAN. We can see that the overlap region, for Capsule GAN is much larger than the overlap region for IWGAN. Most of the images generated by IWGAN are packed closely and have lesser coverage over the real data manifold. The point to note is that despite achieving visual fidelity in images, IWGAN was not successful in capturing the complete manifold discovered by

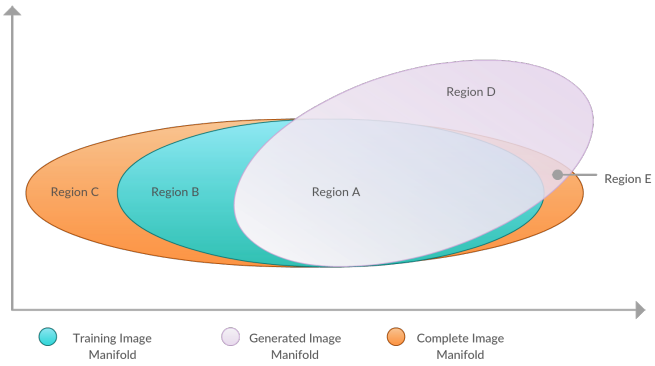

Figure 4.2.1: Overlap of complete image manifold, training data manifold and generative model generator model learned manifold

the Capsule Network projection, in other words, it had inferior coverage as well as diversity in the attributes of the samples it generated. Since, these are projections from the Capsule space, we attribute such a behaviour of the IWGAN to the lack of ability of the CNN critic to learn the features unearthed by the Capsule Network. The principal of serializing positional invariant layers leads to a systematic failure of coverage, making the IWGAN oblivious to the features that Capsule GAN is able to pick up.

Fig. [4.2.3] shows the Capsule projections of synthesized rotated-MNIST into $\mathbb{R}^2$ using PCA. It is strongly evident here that the Capsule GAN strongly benefits with the equivariance in the critic over CNN critics with progressive invariance. images synthesized by Capsule GANs show greater coverage and significantly lesser extraneous sections over the true manifold in comparison to that of the IWGAN with a CNN critic. Class-0 images generated by our architecture shows greater coverage over the second principal axis in comparison to the CNN based critic architectures. Similarly, we can observe that Class-7 images generated by the CNN-based critics are scattered all over the space with images being out of the manifold, whereas we can certainly see a lesser degree of images generated by

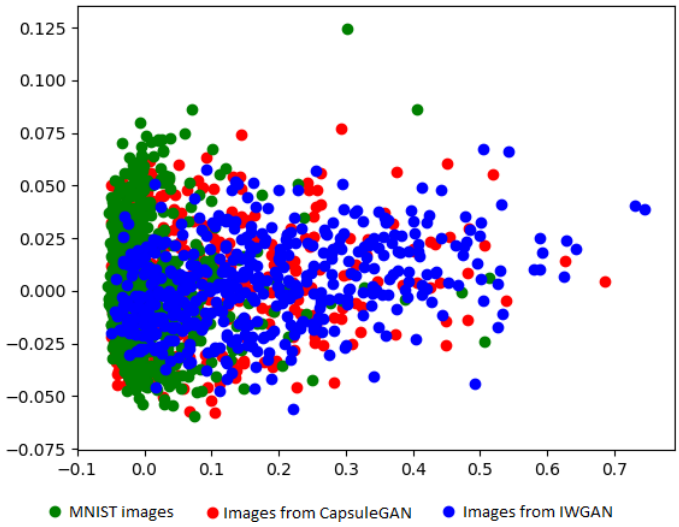

Figure 4.2.2: Projection of Capsule representation of MNIST class-5 images in $\mathbb{R}^2$ by projecting Capsule vectors on the two largest PCA components of Capsules corresponding to the training images

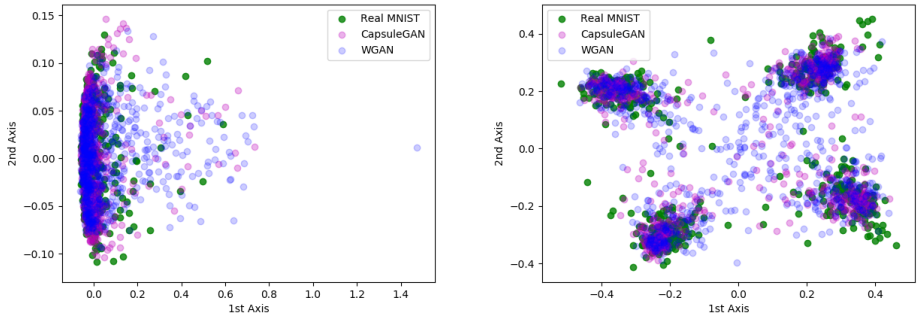

Figure 4.2.3: Projection of Capsule representation of Rotated-MNIST class-0 images (left) and class-7 (right) in $\mathbb{R}^2$ by projecting Capsule vectors on the two largest PCA components of Capsules corresponding to the training images

our architecture lying out of manifold.

Another set of experiments shows that the level of invariance and equivariance

in a network allows alters the structure of the representations learned. These experiments involve varying invariance in the Capsule Discriminator by addition of different degrees of max and average pooling layers into the Primary Capsule layer. By nature, since max-pooling passes all but one activation, it adds greater invariance in comparison to average-pooling, as averages would tend to be an aggregation of all the local activations.

As a part of these experiment, Capsule GANs with varying amounts of invariance are trained on the MNIST dataset with no rotations. Invariance is varied by the kind of pooling layer - max and average, and the strength of pooling over Capsules - the pooling kernel size. The images generated by these models are then evaluated by the Inception Score (which takes into account the quality of the samples, i.e., penalize for high entropy in intra-class activation distributions; and diversity of the samples generated by promoting high inter-class entropy) and the accuracy of classifying images in MNIST datasets with maximum rotations of 0, 5, 10, 15, 20 and 25 degrees, clockwise or anti-clockwise.

Referencing Fig. [4.2.4], we can see that in the plots for the model with vanilla Capsule Network Discriminator, there is a strong positive correlation between the performance of the GAN Discriminator on the datasets and the Inception scores achieved by the images generated by the GAN. This means that the critic has learned a representation that performs well even on novel view points while also helping generate diverse samples of high visual fidelity. However, this trend is disturbed by the addition of pooling layers which essentially distort the extracted spatial features and thus, obstruct what the following layers see. As can be seen in the plots, this the disturbance in the trends becomes more apparent with stronger pooling effects.

Divergence from the trends in the critics with the average pooling layers isn't as strong as in the critics with max-pooling layers. In average pooling layers, we can see a general dip in the accuracy of the critic in classifying the samples with

induced rotations as real despite having the GAN deliver images with just slightly lower inception scores. However, in case of max-pooling layers, we can see a strong deviation from the original trend as the critics with the highest accuracy, that is, critics that classified most of the images with induced rotations as "real", also led to the worst inception scores - meaning either images with low visual fidelity or lack of inter-class diversity. Further analysis showed a drop in both visual fidelity and inter-class diversity. Also, points with high inception score have a low accuracy - meaning that the model is producing the images of visual fidelity and inter-class diversity but while learning a limited view of the manifold as such critics end up classifying most of the "real" images as "fake". This shows that the models with max-pooling are learning poorer representations of the dataset which leads them to either classify points from the extraneous manifolds as "real" or learning a limited view of the true manifold. Thus, leading to poorer critics and poorly synthesized images.

4.3 Inferences from Experiments

In the earlier sections, we have discussed about Capsule Networks being able to capture spatial relationships better and therefore, better features in images when compared to CNNs. A direct impact was seen in the quantity of the training data and epochs required to achieve state of the art results. The Primary Capsule layer looks at the complete image and all the lower-Capsules contribute to the Capsules in the Secondary Capsule layer weighted by the agreements between them. This gives a better global view of the features when compared to CNNs which only look at local features progressively through the layers. Thus, Capsules are able to capture certain features which are completely missed out by the CNNs.

Apart from having an impact on the training statistics, there is a strong impact

on the diversity of the images generated by a GAN having a critic that learns invariant features. Referring to one of our key ideas about Wasserstein GAN critics that the generators image synthesis quality is only as good as the critic that judges it. If the critic is unable to capture all the aspects of the true image distribution, then the Wasserstein Loss tries to pull the distribution of the synthesized images to the more occluded critic's understanding of the distribution. Thus, limiting the overall coverage of the generated distribution over the actual distribution. Experiments from the previous sections show that such spatially invariant representations of images and features indeed lead to poorly learned representations of the datasets and lead either to a lower coverage of the manifold or learning of an extraneous manifold. Thus, synthesizing poorer images.

Therefore, a critic architecture that can cover equivariance over a transformation that is intrinsic to a dataset, can significantly boost the performance of the GAN in terms of the speed of convergence and image quality. For example, datasets with intrinsic rotations in them can be handled with Harmonic CNNs [41], or with Capsule Networks with Harmonic feature extractors.

Dataset	Model	0	1	2	3	4	5	6	7	8	9
MNIST	IW DCGAN	2.07	1.74	3.00	2.24	3.17	3.59	2.29	1.25	3.56	2.61
	Capsule WGAN (Ours)	0.30	0.16	0.46	0.15	0.26	0.48	0.42	0.15	0.41	0.35
	Cond. IW DCGAN	0.59	0.23	0.97	1.777	0.25	1.559	0.60	0.51	1.372	0.96
	Cond. SAC GAN (Ours)	0.13	0.15	0.13	0.17	0.20	0.13	0.12	0.24	0.15	0.16
CelebA	IW GAN	0.73	0.94	0.46	1.66	1.345	0.41	1.53	1.953	2.82	0.56
	Capsule WGAN (Ours)	0.42	0.79	0.44	1.28	2.16	0.24	1.19	1.93	0.73	0.31

Table 4.1.1: FID Scores ($\times 10^{-2}$) for GANs trained on MNIST and CelebA datasets over classes 0 - 9. Lower is better.

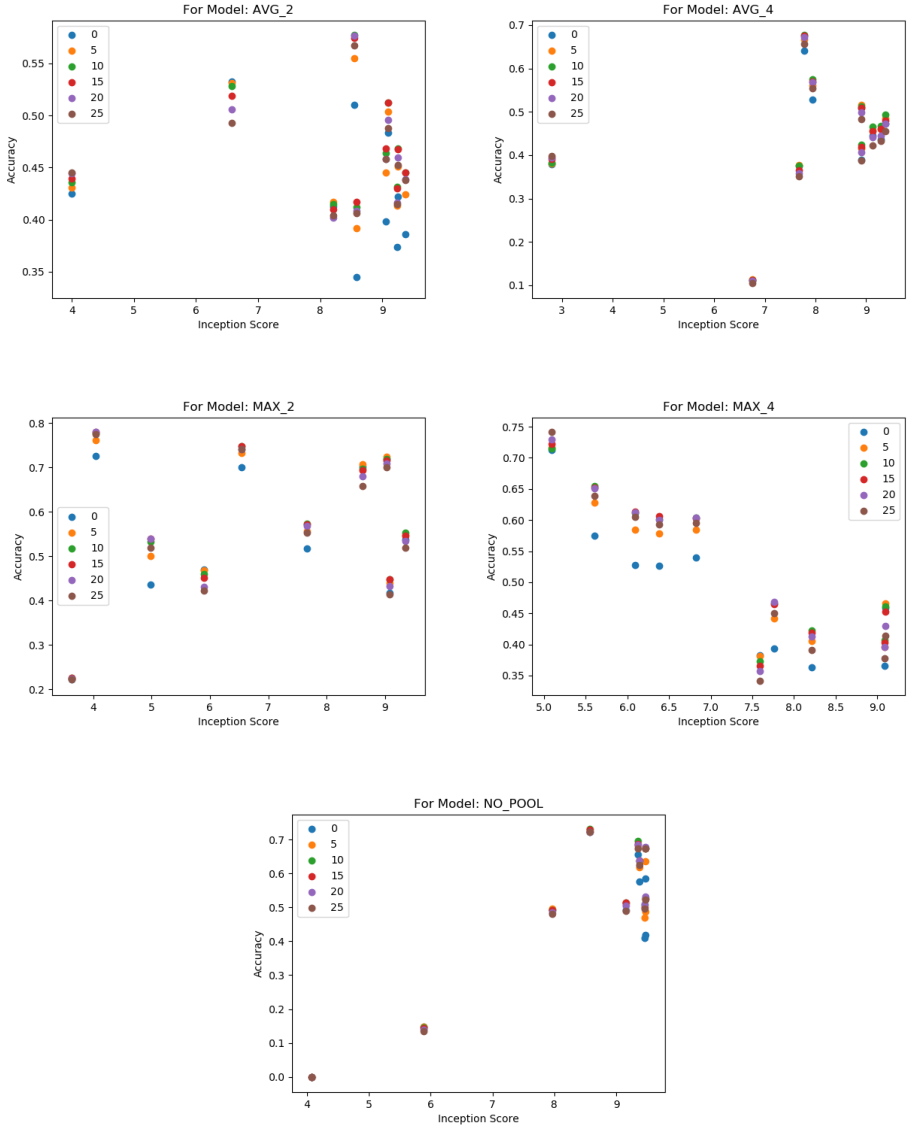

Figure 4.2.4: We can see that the model with no pooling layers (bottom) in the Capsule Discriminator has a strong correlation with the increase in the Inception score of the model. The deterioration in accuracy of the discriminator starts with the introduction of max and average pooling of a pooling size of 2 (middle and top left respectively). With application of stronger max and average pooling layers with kernels of size 4 (middle and top right respectively), we can see a stronger deviation in the discriminator accuracy with respect to its accuracy in classifying the digits in the dataset with rotations.

5

Conclusion

Building up on the foundation that positional-equivariance is superior to positional invariance in image based tasks, successive convolutional layers, especially with the use of intermittent pooling layers, leads to a lossy compression of the image. On the other hand, models with bake d-in equivariance, like Capsule Networks, learn representations that disentangle meta-features from the features themselves. This aids them become better feature learners which enables them to get better performance in comparison to CNNs, even with significantly lesser amount of training data and training epochs.

Since the critic in a GAN is the torch bearer for the process of manifold learning, incorporating networks with the right kind of equivariance can be be beneficial for faster learning of the manifold, as well as for the quality of the manifold being learned. This document explored that spatial-equivariance is a

better property than spatial-invariance for the task of image-synthesis and thus, incorporating Capsule Networks as critics yields much better performance in terms of the speed of manifold learning, image quality and image diversity.

The fact that replacing Convolutional Neural Networks with Capsule Networks could bring significant improvements to the overall performance of GANs, points us in a direction which encourages exploration of various different kinds of equivariances for different tasks. A task-based equivariant property in a critic can significantly improve the performance of the GANs and also reduce pave way for faster manifold learning with lesser amount of data.

Another direction to look forward into is the inversion of Capsule Network in order to use it as a Generator. This would require decoding the images represented in a Capsule to be mapped back into the real image space. However, inverting the Dynamic Routing process is non-trivial but to overcome this, one might look into the Matrix Capsules that use Expectation Maximization Routing.

References

[1] Book reviews. *Technometrics*, 57(3):440–443, 2015. doi: 10.1080/00401706.2015.1070657. URL https://doi.org/10.1080/00401706.2015.1070657.

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

[3] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[4] Ali Borji. Pros and cons of gan evaluation measures. *arXiv preprint arXiv:1802.03446*, 2018.

[5] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[8] Kunihiko Fukushima. Neocognitron. *Scholarpedia*, 2(1):1717, 2007.

[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] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[11] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and R Venkatesh Babu. Deligan: Generative adversarial networks for diverse and limited data. In *CVPR*, pages 4941–4949, 2017.

[12] 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.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[14] Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[15] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. 2018.

[16] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1): 215–243, 1968.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[18] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1857–1865. JMLR. org, 2017.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Jyri J Kivinen and Christopher KI Williams. Transformation equivariant boltzmann machines. In *International Conference on Artificial Neural Networks*, pages 1–9. Springer, 2011.

[21] 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.

[22] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[23] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6): 555–559, 2003.

[26] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[27] Sparsh Mittal. A survey of fpga-based accelerators for convolutional neural networks. *Neural computing and applications*, pages 1–31, 2018.

[28] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[29] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[30] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

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

[32] Avraham Ruderman, Neil C Rabinowitz, Ari S Morcos, and Daniel Zoran. Pooling is neither necessary nor sufficient for appropriate deformation stability in cnns. *arXiv preprint arXiv:1804.04438*, 2018.

[33] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[34] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[35] Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1233–1240, 2013.

[36] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

[38] Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 868–873. AAAI Press, 2005. ISBN 1-57735-236-x. URL http://dl.acm.org/citation.cfm?id=1619410.1619472.

[39] 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.

[40] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. In *Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017. URL https://vision.cornell.edu/se3/wp-content/uploads/2017/04/CSN_CVPR-1.pdf.

[41] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

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

[43] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3676–3684, 2015.

[44] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[45] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[46] Richard Zhang. Making convolutional networks shift-invariant again. 2018.