WORCESTER POLYTECHNIC INSTITUTE

MASTER'S THESIS

Cycles Improve Conditional Generators

Author:
Alexander MOORE

Advisor:
Dr. Randy PAFFENROTH

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Data Science.

Spring 2021

Advisor	Date
Reader	Date

Abstract

Alexander MOORE

Cycles Improve Conditional Generators

Learning information-dense, low-dimensional latent representations of high-dimensional data is the thesis of deep learning. The inverse problem of learning latent representations is data generation, in which machines learn a mapping from information-dense latent representations to high-dimensional data spaces. Conditional generation extends data generation to account for labelled data by estimating joint distributions of samples and labels. This thesis connects learning meaningful latent representations through compressive and generative algorithms and contains three primary contributions to the improvement and usage of conditional GANs. The first is three novel architectures for conditional data generation which improve on baseline generation quality for a natural image dataset. The second is a novel approach to structure latent representations by learning a paired structured condition space and weakly structured variation space with desirable properties. Third, a novel application of conditional data generation to a chemical sensing task with beneficial leaking augmentations for extremely low-data paradigms ($n < 100$) demonstrates that conditional data generation improves the testing performance of downstream supervised models.

Acknowledgements

Thank you to my advisor Dr. Randy Paffenroth and my wonderful family.

Contents

List of Figures **v**

List of Tables **vi**

1 Introduction **1**
1.1 Summary . . . 1
1.2 Background . . . 2
1.2.1 The Generative Assumption . . . 2
1.3 Background . . . 4
1.3.1 Autoencoders . . . 4
1.3.2 Variational Autoencoders . . . 5
1.3.3 Generative Adversarial Networks . . . 6
1.3.4 VAE-GAN . . . 7
1.3.5 Discriminators . . . 8
1.3.6 Conditioning . . . 8
1.3.7 Conditional Autoencoders . . . 9
1.3.8 Conditional GANs . . . 10
1.3.9 Auxiliary Classifier GAN . . . 11
1.4 Model Architecture . . . 12
1.4.1 Hyperparameter Selection . . . 12

2 Architecture Contributions **13**
2.1 Autoencoder-GAN . . . 13
2.2 Inverse Autoencoder-GAN . . . 15
2.3 Cycle Autoencoder-GAN . . . 16

3 Results **18**
3.1 Quantifying Generative Quality . . . 18
3.1.1 Low-Data Regime . . . 21
3.1.2 Snapping . . . 23
3.1.3 Generative Diagnostics . . . 24
3.2 Augmenting Training Data . . . 24
3.3 Cyclical Models Perform Classification . . . 27

4 Application **28**
4.1 Chemical Sensing . . . 28
4.2 GANs for Chemical Sensing . . . 30
4.2.1 Leaking Augmentations . . . 31
4.3 Application Results . . . 32
4.3.1 Comparing Stochastic Augmentation to GAN Augmentation . 33

A Appendix A **36**
A.1 Factors of Variation . . . 36
A.2 Model Architectures . . . 37
A.2.1 Encoding Architecture . . . 37
Conditional Encoders . . . 37
Discriminator Architecture . . . 38
Predictor Architecture . . . 39
A.2.2 Decoding Architecture . . . 39
Unconditional Generation . . . 39
Conditional Generation . . . 40
A.3 Structuring . . . 40
A.4 Chemical Sensing . . . 41
A.5 Generated samples per model . . . 41
A.6 Specific Model Architecture . . . 41
A.6.1 GAN . . . 41
A.6.2 Conditional GAN . . . 41
A.6.3 Autoencoder . . . 41
A.7 FID Extended . . . 41

Bibliography **46**

List of Figures

1.1 This thesis proposes a family of cycle-inducing models which improve on baseline generators. 1
1.3 9 CIFAR samples 3
1.4 Natural images lie near a low-dimensional manifold 4
1.5 Autoencoder Architecture. 4
1.6 A generator and a discriminator make a GAN. 7
1.7 Conditioning adds structure to a latent space. 9
1.8 Conditional Autoencoders condition during encoding and decoding . 10
1.9 Conditional GAN architecture. 11
1.10 Auxiliary Classifier GAN architecture. 11

2.1 Hidden Conditional Autoencoder-GAN Architecture. 14
2.2 Inverse Conditional Autoencoder-GAN Architecture. 15
2.3 Cycle Conditional Autoencoder-GAN Architecture. 16

3.1 CIFAR-10 is a benchmark natural image data set. Image from Krizhevsky, Hinton, et al., 2009. 18
3.2 Sample CCAEGAN conditional generation of image classes 21
3.3 Higher estimated manifold dimensions indicate more variation in generated classes. 24
3.4 Testing accuracy increases linearly with conditional image quality. . . 26

4.1 28
4.2 Baseline regression task results given variable exposure windows. . . . 29
4.3 Sensor responses are not linear with exposure magnitude. 30
4.4 A training sample and an approximated sample at the moment of chemical exposure. 33
4.5 Testing loss with a variable number of generated samples. 34

A.1 Unconditional generation by the GAN. 41
A.2 Conditional generation by the CGAN. 42
A.3 Conditional generation by the ACGAN. 42
A.4 Conditional generation by the CAEGAN. 42
A.5 Conditional generation by the ICAEGAN. 43
A.6 Conditional generation by the Cycle-CAEGAN. 43
A.7 Autoencoder Architecture, simplified 43
A.8 Hidden Conditional Autoencoder Architecture. 44
A.9 Inverse Conditional Autoencoder Architecture. 44
A.10 A minimal GAN architecture diagram. 45
A.11 A minimal CGAN architecture diagram. 45
A.12 Non-conditional Autoencoder 45

List of Tables

1.1 Conditional and unconditional Fréchet scores across model architectures. Lower is better. 2
1.2 Testing loss for the combination of stochastic augmentation and data set augmentation with a conditional GAN. Lower is better. 2

3.1 Fréchet distances across model architectures. Lower is better. 20
3.2 Small-data Fréchet distances across model architectures. Lower is better. The GAN has not been bolded as it serves as a demonstration of the kind of generative quality one can expect, but fulfills a different task to the other models. 22
3.3 Differences between quarter-data and small-data Fréchet distances across model architectures. Lower is better. 22
3.4 Proposed alterations to the autoencoding model inductive bias on the conditional encodings. Lower scores are better. 23
3.5 Test accuracy when training data is augmented with fake data. Higher is better. 25
3.6 Top-1 test accuracy without regard for generation. Higher is better. . . 27

4.1 Testing loss given by ten trials using the optimal hyperparameters from the validation stage. 33
4.2 Probability of retaining the null hypothesis given the distribution of test performances. Low values indicate higher certainty that the mean of the distributions is unequal. Bold results indicate a certainty below the heuristic threshold of $p = 0.05$. 34

A.1 Optimal hyperparameters for each paradigm by random 10-fold cross validation. 41

Chapter 1

Introduction

1.1 Summary

Conditional generative adversarial networks are increasingly used in applications (Karras et al., 2018). The extension of generative modelling to conditional generative modelling begets many interesting opportunities from architecture and training algorithm perspectives. This thesis proposes a family of three models which improve upon baseline conditional generative GANs by inducing cycles inspired by CycleGAN (Zhu et al., 2017) with extensions to conditional data generation. Cycles are defined as a mapping from an initial space to an intermediate space and back with low reconstructive error such that each image of the distribution fulfills some metric.

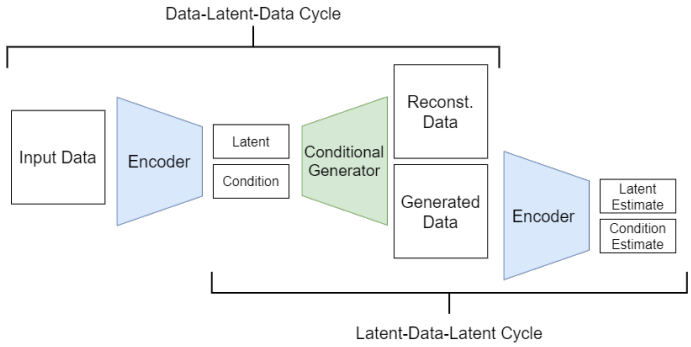


FIGURE 1.1: This thesis proposes a family of cycle-inducing models which improve on baseline generators.

The efficacy of a generator can be quantified with established metrics such as the Fréchet Inception Distance (Heusel et al., 2017). Quantifying the quality of a *conditional* generator requires an extension accounting for the joint probability distribution of classes and images, given by the Fréchet Joint Distance (DeVries et al., 2019). These two metrics are introduced in Chapter 3 and will be the primary measure of quality for the baseline and proposed models of this thesis. Table 1.1 quantifies the performance of two baseline conditional generators and the proposed cyclical models of this thesis in a controlled experiment on the natural image data set CIFAR-10 (Krizhevsky, Hinton, et al., 2009).

Lowering the FID and FJD scores of a generative model corresponds to generating samples closer to the data manifold given by an unseen holdout subset of the CIFAR-10 data set, and correspondingly generating higher-quality, more realistic naturally images. Generating high-quality natural images is a competitive environment for machine learning, albeit with few practical applications (Mishra et

	FID	FJD
CGAN	44.37 ± 0.54	51.97 ± 0.59
ACGAN	43.91 ± 0.31	61.59 ± 0.62
CAEGAN (ours)	37.67 ± 0.37	46.42 ± 0.47
ICAEGAN (ours)	39.06 ± 0.25	48.16 ± 0.27
CCAEGAN (ours)	$\mathbf{35.72} \pm 0.27$	$\mathbf{43.22} \pm 0.32$

TABLE 1.1: Conditional and unconditional Fréchet scores across model architectures. Lower is better.

al., 2018, Odena, Olah, and Shlens, 2017). Generating high-quality *labelled* natural images, however, is a practical pursuit: Chapters 3 and 4 demonstrate applications of using conditional generation as a data set augmentation, in which both real and generated training samples work together to improve the training of a supervised learner. Models trained in this manner exhibit lower testing error than models trained with the training set alone: the inclusion of conditional generation for chemical sensing also significantly improves the testing performance of a downstream supervised model which utilizes some amount of generated samples. In this paradigm, the desired task is a machine with low testing multiple regression error predicting the chemical exposure on a testing set. Table 1.2 demonstrates that the inclusion of stochastic data augmentation and data set augmentation with a conditional GAN dramatically improves testing performance.

	No Augmentation	With Augmentation
No CGAN	23.11 ± 0.23	29.46 ± 0.32
With CGAN	24.05 ± 0.22	$\mathbf{18.42} \pm 0.22$

TABLE 1.2: Testing loss for the combination of stochastic augmentation and data set augmentation with a conditional GAN. Lower is better.

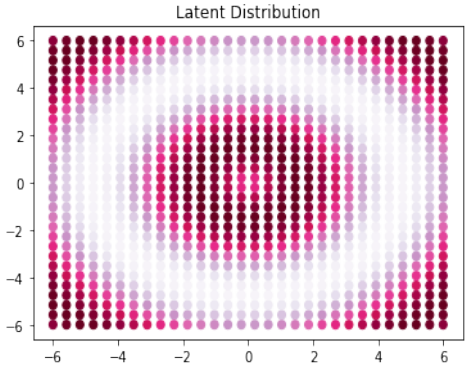
In Chapter 1, language, notation, and current models specific to conditional generative machine learning are introduced. Chapter 2 details the architecture and training algorithm of the novel cyclical models we propose. Chapter 3 provides a proof-of-concept comparing baseline data generation approaches to cyclical models on the CIFAR-10 natural image dataset, with additional experiments designed to dissect model behaviors and scoring metrics. Lastly, Chapter 4 details a novel application of conditional generation with beneficial leaking augmentations to a chemical sensing application which improves baseline supervised learners.

1.2 Background

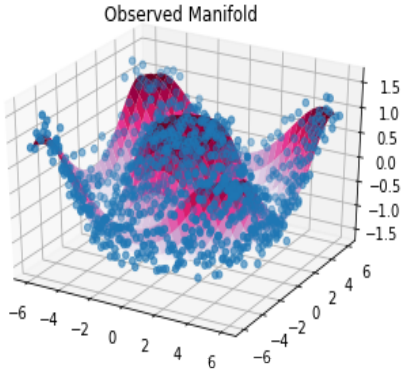
1.2.1 The Generative Assumption

The foremost assumption made in representation and unsupervised learning is the generative or manifold assumption. This regime asserts that real data samples x in data space $X = \mathbb{R}^n$ are distributed near some lower-dimensional manifold $M \subseteq X$ up to some random perturbation ϵ. Though the data in plot 1.2b lie in the data space $X = \mathbb{R}^3$, we observe they lie *near* the lower dimensional manifold $M \subseteq \mathbb{R}^3$, a

surface defined by some function on the latent space $\mathbb{R}^2$. The semantic information of the data (here, the z-coordinate) is uniquely described within the low-dimensional manifold up to some small perturbation ϵ.



(A) The latent distribution.



(B) Data samples lie near the manifold.

Unsupervised learning often posits that this data manifold is substantially lower than the data domain X, which is particularly evident in spaces such as high-resolution natural images. Demonstrated in plot 1.4, even for a linear dimensionality estimation method the estimated manifold of natural images is extremely low: 98 dimensions made of linear combinations of the data dimensions are needed to express 90% of the variability in the training data. Relative to the data dimension of 32×32 pixel RGB square images defined by 3072 real numbers, a conservative estimate is that only 3% of the dimensions are needed to express 90% of the variation of these images.

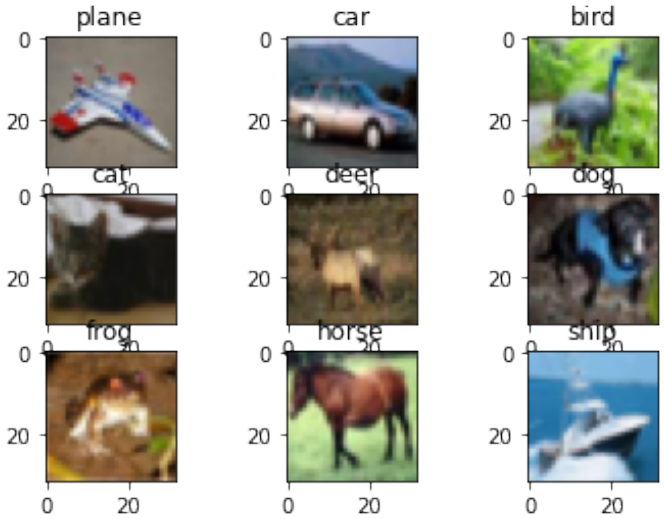


FIGURE 1.3: 9 CIFAR samples

Learning more flexible, non-linear models can provide even better dimensionality reduction. We will demonstrate how learning rich representations of these compressed spaces can lead to superior predictive models, inference, and ultimately generation of samples drawn from estimations of the training distribution.

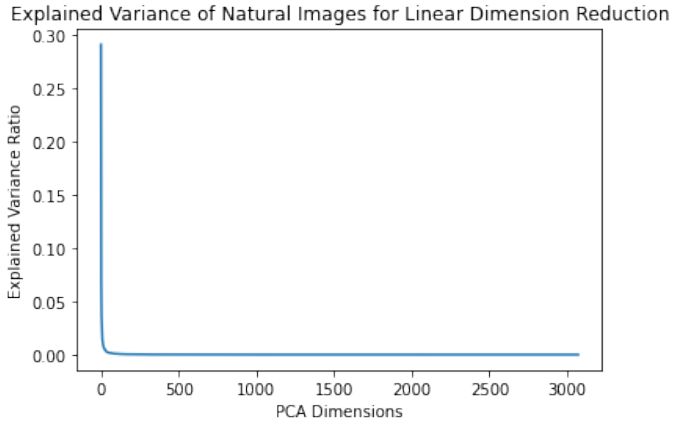


FIGURE 1.4: Natural images lie near a low-dimensional manifold

1.3 Background

1.3.1 Autoencoders

Autoencoders are a family of machine learning models which learn an approximation to the identity function I on training samples x given by $\hat{I}(x) = \hat{x}$. Rather than a simple identity, the autoencoder must recover samples under a bottleneck condition that the data must pass through some low-dimensional real-valued space before being reconstructed back to the data space. These models leverage representation-learning layers for compression which move the data to some low-dimensional, information-dense space where dimensions represent variation in the semantic meaning of their corresponding data-space samples.

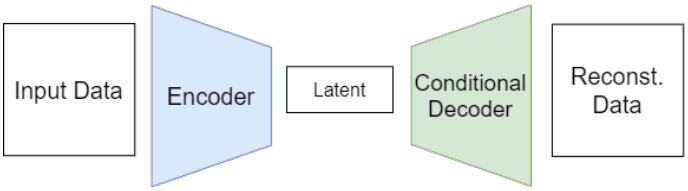


FIGURE 1.5: Autoencoder Architecture.

The process to reconstruct samples $\hat{x}$ after encoding them to a low-dimensional representation z can be decomposed into an encoder $Enc(x) = z$ and decoder $Dec(z) = \hat{x}$ functions, which are composed to create the identity approximation $Dec(Enc(x)) \approx x$. Autoencoders are typically trained by minimizing the loss between an input sample x and the corresponding autoencoder image $f(x)$, given by a mean reconstruction error over the elements i of each sample, specifically pixels for image data. For one sample, the pixel-wise difference is defined as the distance between each of n pixels x_i in training image x, and each pixel under the image of the learned identity approximation $\hat{I}(x)_i$ in the reconstructed image $\hat{x}$:

$$\mathcal{L}_{MSE}(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{x}_i - x_i)^2 \tag{1.1}$$

Depending on the domain of the preprocessed data a binary cross-entropy loss is often appropriate as well:

$$\mathcal{L}_{BCE}(x, \hat{x}) = -\frac{1}{n}\sum_{i=1}^{n} x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i) \tag{1.2}$$

The per-sample error (implemented as the mean over a batch of sample) given by Equations 1.1 or 1.2 is back-propagated through the autoencoder, so that both decoding and encoding weights are updated. Using this loss for gradient descent the decoder learn to map from low-dimensional representations back to the data space, and simultaneously allows the encoder to learn a distribution which lends itself to easier decoding. Significantly, the learned distribution has some inherent meaning as points $z \in Z$ have some learned bijective relationship with data point $x \in X$ on the decoder-learned manifold. Changes to the latent vector under decoding induce changes to the decoded sample's semantic content, and contemporary research such as Locatello et al., 2019 measure performance of disentangling this latent space such that representations dimensions correspond one-to-one with data space semantic meaning.

The autoencoder architecture is a vital departure from linear manifold learning algorithms, and is a blank slate for inducing architecture alterations and penalties which can dramatically change the content and interpretation of the completely unregulated latent space. The space of latent representations is targeted for modifications as structuring this information-dense layer opens opportunities for data generation and inference, discussed in Sections 1.3.2, 1.3.4, and 1.3.7

1.3.2 Variational Autoencoders

A significant modification to autoencoders, both in architecture and adopted priors, is the variational autoencoder (VAE) proposed by Kingma and Welling, 2014. The VAE distinguishes itself from autoencoder architectures by imparting the assumption that latent representations should be distributions, not vectors. The inductive assumption that latent representations are better interpreted as distributions than vectors changes the latent space of the bottleneck dimension from a $\mathbb{R}^n$ space to a paired $\mu \in \mathbb{R}^n \bigcap \sigma^2 \in \mathbb{R}^n$ mean and variance space. Into this distribution-space data samples x are encoded as

$$Enc(x) \backsim \mathcal{N}(\mu_x, \sigma_x^2)$$

Encoding to a distribution is a significant departure from encoding to a vector. This interpretation means that a single input sample corresponds to a distribution of output samples under the deterministic decoder network, and the encoder must balance encoding the variance of the distribution against the reconstruction penalty that may be incurred by a random sample of the point.

The decoder architecture is identical to an autoencoder, with the exception that the latent vector is not specified by the encoder, but rather the learned distribution is *sampled* then decoded. The rephrasing of the latent representation vector as a representation distribution also redefines the VAE loss as

$$\mathcal{L} = \mathbb{E}_{q_\theta(z|x)}[\log p_\theta(x|z)] + \mathcal{L}_{prior}$$

Where the expectation $\mathbb{E}$ of the encoded sample distribution $q_\theta(z|x)$ given training data x is taken over the pixel-wise loss (Equation 1.1, 1.2) $logp_\theta(x|z)$, representing the decoded distribution corresponding to the encoding samples z. In addition

the VAE includes another penalty term on the prior:

$$\mathcal{L}_{prior} = -\beta D_{KL}(\log q_\theta(z|x)||p(z) \tag{1.3}$$

The deterministic reconstruction loss has been replaced with the expectation over the reconstructed random samples given the encodings $q_\theta(z|x)$ (Kingma and Welling, 2014), and second term D_{KL} is the KL-Divergence, an estimation of the distance between two probability distributions p and q. The D_{KL} term penalizes the distance between the multivariate standard normal and the distribution of the encoded representations. Including the KL-Divergence loss means that the VAE may choose to assign a low variance to an encoded point on the assumption that certainty in the sampling step will consistently lead to low reconstruction errors - however this will be balanced by the KL-Divergence penalty so only a select few difficult points may be sampled with low variability.

The assumption behind the stochastic decoding design is that sampling latent vectors instead of deterministic latent vectors provides the model with more generalizability, helps to mitigate overfitting with overly complex encoded distributions, and allows sampling of the latent space to generate images approximating the training distribution. The variational design choice enforces a strict structure on the latent space which can be tuned by a hyperparameter proposed in the β-VAE (Higgins et al., 2016), which anneals the coefficient on the KL-Divergence to allow increasingly more flexibility to the encodings and increasingly less variation in the sampling. This process yields one estimation of a disentangled representation of the training distribution, in which each dimension of the latent space corresponds to a single semantic feature of the data space samples. Similar methods for tuning the level of structure in a learned latent space are discussed in Section 3.1 and 3.1.2.

1.3.3 Generative Adversarial Networks

Generative Adversarial Networks are a broad new class of statistical models based on a game-theoretic min-max game for data generation. In this design two agents called the generator G, Figure 1.6a, and discriminator D, Figure 1.6b, alternate weight updates to maximize the adversary agent's error. The generator player samples from random latent space Z (often a multivariate standard normal) a vector as input to a learned deep network, then passes the output $G(z)$ to the input of D. The discriminator then evaluates the sample by returning a the probability that the sample was drawn from the training distribution, rather than the generator distribution. The agents update in turn, where the generator tries to minimize the objective against a maximizing discriminator solved by Equation 1.4.

The min-max optimization uses expectation over samples from the training set $x \approx p_{data}$ of the error of the discriminator given by $\log D(x)$. The discriminator error on real samples is balanced against that of error on generated samples given by the expectation over samples $z \approx p_z(z)$ decoded by the generator $G(z)$, where z is the multivariate standard normal prior (Goodfellow et al., 2014).

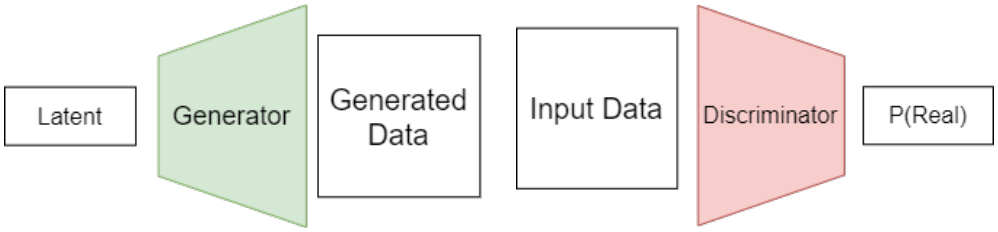
$$\min_G \max_D V(D,G) = \mathbb{E}_{x \backsim p_{data}}[\log D(x)] + \mathbb{E}_{z \backsim p_z(z)}[\log 1 - D(G(z))] \tag{1.4}$$

Vitally, the objective of the generator is not to generate plausible training samples. Rather, generating estimations of the training distribution by the generator and plausible evaluation by the discriminator is a by-product of the two agents trained to maximize each other's error. The min-max process which updates each model

according to the rival model error defines the nominal adversarial loss, and can induce instability in training where the adversarial process diverges from approximating the training distribution. The results of this thesis are discussed in this light, as regularization to the loss with additional model tasks (Chapter 2) is beneficial to generator quality (Section 3.1).

Adversarial training is sensitive to hyperparameter selection and training algorithms, and is prone to pitfalls in practice. This thesis will work in part address a common shortcoming in GAN training, which is the mode collapse. Mode collapse occurs when the generator output distribution collapses to produce a single point, sometimes of high quality, in the training distribution. Training divergence creates a chasing game where the discriminator and generator perform some cyclic update pattern, or converge to some multi-modal distribution of a limited number of points(Goodfellow et al., 2014).

Vital to adversarial training is the discriminator's ability to discern the proximity of a batch of data samples to the training distribution which drives the generator's decoded distribution to be of high quality. However, the discriminator's task is quantifiably easier than the generator's, especially early in training as evidenced by the frequency of mode collapse and overfitting (Donahue, Krähenbühl, and Darrell, 2016, Mescheder, Geiger, and Nowozin, 2018). For this reason the discriminator loss often converges to 0 before the generator can produce reasonable distributional estimation. Sometimes referred to as discriminator overfitting, the discriminator may memorize the training set and converge to 0 loss which by definition ends training as the min-max game cannot be updated further by the adversarial loss alone. Significant research in non-saturating and non-collapsing losses is directed at stabilizing against generator collapse to prolong training (Jolicoeur-Martineau, 2018, Arjovsky, Chintala, and Bottou, 2017).



(A) Generator architecture. (B) Discriminator architecture.

FIGURE 1.6: A generator and a discriminator make a GAN.

1.3.4 VAE-GAN

Previous work has proposed collapsing the decoder of an autoencoder and the generator of a GAN and learning the parameters jointly, as this thesis proposes and extends in Chapter 2. The work by Larsen et al., 2016 makes this assumption and trains a variational autoencoder (VAE) and GAN simultaneously. The authors make the additional alteration of re-phrasing the reconstruction loss as a distance in a representation layer of the discriminator. This triple criterion is expressed by 1.5 as the optimization over the VAE prior (KL-Divergence) given by Equation 1.3, the novel representation distance $\mathcal{L}_{llike}^{Dis_l}$, and the min-max GAN Equation 1.4:

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l} + \mathcal{L}_{GAN} \tag{1.5}$$

Arguably the power of the model comes from the ability to perform the nominal "autoencoding beyond pixels using a learned similarity metric" - meaning that the reconstruction error is no longer expressed as a pixel-wise difference between a sample and its image, but rather as the distance between the representation of the sample and its image in a layer of the discriminator. This leap from pixel-wise to image-wise reconstruction quality quantification comes from the intuition that a GAN discriminator learns an excellent function mapping samples to their perceived quality, which is the goal of reconstruction: minimizing the perceived difference between two samples. The loss component $\mathcal{L}_{llike}^{Dis_l}$ therefore is given by the distance between a sample x and its image $f(x)$ when each is embedded into some representation layer of the discriminator - Larsen et al., 2016 uses a depth of three. Intuitively this distance corresponds to the difference in representation learned by the discriminator, which is trained to evaluate the quality of samples by processing the entire image, rather than a pixel-to-pixel analysis. The usage of representation-space distances is also explored in Section 3.1 to quantify GAN performance.

1.3.5 Discriminators

Often, GAN research centers the importance of the generator as the product of a trained model: from the inception of the GAN in Goodfellow et al., 2014 to recent state-of-the-art (Zadorozhnyy, Cheng, and Ye, 2020), demonstrations and state-of-the-art contests rely on demonstrations of scores of the generative component. The generator is extremely useful, and success can be quantified by demonstrating success in some well-established metrics (3.1). Though discriminators often have the easier of the two tasks during training, it may harder to quantify performance of one. The discrimination task explicitly is to balance two losses: the loss to return a high probability when evaluating real samples, and the adversarial loss to return a low probability when evaluating generator samples. This task has been noted by Larsen et al., 2016 as a general-purpose description of a neural network who's task is to return the 'goodness' of a sample relative to some reference distribution.

In a logical leap, one could describe a discriminator as an abstracted distribution divergence evaluator. Whereas metrics such as the KL-divergence evaluate the distance between two distributions p and q, when these distributions are high-dimensional manifolds such as a training set of natural images and a generator's learned distribution solving for pixel-wise distributional distances quickly loses usefulness and becomes costly. The discriminator, however, is implicitly trained as a distance function between two distributions - this interpretation has even been formalized as the Wasserstein-GAN (Arjovsky, Chintala, and Bottou, 2017). Through adversarial training the discriminator is incentivized to learn a tight bound around the training data manifold, and by doing so becomes a powerful metric for the goodness of an estimation of the training distribution. This insight is leveraged by the VAEGAN (1.3.4), but additionally in a novel way in Section 2.2.

1.3.6 Conditioning

Conditioning refers to structuring inputs to a function with some additional information. This information is often a class label for multi-modal data but could be a semi-supervised learned or set of labelled targets. For the purposes of this paper's discussion, conditioning refers to a disjoint set of class labels for multi-modal data which may be the target for some downstream classification task, though in practice

facile changes to architectures would allow for intersecting classes or scalar conditions as shown in Figure 1.7.

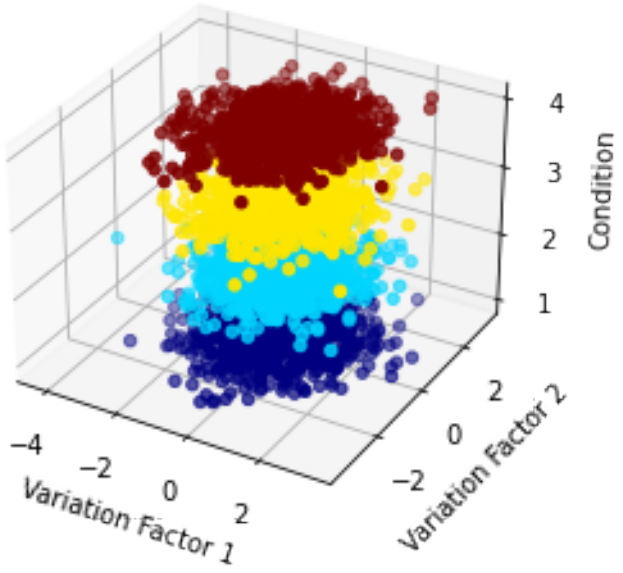


FIGURE 1.7: Conditioning adds structure to a latent space.

Figure 1.7 demonstrates the particularities of conditioning for deep learning. In the case disjoint class conditioning, each condition is an orthogonal dimension to each other when interpreted as one-hot conditioning. In the scalar conditioning paradigm, distributions in each dimension define the structure of the space.

The premise of conditioning is modifying functions $f(v) = x$ mapping from variation spaces from which multivariate standard normal samples are drawn to the data space X to include another component of information c, a condition which modifies the desired semantic content of the output. Rather than c being only an argument to a function, c specifically alters the geometric structure of the latent space in the deep learning model of conditioning as interpreted in this thesis. Structuring the latent space means that the generator in one interpretation maps from the joint (v, c) distribution to the conditional joint distribution $p(x|v, c)$. Figure 1.7 demonstrates the geometric interpretation of the structure as experienced by conditional decoders as the dimension of the problem is increased by the condition dimension. The variation factors are specified as standard normal distributions, while the condition space disjoints the distributions by the number of classes. In this instance, the condition is one-dimensional of a scalar magnitude, but often classes are disjoint one-hot encodings each represented by one dimension, or some real-valued vector embedding (Wieting and Kiela, 2019 demonstrates one possible random-encoding method for a natural language task). This difference signifies that in the deep learning context conditioning adds both substantial dimensionality as well as complication to the latent space, in the effort to facilitate class-dependent generation.

1.3.7 Conditional Autoencoders

Other work in Variational Autoencoders (Kingma and Welling, 2014) accounting for the conditioning of multi-modal data has demonstrated the success of structuring the data and latent space (Mishra et al., 2018). These models structure both inputs to

the encoding function $Enc(x|y)$ as well as the decoding function $Dec(z|y)$, shown in Figure 1.8 as an injection of conditions into both functions of the autoencoder. Thesis thesis explores two modifications to this architecture by removing the need for the conditioned encoder and inducing an adversarial loss on the model, discussed in Section 2.1.

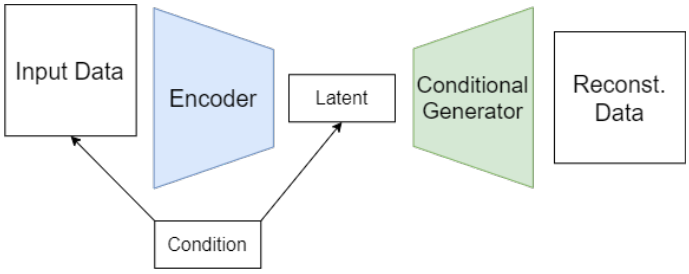


FIGURE 1.8: Conditional Autoencoders condition during encoding and decoding

The addition of conditioning to a model allows function components to incorporate distinct information on multi-modal data, add degrees of freedom, and tune by labelled factors of variation. Conditioning alters models $f(x)$ to account for the conditional probability of the sample $p(x|c)$, learned by feedback on evaluation of $f(x|c)$. This departure from the typical autoencoder affords benefits in the interpretation of the model as both the data space and latent space are partially structured: the encoding and decoding models see some proportion of the domain as a condition which informs the subsequent representations.

Conditioning the decoder in particular begets many advantages, particularly when paired with latent structuring such as in the VAE: using this design, the latent space can be sampled at will and conditional generated samples may be produced. This remarkable benefit, discussed in Section 1.3.8 and 2.1, extends representation learning to training data approximation.

1.3.8 Conditional GANs

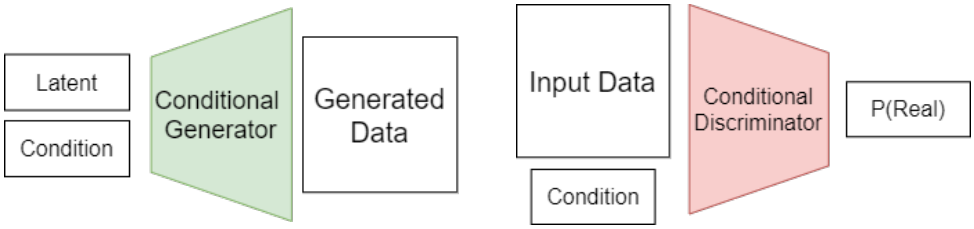
Training data approximation is the task of creating not only estimations of the training distribution $p(x)$ with $p(\hat{x}|z)$, but estimations of the joint training distributions given by $p(x|c,z)$, where $c \in C$ may be the labels of interest to a downstream supervised model.

The conditional GAN departs from the standard model by concatenating additional class information c to inputs to the generator $G(z,c)$ and discriminator $D(x,c)$. This alteration gives both models inference to relate the condition to the expected content of data space samples. For the discriminator, this is directly learned from seeing (x,c) for training samples x and corresponding conditioning labels c. For the generator, the relationship between c and the desired output $x|c$ must be inferred from losses returned by D.

Though the collapse of the generator-decoder architecture has been proposed and is demonstrably useful, the distinction between the two remains interesting. The generator has no control of the distribution which is must decode to the data space

In a GAN, The discriminator can use this information to discern in the training set how each condition should relate to the properties of the samples, and the generator then gets feedback on how the "seeding" condition applied to random normal

samples should correspond to generated outputs. The generator models the *conditional* probability $p(x|z,c)$ where c is a sample from the class distribution $p(c)$. Likewise, the discriminator evaluates $p(x|c)$, the conditional probability that the sample is drawn from the training distribution under the class condition c.

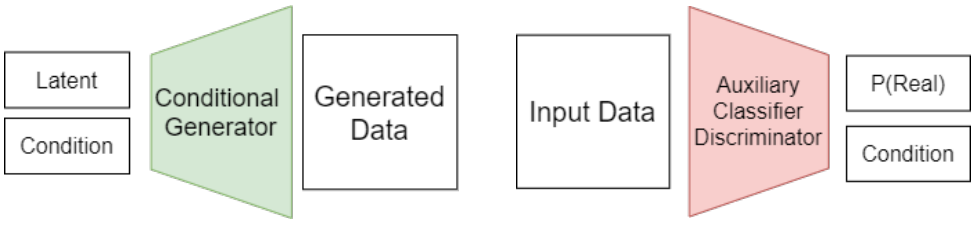


(A) Conditional generator. (B) Conditional discriminator.

FIGURE 1.9: Conditional GAN architecture.

1.3.9 Auxiliary Classifier GAN

The initial formulation of GANs by Goodfellow et al., 2014 laid the foundation for the broad class of game-theoretic generative models which were both cheap to sample and made minimal assumptions on the image synthesis distribution. However the basic implementation has success constrained to low-variable, low-resolution task (Radford, Metz, and Chintala, 2015). The auxiliary classifier GAN (ACGAN) demonstrated that the structuring of the latent space produced higher-quality samples, particularly when moving to higher-resolution tasks in natural images (Odena, Olah, and Shlens, 2017). This represented a significant step in improving the discriminability and diversity of samples, as the invariances and structures of natural images were a large barrier for early GANs to overcome. As in this thesis, the ACGAN generator demonstrated success on its own merits, without the GAN training techniques which can drastically alter model behavior (Salimans et al., 2016). The ability to maintain image structure and invariances across distant regions of a generated image demonstrate a significant improvement when generative models access a conditioning element which partitions the latent space in disjoint regions.



(A) Conditional generator. (B) Auxiliary Classifier discriminator.

FIGURE 1.10: Auxiliary Classifier GAN architecture.

This model mimics the CGAN with the exception that the discriminator does not get the condition appended to the input. Instead, the discriminator must predict using the nominal classifier split as a classification head on the penultimate layer of the discriminator. When training this model, the discriminator coefficients are updated with respect to both the classification and adversarial errors, effectively training a two-task model. Thesis thesis uses an extension of this encoding philosophy in Section 2.1 which extends image-wise quality evaluation learned by the discriminator to image-wise representation encoding seen in an autoencoder.

1.4 Model Architecture

Each of the established and proposed models of this thesis could operate with one of countlessly many possible architectures: inclusion of batch normalization, activations, depth, width, and network learning rate are some of many designer choices a neural network engineer must grapple with when designing any model. For the purposes of this thesis, a heuristic middle ground must be chosen in order to compare a diverse set of algorithms over a various experiments. As each encoder, decoder, generator, discriminator, and predictive model ultimately serves one of two general roles ("data to latent" or "latent to data"), two primary architectures are used as a starting point before necessary tweaks are made to accommodate the differences between algorithm design. Encoders and discriminators for all models share generally identical architecture with the exception of small input and outputs to account for unique training algorithms. Generators and decoders generally share identical architecture with the same exceptions for input and output type.

Explicit design for architectures shared by all models discussed in this thesis are discussed at length in Appendix Section A.2 with corresponding diagrams, activations, motivations, and layer descriptions.

1.4.1 Hyperparameter Selection

One manner to control the instability of GAN training is to select a heuristic hyperparameter for use in Section 3. Fortunately once the model architectures have been established, many "rule of thumb" hyperparameters have demonstrably good performance in GAN training. The learning rate which determines the magnitude of the gradient updates at each training step can be safely set to $2 * 10^{-4}$ for many tasks considering the number of parameters and batch size (Mescheder, Geiger, and Nowozin, 2018). Training time for GANs on the CIFAR-10 task determined by implementations of Karras et al., 2020a determine a batch size of 8 to be admissible as it begets a number of training updates corresponding to the number of training epochs, chosen to be 10 for the results of Chapter 3. Quantitative evaluation of trained generators confirms that increases to quantitative performance are extremely slow after the 10-epoch range due to limited model capacity of the model architectures discussed in this thesis.

Chapter 2

Architecture Contributions

2.1 Autoencoder-GAN

This thesis proposes the Autoencoder-GAN (AEGAN) both as an alternative architecture to the GAN and CGAN for the purposes of generative modelling, and an alternative to autoencoders for representation learning. This alteration comes from the observation that both the decoder of an autoencoder and generator of a GAN both map from a low-dimensional latent space to a data space. In the autoencoding paradigm, both the encoder and decoder components contribute to learning the unstructured distribution of the latent space as the loss given by reconstruction is back-propogated to the encoder, which controls the training sample encodings. For the GAN generator, this distribution is out of its control: the distribution is a prior imparted which the model must approximate the corresponding transformation to the training data distribution. Usually stipulated to be a multivariate standard normal distribution, the generator must learn to assign meaning to the dimensions of this space such that the generated samples contain enough variability and structure to minimize the loss returned by the discriminator.

The AEGAN and the conditional extension CAEGAN combine the explicit representation learning of an autoencoder with the semantic decoding of random samples of a latent space performed by a generator. This process weakly structures the variation space without the need for an explicit prior as in the VAE (1.3.2). Equation 2.1 gives the triple-criterion optimized by the CAEGAN:

$$\mathcal{L} = \mathcal{L}_{cond} + \mathcal{L}_{pixel} + \mathcal{L}_{GAN} \tag{2.1}$$

Two of the loss elements are directly borrowed from the nominal models: the adversarial GAN loss $\mathcal{L}_{GAN}$ (Equation 1.4) and the autoencoders' pixel-wise reconstruction loss $\mathcal{L}_{pixel}$ (Equation 1.1, 1.2). The $\mathcal{L}_{cond}$ loss is a departure from the typical conditional autoencoder, given by a supervised loss (binary cross-entropy for one-hot labels, Equation 1.2) from the encoder's prediction of the input sample label, meaning the model must *predict* the corresponding conditions of the input sample rather than be provided it as an argument. This predicted label is used to condition the decoding-generative portion of the model as well, meaning the model receives a loss on the encoding step from all three components of the optimization Equation 2.1. The autoencoder-GAN collapses the GAN generator and autoencoder decoder into a shared model, where gradients are summed and back-propogated through the entire model including the encoder. This means that generative and reconstructive tasks must be managed by the same set of coefficients, leveraging our prior that the manifold assumption ensures the training distribution and the learned approximation share a latent representation.

The inclusion of conditioning is a vital part of the design on this model. The third element $\mathcal{L}_{cond}$ represents a structural prior imparted on the condition space, one of the two latent spaces. This inductive bias is implemented depending on the task as a predictive layer into the conditional latent space representing a class label which may be double as a target of interest for a classification task. This is what defines the Conditional Autoencoder GAN (CAEGAN). This design allows the encoder to map any information into the unstructured variation space V, but must learn to map input samples x with corresponding condition c into a latent point $(\hat{v}, \hat{c})$. This process dramatically differs from the conditional autoencoder (1.3.7), which conditions both the encoding and decoding functions. While the variation space is totally unstructured for this thesis, it could have additional structure such as in the variational autoencoder. The condition-space C however is highly structured by a supervised loss (Binary Cross-Entropy int he disjoint one-hot condition space given by Equation 1.2). This enforces that up to some penalty, input samples x must be encoded to predict their class in the C-space and the learn inter-class factors of variation in the V space. Increasingly strict assumptions about the nature of the structure of C are explored in Section 3.1.2.

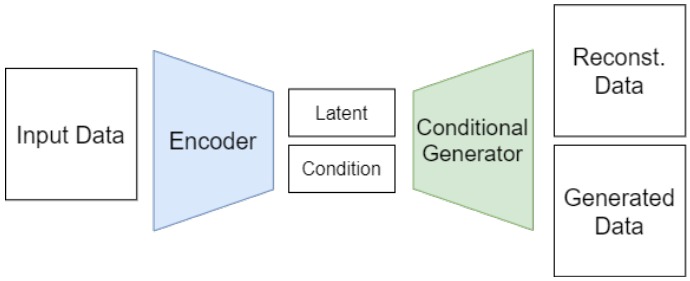


FIGURE 2.1: Hidden Conditional Autoencoder-GAN Architecture.

A proposed benefit of this architecture is the loss-disincentive of the modal collapse GAN and CGAN are prone to. These models often rest in shallow local minima in which the discriminator loss quickly reaches 0 before the generator can roughly approximate the training distribution. Another training failure mode is when the generator learns to nearly replicate one or a small subset of training samples, and periodically moves between these modal distributions with the discriminator "chasing" the updates. By enforcing a reconstruction loss on the shared decoder-generator, the generative component of the model does not rely solely on feedback from the discriminator and instead updates according to the non-adversarial reconstructive task, which grounds the learned coefficients to produce data-space samples: not just fool the discriminator. This also allows the model to escape loss-collapses where 0 adversarial loss would end training for a generator relying on the discriminator alone by nullifying gradients. The proposal that additional loss terms regularize the side effects of adversarial training will be a common theme in this thesis.

In order to prove the efficacy of the CAEGAN, it must be demonstrated that either the encoding portion of the model performs the classification task better than a predictive model with only the classification loss and identical architecture performs the task, or that the decoding portion of the model likewise outperforms an identical decoder (CGAN) which is informed solely by the adversarial loss. This thesis hypothesizes that for all proposed models, the incorporation of additional loss terms regularizes each component, lessens reliance on adversarial losses which can be unstable or move the parameters into undesirable regions, and incorporate indirect information on the task goals that simple models are not privy to; for example

feedback from decoder reconstruction influencing how the encoder maps samples into the condition-variance space.

2.2 Inverse Autoencoder-GAN

The Inverse Autoencoder GAN is the second proposed alternative architecture of this thesis. The nominal inversion performed by this model comes from two sources of inspiration: the CycleGAN (Zhu et al., 2017) and contemporary research in latent representation learning (Donahue, Krähenbühl, and Darrell, 2016). Recovering latent representations of data is an ongoing and highly researched topic in machine learning. Methods such as contrastive learning (Dai and Lin, 2017) and BiGAN (Donahue, Krähenbühl, and Darrell, 2016) emphasize the importance of recovering the sampled latent code which led to the generation of an image. This inverse mapping with demonstrable semantic meaning in a dense space is not provided by a typical GAN - though it is known to be useful for auxiliary supervised feature learning. A non-conditional formulation of the inverse autoencoder-GAN (IAEGAN), called the latent regressor model, is proposed in Donahue, Krähenbühl, and Darrell, 2016 as was shown to under perform the BiGAN model. The authors hypothesize that the latent regressor model struggles to recover latent representations of complex natural images, though this thesis argues that conditioning for modalities may ameliorate that challenge.

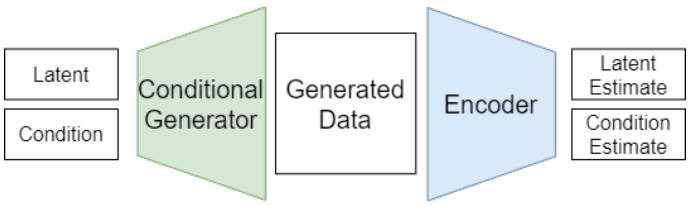


FIGURE 2.2: Inverse Conditional Autoencoder-GAN Architecture.

An inverse autoencoder as proposed by this thesis is identical to a conditional GAN - the only departure is the addition of an encoder network borrowed from the autoencoder family. This alteration modifies the otherwise normal GAN to include a reconstructive loss. Whereas reconstructive losses are usually taken on the data space samples passed though a bottleneck autoencoder, this reconstruction loss is taken between the latent sample v and the reconstruction $\hat{v}$. The inverse autoencoder-GAN utilizes the same loss formulation as the CAEGAN (2.1), though with very different interpretations. The adversarial model is described as:

$$\mathcal{L} = \mathcal{L}_{cond} + \mathcal{L}_{v\ recon} + \mathcal{L}_{GAN} \tag{2.2}$$

In the ICAEGAN model, the reconstructive loss $\mathcal{L}_{recon}$ is given not in the data space for training sample reconstruction, but rather in the latent variation space V. Like a typical GAN, the variation-space is sampled (multivariate standard normal), and decoded by the standard conditional decoding architecture. Continuing the similarity to a GAN, this decoded random latent sample is evaluated by a discriminator, and each model receives an adversarial loss $\mathcal{L}_{GAN}$ given by 1.4. However, this thesis insists on cycles induced by reconstructions of encoded or decoded points, so this generated sample is now returned the the condition-variance space by the encoder, which defines the reconstructive and predictive losses $\mathcal{L}_{cond}, \mathcal{L}_{recon}$. The conditioning

appearing in the inputs to the generator and outputs to the encoder alter the inverse autoencoder GAN to be a inverse conditional autoencoder-GAN (ICAEGAN).

The design of this inverted autoencoder was inspired by the CycleGAN paradigm (Zhu et al., 2017). In CycleGAN, there exist two unpaired image sets drawn from distinct training distributions X and Y. Effectively two GANs are trained, one which learns $F(x) = y$ dictated by a discriminator trained to distinguish real and fake samples in x, and another mapper who learns $G(y) = x$ taught by a discriminator who learns to distinguish real and fake samples in Y-space. The return of the CycleGAN design to a single-distribution problem is inspired by the observation that a GAN learns the unpaired relationship between some prior sampling distribution V and a target distribution X. In the same manner as CycleGAN, this mapping must be bijective and invertible to minimize the corresponding loss function which requires not only high-quality images in each domain, but a cyclical loss in which samples must maintain the reconstruction $F(G(y)) \approx y$ and $G(F(x)) \approx x$ under cycles through the unpaired space.

The hypotheses of the implications of these additional losses are as follows. The reconstruction and conditional prediction of generated samples encourages the model to maintain a conditional manifold $P(x|v, c)$ of dimension weakly bounded below by the dimension of the latent variation space V. This is because when the model ultimately re-encodes the sample to the variation space, if the learned conditional data space manifold is substantially lower dimension than V, the reconstruction of the latent code will be impossible as the co-domain dimension is bounded above by the domain dimension for linear functions f.

In addition, one experiment and practical usage of conditional generation is for supervised data augmentation (Section 3.2). By training a model capable of re-encoding its own generated samples, it may be possible to improve classification accuracy and latent-representation reconstruction by including these goals at training time, as the CAEGAN is not optimized to re-encode its own samples for class and latent prediction - though it does indirectly perform this task without supervision.

2.3 Cycle Autoencoder-GAN

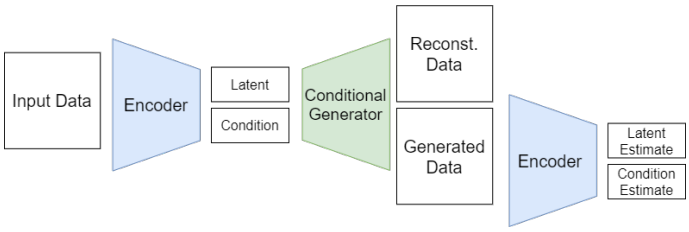


FIGURE 2.3: Cycle Conditional Autoencoder-GAN Architecture.

The cycle autoencoder-GAN is the logical extension of the CAEGAN and ICAEGAN. Where the CAEGAN and ICAEGAN perform reconstruction of a space under the bottleneck of another space (data-latent-data and latent-data-latent, respectively), the cycle autoencoder-GAN does both. This leads to a four part loss, given by:

$$\mathcal{L} = \mathcal{L}_{cond} + \mathcal{L}_{pixel} + \mathcal{L}_{v\ recon} + \mathcal{L}_{GAN} \tag{2.3}$$

Where the $\mathcal{L}_{cond}$ comes from two sources: supervised learning of training data labels, and reconstructing the random samples of C-space by generating and re-encoding latent samples:

$$\mathcal{L}_{cond} = \mathcal{L}_{c|x} + \mathcal{L}_{c|Dec(v,c)} \tag{2.4}$$

This model extends the goals of the CAEGAN and ICAEGAN by performing both tasks: autoencoding training data to learn a meaningful latent representation of the training distribution, as well as performing reconstruction of generated samples as inspired by BiGAN (Donahue, Krähenbühl, and Darrell, 2016). This design lends itself to inflated losses: only a small component of the overall magnitude of the gradients comes from the adversarial loss itself, this model relies heavily on the assumptions of CycleGAN between the unpaired prior space and the data space (Zhu et al., 2017), where the discriminating adversary on the prior space (in this case, multivariate standard normal samples) has been replaced with a supervised loss because of the special case where we know the structured distribution over (v, c).

The cycle autoencoder-GAN incorporates conditioning in the same manner as the CAEGAN and ICAEGAN, forming the CCAEGAN. This model serves a special purpose for the sake of experimental integrity in this thesis. One valid argument for why the CAEGAN may perform well, as analyzed in Section 3.1, because it has the unfair advantage of reconstructing training samples. This could lead to the model performing well on quantitative metrics as a model which "generates" training samples exactly will receive good scores without having truly generated a new sample which does not already exist in the training set. The ICAEGAN and CCAEGAN are important goalposts against which the performance of the CAEGAN must be compared: the ICAEGAN only experiences training labels in the same manner as a GAN: indirectly through the lens of the discriminator's feedback. The CCAEGAN, however, does get to directly autoencode training samples: it exists as an in-between model to the models it extends upon, with surprising results in Chapter 3.

Chapter 3

Results

3.1 Quantifying Generative Quality

This section evaluates the generative quality of the proposed and baseline models. Natural image data sets are a common benchmark which allow generative models to be analyzed, and representations learned on one natural image data set are often generalizable for transfer across vision tasks (Pan and Yang, 2009, Touvron et al., 2021, Foret et al., 2020). For this reason this section uses the CIFAR-10 dataset, a collection of $60,000$ natural images evenly distributed across 10 classes of $32x32$ pixels in three RGB channels (Krizhevsky, Hinton, et al., 2009). There exists a pre-defined train-test split, which is used to evaluate supervised learners on the unseen partition as well as compare distributional distance between generated samples and unseen testing examples to evaluate generative quality. It is important to report both the quality of generated data distributions and the adherence of the conditional samples to their conditional distribution to measure both the quality of the learned generator, as well as its adherence to inter-class similarity and intra-class differences. During GAN training, this is often done by inspection - the designer can review a swathe of samples by eye, and judge the quality (Mathiasen and Hvilshøj, 2020). The qualitative eye-test metric is extremely useful for implementation and rapid feedback, but is not sufficient for accurately determining high-quality generation.

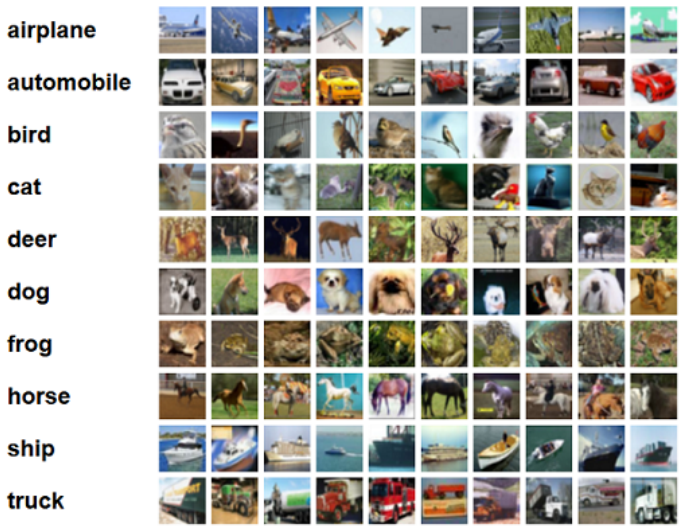


FIGURE 3.1: CIFAR-10 is a benchmark natural image data set. Image from Krizhevsky, Hinton, et al., 2009.

In a competitive setting, there exist multiple compute-expensive options to quantify GAN performance after training is completed, though there is presently research

for more rapid quality reporting (Mathiasen and Hvilshøj, 2020). The Fréchet Inception Distance (FID Equation 3.1), proposed by Heusel et al., 2017, quantifies the quality of a generated distribution with respect to a target distribution by encoding each in a learned representation of the Inception-v3 model, which was trained to be a record-setting ImageNet classifier but has had its' usage expanded to be a more general workhorse of natural image representations (Szegedy et al., 2015). Because of Inception-v3's powerful representation-learning layers, it doubles as a robust image-content embedder by utilizing the penultimate representation activation values before the classification layer. Though embeddings of samples into the massive model are expensive, the FID is the standard measure for competitive generative models (Karras et al., 2020b, DeVries et al., 2019, Zhang et al., 2019).

For the purposes of calculating the FID, one generated data set and one holdout data set of real images are embedded into a representation-space by the Inception model. This embedding yields the real-data representation distribution given by the encoded CIFAR-10 testing samples, as well as a distribution of embedded generated points made by the decoding of random multivariate standard normal samples. The Wasserstein-2 (Korotin et al., 2020) distance then returns the FID score as the distance between the testing distribution and generated distribution expressed as normal distributions of maximum likelihood mean and variance in the representation layer shown in 3.1, where Tr indicates the trace operator, μ_r, μ_g indicates the mean of the embedded real and generated distributions, and C_r, C_g are the covariance matrices of the embedded real and generated distributions (Heusel et al., 2017):

$$d^2 = ||\mu_r - \mu_g||^2 + Tr(C_r + C_g - 2\sqrt{C_r * C_g}) \tag{3.1}$$

Lower FID scores are better, as an ideal generator would minimize the distance between the generated distribution and target distribution represented in Inception-v3's layers. Significantly, the Fréchet Inception Distance does not account for the joint distribution of samples and classes for conditional data: one must either partition the testing data set and compute one FID for each discrete class, or turn to new research on the specific evaluation of conditional generative models. However, using this approach makes for substantially slower evaluation, and in the case of CIFAR-10, splitting the testing set into 10 classes yields only 1000 images per class - a sparse sample of the high-dimensional and complex natural image manifold. In addition, FID estimation of the conditional generation quality only holds for discrete conditions, as in CIFAR-10: If the conditioning space were continuous this approach would be intractable without discretization as the FID requires embedded *distributions*, not points (DeVries et al., 2019).

The Fréchet Joint Distance (FJD) proposed by DeVries et al., 2019 accounts for *joint* distributions of images and conditions to express generated sample quality, adherence to the intended class, and diversity from other classes. The FJD accounts for the joint image-condition distribution in the embedded space space when calculating the distance between maximum likelihood Gaussian estimations. FJD further improves upon the FID for evaluating the conditional quality of generated images by extending discrete classes to continuous, a task necessary for applications in Chapter 4 and one which is intractable for the FID.

For the results in Table 3.1, each of the models uses identical architecture for encoding and decoding portions as discussed in 1.4. Additionally, the experiments were performed ten times for each model, accounting for the variability in GAN training and hedging against potential collapse. The variability in the results of these experiments, which is a significant metric for generative quality, is recorded

in the standard error reported for each experiment. For each trial, the model was trained with the same hyper parameters and at training conclusion of ten epochs, the generator was passed to the metric evaluator which returns the FID and FJD of the generated samples compared to the designated testing set of CIFAR-10. The discriminators and hyper-parameters are all held constant across all trials and architectures, determined *a priori* by typical GAN heuristics (Larsen et al., 2016, Salimans et al., 2016). Additional GAN heuristic modifications and training algorithm alterations have not been explored here, including: multi-step updates (Salimans et al., 2016), discriminator loss blocking (Zadorozhnyy, Cheng, and Ye, 2020), relativistic discrimination (Jolicoeur-Martineau, 2018), and augmentations (Karras et al., 2020b), though augmentations are discussed for applications in Section 4.2.

The ACGAN does not use weight tuning on the supervised and adversarial losses, and uses unweighted contributions. The cycle models we propose (CAEGAN, ICAEGAN, and CCAEGAN) also lack tuning on relative weighting between conditional, reconstruction, and adversarial losses - all are unweighted despite using different loss functions. However we see cyclical models CAEGAN, ICAEGAN, CCAEGAN all outperform strictly generative models GAN, CGAN, ACGAN in FJD. The discrepancy between the baseline and cycle models cannot be explained only by the autoencoders generating samples based on leaking reconstruction of the training distribution as foretold in Section 2.3, as the ICAEGAN does not perform training sample reconstruction. Rather, generative performance is improved when the learned weights are updated at least in part by a non-adversarial task.

	FID	FJD	Difference
Training Distribution	3.07	33.47	30.4
GAN	33.09 ± 0.31	56.48 ± 0.25	23.39
CGAN	44.37 ± 0.54	51.97 ± 0.59	7.60
ACGAN	43.91 ± 0.31	61.59 ± 0.62	17.57
CAEGAN (ours)	37.67 ± 0.37	46.42 ± 0.47	9.05
ICAEGAN (ours)	39.06 ± 0.25	48.16 ± 0.27	9.10
CCAEGAN (ours)	$\mathbf{35.72} \pm 0.27$	$\mathbf{43.22} \pm 0.32$	**7.50**

TABLE 3.1: Fréchet distances across model architectures. Lower is better.

While FJD captures image quality, diversity, and class adherence, FID only measures quality of generated images with respect to the training distribution. In Table 3.1, "Training Distribution" demonstrates what a floor on the FID would look like: by sampling random unconditional training observations and comparing the distance under the Inception embedding to testing samples, a Wasserstein-2 (Korotin et al., 2020) distance of around 3 is expected. However, since this random sampling does not account for the joint distribution over classes, we see the FJD is substantially higher. The difference between FID and FJD is not an established metric, but is useful to quantify how well the model is balancing the quality and conditional components of the FJD score relative to the FID score. It is demonstrated here that to minimize conditional quality score, a model could to some extent get away with ignoring the condition as long as the image quality is sufficient, though this exploitation of the score would fail in metrics such as 3.5 which require class consistency more explicitly.

The unmodified GAN outperforms in image quality alone, in part because it is trained without regard for the conditional task, while the CCAEGAN demonstrates success in both metrics, scoring as the lowest-FID conditional model as well as the single-best FJD model. The GAN has not been bolded as it serves as a demonstration of the kind of generative quality one can expect. The training distribution demonstrates that high-quality images, ignoring conditioning, still lead to good FJD scores from nearness to the training manifold under the Inception-v3 representation. The CCAEGAN additionally has the lowest difference between its FID and FJD performance, demonstrating the strictest adherence to inter- and intra-class differences in content, though this difference is not an established metric.

It is worth noting that the ICAEGAN and CGAN are nearly identical models, down to hyperparameters, architectures, and training algorithms. The only difference is that the ICAEGAN "re-encodes" its own generated samples, and receives a supervised loss from this self-supervised operation. However, when quantifying the quality of the generated distribution, the ICAEGAN actually outperforms the CGAN to a substantial degree. Ignoring the other models and focusing on these two paints a sharp image as cyclical losses being a substantial benefit to the training of generative models. There is no training data leakage, increased model capacity, or other confounding factors beyond the possible regularization or disentangled representations being imparted on the generative component by the encoder. This is one example of a compelling argument for the inclusion of cycles as a form of regularization, or perhaps an advantage of disentangled learning which requires further study.

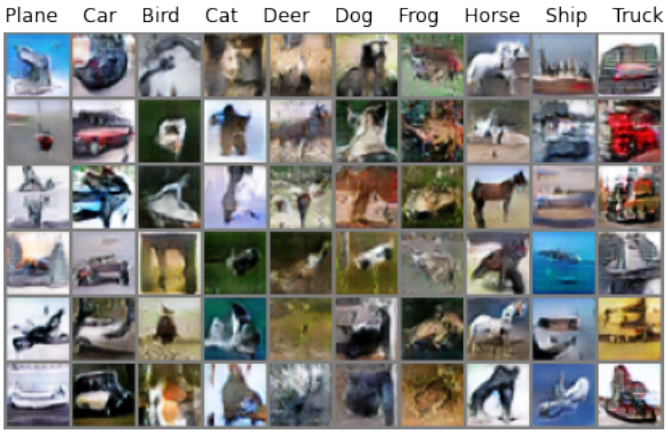


FIGURE 3.2: Sample CCAEGAN conditional generation of image classes

Figure 3.2 provides a sample of what a trained conditional image generation generator creates. By taking random samples of the variation space and combining with specified type tuples, visual inspection shows that the generator somewhat adheres to inter-class and intra-class variation, though the image quality is not high.

3.1.1 Low-Data Regime

Often, real data sets have few observations. This can further exacerbate the curse of dimensionality or sparseness of the high-dimensional data space of natural images. The challenges posed by GANs such as modal collapse and adversarial divergence are made worse when overfitting to the training data is an easier task due to the reduced problem difficulty (typically for the discriminator). Additionally, it has

been found that even the full-data CIFAR-10 task can be considered a limited data benchmark: Karras et al., 2020b demonstrate how augmentation of even the full the training set can substantially increase generative quality, implying that the training set alone is not a rich-enough distribution for optimal generation. In this section we investigate how the performances reported in Table 3.1 change with a substantially smaller data set, with one-quarter the training dataset of CIFAR-10 (12500 images) in order to evaluate potential collapses or differences more apparent in the model designs for this realistic problem.

	FID	FJD	Difference
GAN	67.70 ± 0.54	87.19 ± 0.47	19.49
CGAN	79.70 ± 0.83	90.56 ± 0.90	10.86
ACGAN	$\mathbf{71.02} \pm 0.40$	88.58 ± 0.41	17.56
CAEGAN	$\mathbf{71.77} \pm 0.67$	$\mathbf{83.00} \pm 0.68$	11.23
ICAEGAN	86.04 ± 1.04	99.01 ± 1.19	12.93
CCAEGAN	$\mathbf{73.00} \pm 0.80$	$\mathbf{84.11} \pm 0.85$	11.11

TABLE 3.2: Small-data Fréchet distances across model architectures. Lower is better. The GAN has not been bolded as it serves as a demonstration of the kind of generative quality one can expect, but fulfills a different task to the other models.

In the low-data paradigm, the previous reasonable success of the ICAEGAN has been totally diminished, falling dramatically behind the competition. While the GAN continues to outperform in unconditional generation, the CAEGAN beats the CCAEGAN in conditional generation measured by the mean FJD, though the standard error indicates that this is a uncertain claim. It can be assumed that if regularization limiting the plausible coefficient space on the generative models contributed to their success in the full-data task, that the same effect is even more apparent when the training data is substantially reduced. The ACGAN has substantially lower standard error estimates than the rival models, indicating consistency even in the low-data paradigm, while the other models have substantially increased variability in generated quality. The autoencoding-cycle models CAEGAN and CCAEGAN do outperform the competition with the lowest mean FJD scores, however, likely assisted by the reconstruction error gradient weights maintaining the decoder's proximity to the training distribution.

	FID Gap	FJD Gap
GAN	34.61	30.71
CGAN	35.33	38.59
ACGAN	27.11	26.99
CAEGAN	34.10	36.58
ICAEGAN	36.98	50.85
CCAEGAN	37.72	40.89

TABLE 3.3: Differences between quarter-data and small-data Fréchet distances across model architectures. Lower is better.

The change in performance is shown in Table 3.3, in which the differences between the full-data and low-data paradigms are quantified. The ACGAN loses less

ground than its competition when three quarters of the training data have been removed. It is possible that the more difficult discriminator task of simultaneous quality-evaluation and class-prediction regularizes the discriminator against overfitting to the small training set. Despite losing less ground in performance than the competitors, the ACGAN fails to outperform the cyclical models with regard for FJD.

3.1.2 Snapping

The interpretation of the CAEGAN and CCAEGAN latent conditions under the initial design was initially a non-probabilistic encoding over the classes by using the sigmoid activation into the latent code space. This means that the model can assign some vector $c \in \mathbb{R}^{10}$ over C for each sample, according to the loss taken between the encoding and the 10-class label in the case of CIFAR-10. In this paradigm, the model can incorporate some nonzero amount of information about the samples into the condition space which is effectively a second variation space despite a penalty. However, it could be beneficial to use the inductive assumption that the C space is a multi-class classification task, for which we use the softmax (Equation A.1) of the output of the encoder into c space for tasks known to have disjoint labels, such as CIFAR-10. Furthermore, to prevent the model from using the condition space to include information beyond only the class label, a differentiable rounding function can be applied to the encoded condition-space samples $Enc_c(x)$ before the decoding step, but after taking the condition-encoding loss, $c = max(softmax(Enc_c(x)))$. Taking the max of the softmax of the encodings is an argmax function: the most likely class is chosen as the effectively one-hot encoded class. Rounding in this way means the model decodes samples according to the same specifications as the generation task, in which only one-hot labels are used as conditions - not distributions over labels, as in the softmax and sigmoid activations. No additional information may be encoded into the condition space using this design, which is in line with the assumptions of the bipartite split.

Table 3.4 catalogs the FID and FJD of the proposed changes to the CAEGAN architecture including an alternate classification activation and force-sparsity Snap layer, though no effect is noticeable. It is likely that the Snap model has superior characteristics for analysis of the latent space z, since all non-class variation must be encoded there. This begets the double benefit of making the decoding and generation tasks more similar, as well as assures that the variation between samples of a constant class exist only in the variation space.

	FID	FJD	Difference
CAEGAN (Sigmoid)	37.67 ± 0.37	$46.42 \pm .47$	9.05
CAEGAN (Softmax)	36.61 ± 0.40	45.00 ± 0.47	8.39
CAEGAN (Snap)	37.27 ± 0.40	45.70 ± 0.35	8.43

TABLE 3.4: Proposed alterations to the autoencoding model inductive bias on the conditional encodings. Lower scores are better.

Table 3.4 demonstrates that the selection of inductive bias on the condition space matters very little. The benefits of the softmax activation match the assumptions of the data labels and naturally performs the best, though the flexibility of the sigmoid

is reassuring for extensions to intersecting label tasks. Though the Snapping algorithm is a strong assumption to structure the condition space, it has little effect on the Fréchet distances as the generative task of the model is unaffected.

3.1.3 Generative Diagnostics

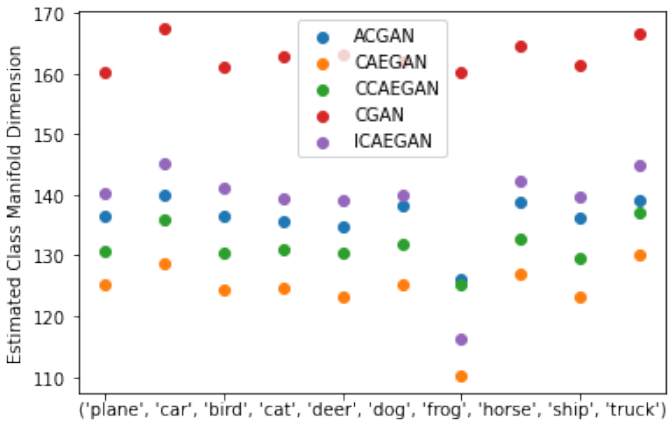


FIGURE 3.3: Higher estimated manifold dimensions indicate more variation in generated classes.

Figure 3.3 gives a sense for the complexity of the generated samples on a per-class basis. In this demonstration, each trained model produces 2000 samples per class condition (x-axis). The dimension of each of these batches is then estimated with PCA, where the estimated dimension is given by the number of dimensions formed of linear combinations of the data space dimensions have a cumulative explained variance ratio above 95%. Figure 3.3 implies that the number of dimensions needed to explain 95% of the variation in each class generated by the CGAN is significantly higher than the other models. In addition, it is remarkable that the CAEGAN and CCAEGAN, which outperformed the competition in the FJD metric, have substantially lower variability in their class samples. This is related to the low FJD scores reported on each model, as inter-class similarity is an important contribution to the score. It is possible that studying the linear dimension estimation is capturing variability in a negative sense: the CGAN and ICAEGAN are producing more "complex" classes by blurring the lines between classes. The CAEGAN and CCAEGAN, however, are adhering more strictly in inter-class similarity which minimizes FJD but may lead to a lack of interesting diversity which demonstrates true understanding of the class manifolds.

3.2 Augmenting Training Data

One proposed benefit of training data approximation by conditional generation is augmenting the data set with generated samples to improve a downstream predictive model. Conditional generators have the advantage of producing estimated labelled data tuples $G(z|y) = (x, y)$, of which generation is nearly free: once the model is trained, sampling of the latent space, generating, and saving samples is facile, and could be done as many times as desired regardless of the size of the training data. Exploiting indefinite free sampling of random normal distributions, we can augment the dataset with any amount of labelled samples, where the supervised labels

are given by the class which conditioned the generator. Assuming the task at hand for the generator was to minimize the FJD to the testing distribution through exposure to the training distribution, these samples should be beneficial to the supervised learner to augment the training set.

Table 3.5 reports the down-stream test accuracy of a multi-classification neural network according to a variety of augmentation strategies. In each cell, ten trials are performed in which a baseline supervised learner with architecture shown in A.2.1 is trained using a training set which is the indicated percentage of the CIFAR-10 training set. The remainder of the 50,000 images are filled in with generated labelled samples given by the row name. This means that for the 75% column, the first 37500 CIFAR-10 samples are chosen from a pre-determined shuffle of the training data constant for each experiment, and 12500 samples are generated and concatenated to the dataset. Each generative model is compared against a lack of augmentation, the "None" row. In the "None" row there is no additional augmentation performed - only the indicated percentage of the training data is used. In each case, the training data is stratified-sampled down to the indicated percentage. In each case the testing set remains a holdout set of 10,000 samples uniformly distributed across the 10 classes. The top-1 accuracy is the reported metric on this unseen testing set.

It is important to note that the experimental design used here is not identical to how augmentation would be used to optimize a model for strictly testing accuracy. Usually some kind of validation hyperparameter search would be used to find an augmentation loss weight ω which would contribute to the model loss:

$$\mathcal{L} = \mathcal{L}_{Train} + \omega \mathcal{L}_{Augment} \tag{3.2}$$

The augmentation weight is a concession that generated samples can only approximate the training distribution, and may be harmful if they contribute equal weight as the real samples in model optimization. Results in section 4.2 demonstrate this effect. Results in Table 3.5 demonstrate how using $\omega = 1$ contributes to model testing performance for a variety of ratios of training data to GAN-augmented data.

	75% Real	25% Real	10% Real	5% Real	0% Real
None	69.87 ± 0.23	59.12 ± 0.06	$\mathbf{47.09} \pm 0.07$	$\mathbf{32.60} \pm .12$	10.11 ± 0.04
GAN	69.12 ± 0.05	49.62 ± 0.09	26.23 ± 0.16	12.09 ± 0.13	10.47 ± 0.11
CGAN	69.89 ± 0.05	$\mathbf{64.06} \pm 0.38$	34.13 ± 0.16	24.91 ± 0.09	18.55 ± 0.12
ACGAN	69.45 ± 0.04	44.72 ± 0.20	27.93 ± 0.17	21.24 ± 0.17	16.13 ± 0.15
CAEGAN	69.45 ± 0.04	51.18 ± 0.18	34.45 ± 0.19	25.89 ± 0.15	19.45 ± 0.16
ICAEGAN	69.44 ± 0.05	49.61 ± 0.14	32.78 ± 0.17	25.42 ± 0.13	19.25 ± 0.15
CCAEGAN	$\mathbf{70.00} \pm 0.05$	52.41 ± 0.19	36.49 ± 0.21	26.46 ± 0.10	$\mathbf{19.95} \pm 0.10$

TABLE 3.5: Test accuracy when training data is augmented with fake data. Higher is better.

It is apparent in Table 3.5 that the design to back-fill the missing component of CIFAR-10 with generated samples is likely not the correct direction if the explicit task is downstream test accuracy. Rather, this table demonstrates the model's capability to inform a downstream supervised learner. Investigating the first column, it is clear from the standard error of the testing accuracy that the addition of 25% generated data (the 75% Real column) actually serves as a productive regularizer for the supervised model by considering the standard error of the testing accuracy,

albeit with a very slight decrease in average accuracy for the majority of the generative models, which is a worthy trade-off for some applications. For the purposes of dataset augmentation for the explicit task of downstream testing accuracy with a high augmented-sample weight loss of $\omega = 1$, Table 3.5 implies that an augmentation to real data ratio of between $1:3$ and $3:1$ is appropriate. This is under the assumption that the fake samples are weighted by the supervised learner's loss at an equal rate to real samples. It is possible that the benefits of regularization imparted by generative augmentation are improved when fake sample losses are weighted significantly lower than real samples, to improve performance on the data distribution while regularizing with the benefit of indefinite generated samples as discussed in Section 4.2. The third and forth columns demonstrate this problem: at 10% and 5% real samples, the simple predictive model does overfit to the testing distribution and lose some accuracy, but the benefits of not seeing an overwhelming majority of highly-variable, heavily-weighted generated data is a greater boon. Finally, when performing 0-shot learning (Romera-Paredes and Torr, 2015), the CCAEGAN has the highest corresponding supervised learner performance. The relationship between the victorious CCAEGAN in FJD and testing accuracy is not a coincidence - Figure 3.4 shows the tight relationship between conditional image quality and testing performance of a downstream predictive model.

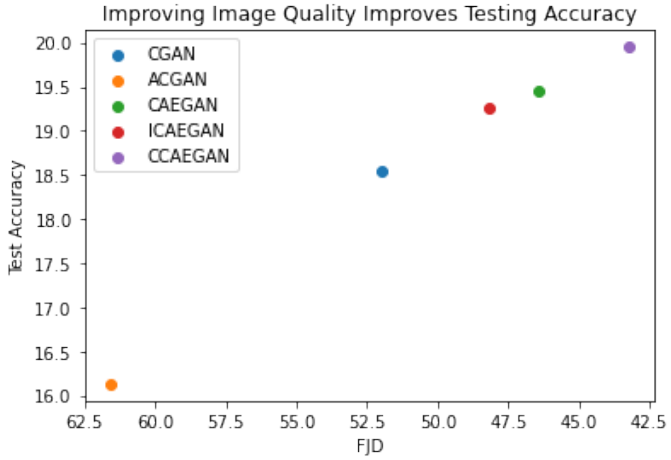


FIGURE 3.4: Testing accuracy increases linearly with conditional image quality.

Figure 3.4 demonstrates that higher FJD-scoring models correlate with higher testing accuracy for predictive models trained only on augmented data. This is a reasonable assumption but helpful to demonstrate that tuning models to optimize performance in the FJD metric is beneficial for downstream data augmentation tasks. Optimizing a single generative model using the variety of GAN best practices such as adaptive augmentation, discriminator loss blocking, and multi-step updates would be a promising direction of research to make theoretically optimal "zero-shot" (no training samples were used for training) accurate models which generalize to the testing set. NVIDIA's StyleGAN2 is a wonderful example of a model which incorporates the best of contemporary research on GAN modifications, and is capable of producing unconditional generation of CIFAR-10 manifold approximations with as low as 2.42 FID (Karras et al., 2020a). Two synergistic extensions to the work presented in this thesis exist: combining the successful StyleGAN2 architecture and training algorithms with the cyclical models, and injecting conditional techniques into the presently non-conditional model. With such incredible FID performance,

one may expect comparatively good FJD performance with the right conditioning technique, and by extension exceedingly powerful data augmentation.

3.3 Cyclical Models Perform Classification

The final result on the benchmarking contribution of thesis is the demonstration that cyclical models do not just improve on baseline generative models. The encoding component of each of the cycle models CAEGAN, ICAEGAN, and CCAEGAN perform the multiple attention task of encoding samples to the variation and code spaces: therefore we expect some degree of accuracy when we interpret the encoders as a solely predictive model. To validate this claim, 4 experiments were run with a comparable design to those in 3.2: a designated set of the CIFAR-10 training dataset was taken as the training data, then each row indicates a model trained from scratch on this subset. Instead of training a predictive model, however, the three GAN models each use the encoding portions of their architectures to encode the testing samples. It is important to note there have been no changes to the models nor training algorithms: each must still manage the tripartite loss given by reconstruction, encoding, and the adversarial generative loss. Rather, the same trained models sets from 3.5 are analyzed. The class space-encodings of this test set are taken as the class predictions, and the top-1 accuracy is taken.

	75% Real	25% Real	10% Real	5% Real
Predictor	**69.87** $\pm$ 0.23	**59.12** $\pm$ 0.06	**47.09** $\pm$ 0.07	32.60 $\pm$ 0.12
CAEGAN	68.55 $\pm$ 0.04	56.34 $\pm$ 0.08	45.78 $\pm$ 0.07	**41.07** $\pm$ 0.17
ICAEGAN	18.60 $\pm$ 0.80	20.71 $\pm$ 0.56	15.30 $\pm$ 0.89	14.05 $\pm$ 0.41
CCAEGAN	65.25 $\pm$ 0.25	50.86 $\pm$ 0.25	41.21 $\pm$ 0.18	34.30 $\pm$ 0.13

TABLE 3.6: Top-1 test accuracy without regard for generation. Higher is better.

Table 3.6 demonstrates that the complexity of handling autoencoding, generative, and supervised tasks is a weight for the models for the first three experimental designs. Though the CAEGAN and CCAEGAN testing accuracy is in stride with the simple predictor for 75% and 25% of the training set, the results fall off for the CCAEGAN somewhat when using 5000 samples. However, a turning point exists somewhere between 5000 and 2500 samples: the regularization imparted upon the encoder by managing the generative, reconstructive, and predictive tripartite loss dramatically improves the model's testing accuracy, as both the CAEGAN and CCAEGAN pull ahead of the simple supervised learner.

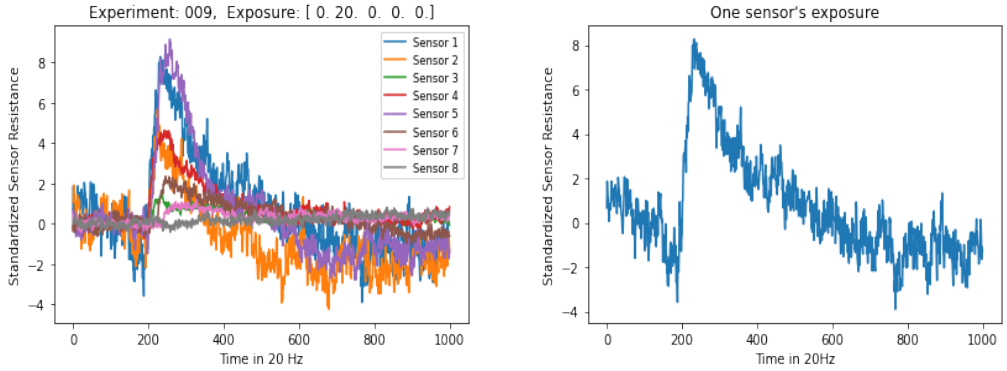
Chapter 4

Application

4.1 Chemical Sensing

The US Army DEVCOM Soldier Center is an Army research laboratory in Natick, Massachusetts. As part of the research assistantship funded by this laboratory, in this chapter we explore the use of conditional models for a chemical sensing task. Contemporary research on chemical sensing detects the exposure of chemical analytes using label prediction or regression given sensor resistance responses.

A physical box of eight sensors with unique polymer coatings is used to detect chemical anayltes in a vapor exposure. The detection device runs an electric current through each polymer-coated sensor which records a resistances through time. The scientists coat the platelets with diverse polymers such that the variation of resistances measured by the device should respond differently to different analyte compounds. In Figure 4.1, an example of experiment composed of eight sensor's concurrent exposure to 20 units of analyte A borne in vapor. From this 8-channel signal the machine must regress the composition of the analyte exposure. The considered set of analytes of interest to the stakeholder are analyte A, analyte B, analyte C, analyte D, and analyte E.



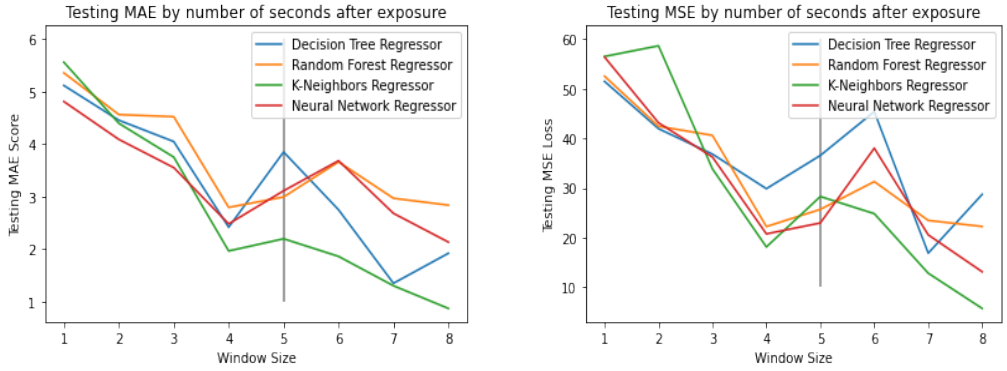
(A) One full experiment: eight polymer coatings are exposed to analyte A borne in vapor.

(B) One coated platelet's resistance during the exposure.

FIGURE 4.1

In the laboratory, the experiments are highly controlled. The sensors rest with typical air present for a baseline ten seconds. This time can be used to calculate the mean shift in sensor resistance to standardize the data, as well as capture large samples of the omnipresent sensor noise, one direction for future research in utilizing the noise distribution as a potential augmentation. Then a controlled flow of analyte vapor is exposed for a response time of thirty seconds, before a valve is shut. The exposure window can be seen in Figure 4.1 as the period of rapid increase in

resistance as the sensors are coated in the analyte. Then a desorption recovery time begins and lasts forty seconds in which the sensor resistances are recorded but no further analyte is exposed, indicated in 4.1 by the lengthy period of gradual decline in resistance. The analyte exposures are quantified by a molecular density in the exposed vapor, and accordingly the problem may be treated as a multiple classification (multiple analyte exposure), multiple regression (multiple analyte exposure magnitude), or a binary classification: the presence or non-presence of a particular agent. For the purposes of conditioning and prediction in this work, the multiple regression task is chosen.



(A) Mean absolute error, lower is better.

(B) Mean squared error, lower is better.

FIGURE 4.2: Baseline regression task results given variable exposure windows.

In an experiment on baseline results summarized in Figures 4.2a and 4.2b, data with only a minimal zero-mean preprocessing translation are used to train a variety of classic machine learning models. Models are selected by cross-validation over a hyperparameter grid search. Scores given as testing mean squared error and L1 loss (mean absolute error) on a holdout test set. K-neighbors outperforms parametric models fairly consistently across exposure window sizes, taken as the number of seconds after the exposure begins. Though research has been done explicitly on learning meaningful preprocessing for sensor data, these models serve as baselines for naive approaches to the multiple regression task Weiss et al., 2018.

For this thesis, the multiple regression task can be considered the highest degree of granularity, as it is a super-task to the multiple classification. Extensions to the disjoint conditioning used in 3 include non-disjoint conditioning for multi-class prediction tasks and sensitivity analysis which analyzes changes in sensor resistances as well as trained models to varying exposure magnitudes. Sensitivity analyses such as these would be crucial to the generalizability of a chemical sensing tool at testing time, as the stakeholder chemists cannot perform every combination of analytes at every exposure magnitude.

Investigation of this sensitivity problem takes two paths for this thesis. The first is to analyze the performance of a trained machine on a testing set with a diverse set of dangerous analyte exposures. These exposures could vary in magnitude from high to low concentrations, as the magnitude of exposure is important. However, the chemists can only supply a discrete set of experiments of harmful exposures. In addition, the sensor resistance response is not a linear relationship with exposure magnitude - experiment 26 uses double the exposure magnitude as experiment 14, sensor 5 responds at a significantly higher level relative to the low-exposure experiment than the differently-coated sensor 1, as illustrated in demonstrated in 4.3.

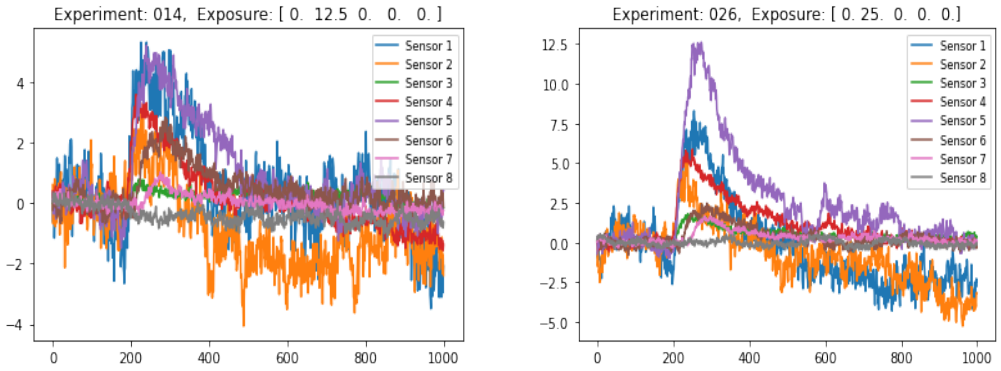


FIGURE 4.3: Sensor responses are not linear with exposure magnitude.

Making a model which is robust to these changes in signal magnitude is a vital inclusion to achieve the desired generalizability. In addition, this sensitivity problem poses some unique questions for developing models which incorporate additional information. Additionally a chemical exposure at a variable place in the chemical window is a practical task for a deployed tool. Data augmentation via generative models is one avenue towards developing models which are robust to the challenges of this task. A generative model maybe be able to exploit conditioning to generate samples which approximate the distribution of sample types not present in the training data, of which there are two types:

1. Magnitude Interpolation: given a set of exposures to one analyte at concentration a and concentration c, generate samples estimating exposure to b where $a < b < c$.

2. Condition Combination: given a set of exposures to analyte x and analyte y, generate samples estimating simultaneous exposure to $x + y$.

The second challenge which defines the chemical sensing task is the low-data paradigm. These experiments require intensive experimental design and supervision by a professional chemist, in addition to challenges with hardware and experimental integrity. For this reason, the research produces around 1000 usable samples per year. Research has been done to address complications of supervised tasks for chemical sensing, including pretraining an autoencoder with a latent Kalman Filter which estimates the first derivative of the signal (Weiss et al., 2018). Pretraining on random data is one approach which marked success in early detection. Another promising approach specific to generative models is differentiable augmentation of samples input to the discriminator in a conditional GAN (Karras et al., 2020b). Karras et al., 2020b further demonstrates that even tasks such as CIFAR-10 with 50,000 training samples can be categorized as low-data paradigms and benefit from adaptive augmentation. All established and proposed models employed in Section 3.1 could utilize this technique which introduces structured augmentations to samples evaluated by the discriminator, which is to be employed in implemented in future research.

4.2 GANs for Chemical Sensing

Little distinction is needed from the space of natural images when discussing the assumptions behind generation for chemical sensing. Just as in natural images the

generative assumption assumes the existence of some latent representation space V describing the meaningful semantic variation in the natural image space X, the same assumption is taken to describe the variation in the space of chemical sensing experiments.

The extension to conditioning in chemical sensing parallels that of the natural image generation task. Conditioning a latent space or class of models corresponds to including some labelled information to alter the latent space used for generative sampling. In the same manner, all models discussed in Chapter 1 may be extended to the chemical sensing task.

The model architectures may again be held constant, though the convolutional layers have been replaced with fully-connected layers as the extension from an RGB 3-channel problem to an 8-channel sensor problem is not well-established.

Further extensions to the conditional generation discussed in this thesis must be made to account for the dramatically different application task. Whereas the labels taken from the CIFAR-10 training images were sufficient to distinguish conditioning by classes for conditional generation in 3, the space of chemical sensing experiments is substantially more complicated than the ten disjoint image classes. The array of eight sensors could be potentially exposed to any combination of five analytes, each of some scalar exposure. These potential analyte combinations, called mixtures, could be any vector in $\mathbb{R}^5$, as opposed to one of ten disjoint classes, though the experiments provided by the stakeholder represent a tiny subset of these possibilities potentially spanning some bounded subset of $\mathbb{R}^5$, as there are five analytes of interest, often provided only single-analyte exposures. This sparse manifold has the potential to be fleshed-out with conditional generation. Thankfully, the models discussed and proposed in Chapters 1 and 2 may all be adapted, though some corresponding theoretical interpretations have been altered.

4.2.1 Leaking Augmentations

The complications and volatility of training GANs is made significantly worse by increasingly low-data paradigms, as discussed in 3.1.1. In this application, there are a mere 90 training samples divided across a multiple regression task with four labels (in the considered batch of experiments, no trials were performed with analyte E). Of these 90 samples, 24 are controls where no analyte was exposed, making an additional task for the regression model as these were kept in the dataset as a potential task of interest: the model is not always incentivized to predict the presence of an analyte. A further 11 samples are a conditioning control exposure of analyte A which stagger the otherwise uniform distribution over classes.

A partial remedy to this problematic number of samples is the inclusion of data augmentation which was not presently in the previous conditional model results in Chapter 3. Data augmentation is a well-established process in supervised learning in which training data are transformed in such a way that the meaning important to the label is preserved, but the pixels have been perturbed such as to prevent overfitting of the supervised learner (He et al., 2019). Augmentations in GANs, however, encourage the generator to generate augmented samples - an undesirable property which lowers quality considering augmentations are often rotations, noising, blocking, and sheering - a corruption called augmentation leaking (Zhang et al., 2019). This does not mean that augmentations are inappropriate for the chemical sensing task. Rather, the kinds of augmentations this thesis specifies are entirely acceptable to leak into the generated samples.

When an augmentation is performed to both training samples and generated samples before being input into the discriminator, it is said that there is a "lens" being placed in front of the camera of the discriminator. Through this potentially warping or destructive lens, the discriminator must infer from prediction of training and generated samples about the training distribution. Augmentations performed on inputs to the discriminator are considered "leaking" if the generator can learn to apply these augmentations itself as part of the adversarial learning process. This means that rather than learning an approximation of the training distribution, the generator learns to approximate the augmented distribution. This is often a problem for natural image tasks in which rotated, color-jittered, or sheered samples would be undesirable and unrealistic generated images. However, for the chemical sensing task, the transforms this thesis proposes are acceptable leaks, as the information content of the signal should remain unchanged under the following transforms:

1. Signal scaling: all sensor resistances in the sample are scaled by a value $0.5 < q < 1.5$. It would be desireable for a supervised learner to predict the same class for a signal and the same signal of some slightly higher or lower magnitude. For the multiple regression task, this transform loses some appeal as the magnitude of the signal is vital to infer the analyte exposure magnitude.

2. Signal translation: all sensor channels in the sample are translated by some number of sampling steps $0 < t < 100$. This corresponds to alternating the moment of chemical introduction to the system, and robust predictors according to this transform are a large benefit for applications involving higher variance in testing data.

3. Signal noising: the signal matrix is perturbed by some small-variance random normal noise. This transform should be a minimally invasive way to mitigate pixel-level overfitting of the supervised models, as there is substantial noise in the dataset. One alternative to random normal noise is using samples of the noise present in sensors when there is no analyte present. This type of "sensor noise" is easy to gather as the laboratory may run the sensor array at any time with no need to expose analytes into the device.

These transforms are considered an acceptable form of leaking, though ongoing investigation will be required if there are undesirable effects related to downstream supervised models. These are acceptable leaks as the distribution of augmented training data should contain the same semantic content as the training distribution - which should be beneficial to supervised learners, which are trained to mitigate overfitting to non-augmentation samples.

4.3 Application Results

The first result is the qualitative evaluation of conditional generation on the chemical sensing task. There is presently a lack of quantitative methods for the evaluation of conditional and unconditional generation of chemical sensor signals as there is FID and FJD for natural images. One work in progress beyond the scope of this thesis is the utilization of adsorption isotherm equations (Bazan et al., 2008). These closed-form equations may give theoretical platelet resistance responses as a function of the exposed analyte. This means given a condition exposure and signal, it may be possible to take the earthmover distance between the theoretical response

and a generated one as a distance from the generated to known target distribution (Arjovsky, Chintala, and Bottou, 2017).

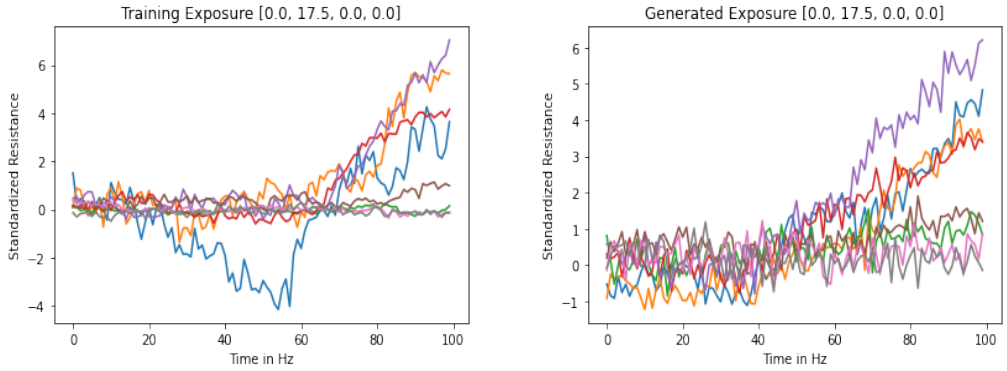


FIGURE 4.4: A training sample and an approximated sample at the moment of chemical exposure.

We compare an example training sample to a generated sample in 4.4. The real sample is one exposure to analyte B at magnitude 17.5 in the training set, and one generated sample by a trained model made by decoding one random factor of variation on the 17.5-unit analyte B condition. The generated sample displays symptoms of leaking augmentations by the level of noise present in sensors which should be fairly stable as judged by the training sample. In addition, some scales of some channels differ due to the scaling augmentation. However, this sample is a feasible exposures in the chemical sensing task and could lead to improved downstream performance as it shows increased resistance in the primary sensors of interest notable by qualitative analysis of the real signal.

4.3.1 Comparing Stochastic Augmentation to GAN Augmentation

This section quantifies if data set augmentation with conditional GANs are a meaningful upgrade to a supervised learner or a learner with training data augmentations. These experiments use the multi-class regression design in which no inductive assumptions are made to match the knowledge that the classes for this data set have disjoint single-analyte exposures - simply a 'relu' activation on the predictor outputs. In order to systematically quantify the improvement offered by transformative augmentations and dataset augmentations by conditional GANs while controlling for hyperparameters such as training epochs and learning rate, an exhaustive grid search is used. Table 4.1 shows testing MSE by training with hyperparameters learned by 10-fold cross validation for each combination of using stochastic augmentations and dataset augmentations with a conditional GAN:

	No Augmentation	With Augmentation
No CGAN	23.11 ± 0.23	29.46 ± 0.32
With CGAN	24.05 ± 0.22	$\mathbf{18.42} \pm 0.22$

TABLE 4.1: Testing loss given by ten trials using the optimal hyperparameters from the validation stage.

Table 4.2 quantifies the certainty in the experimental results from Table 4.1. The hypothesis test returns the probability that the hypothesis that the mean of the testing losses between two pairs of experiments is equal. By calculating the probability that the two means are equal, it is demonstrated with a high degree of certainty that for four of the six combinations of CGAN data set augmentation and data augmentations, the difference between GAN augmentation and stochastic augmentation did alter the mean of the testing loss distribution. This demonstrates that with statistical certainty the combination of CGAN augmentation and stochastic augmentation meaningfully lowered the testing loss in a greater manner than using either or neither. Vitally, this thesis's hypotheses on the utility of data set augmentation with conditional generation are vindicated as with a level of statistical certainty the mean of the CGAN and Aug distribution is lower than the other options - demonstrating the utility of conditional generation even in extremely low-data paradigms.

	No CGAN, No Aug	Aug, No CGAN	CGAN, No Aug
Aug, No CGAN	**0.026**		
CGAN, No Aug	0.680	0.054	
CGAN and Aug	**0.040**	**0.000**	**0.012**

TABLE 4.2: Probability of retaining the null hypothesis given the distribution of test performances. Low values indicate higher certainty that the mean of the distributions is unequal. Bold results indicate a certainty below the heuristic threshold of $p = 0.05$.

Though this result demonstrates the synergy in using both augmentation techniques to mitigate overfitting in the ultra-low data paradigm, by returning to 4.2a it may be observed that the classical models had comparatively low testing MSE scores.

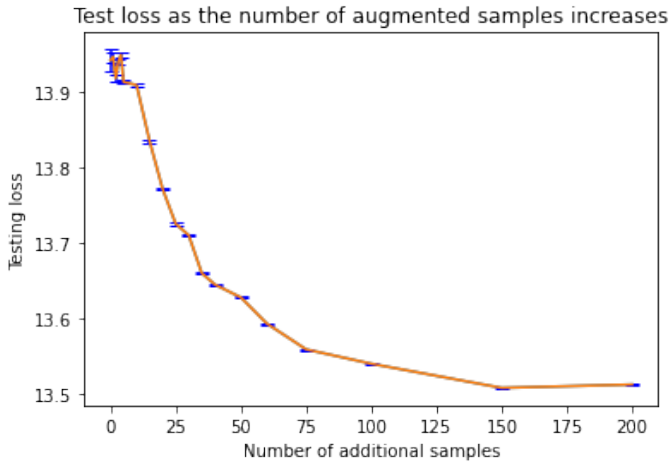


FIGURE 4.5: Testing loss with a variable number of generated samples.

Lastly, Table 4.5 demonstrates how testing losses for one experimental design can be both improved and regularized with the addition of CGAN data set augmentation. In this instance, a K-Neighbors model with fixed hyperparameters is trained using the training distribution as well as a variable number of additional samples made by a trained CGAN. For each point, 50 experiments are performed. It is apparent from inspection of the testing loss curve that a limit to the testing loss exists

where adding more testing samples decreases the testing loss and standard error of the estimate. This demonstration serves to highlight how even a model optimized on cross-validation can be improved by the addition of augmented samples - even in extremely low data paradigms where GANs notoriously struggle.

Appendix A

Appendix A

A.1 Factors of Variation

Both the established and proposed models for data generation include sampling of some latent space Z, samples of which contribute degrees of freedom and a learned semantic meaning in the decoded space. All proposed models must learn a generative function from standard normal independent samples of this low-dimensional space such that the adversarial loss returned by the discriminator is minimized. However, this models impart no design biases on how this latent space should be learned.enforce some structure on the latent space by penalizing that some points be decoded to particular training signals.

Autoencoding models CAEGAN and ICAEGAN use two methods of weakly structuring the latent space: one penalty which imparts that the C space is learned to match the training labels under encoding, and the reconstruction penalty which requires the model to decode both random normal samples, as well as training samples of a distribution learned by the encoder.

In this section, we analyze the tripartite decomposition of the source of variation in all data. As assumed in Section 1.1, the manifold hypothesis describes the meaningful variation in data lying on a significantly lower-dimensional surface than the data space. In this paradigm, all meaningful variations in data can be attributed to movement along some vector in the latent space. However, the types of variation present can be broken down further. As in Section 1.3.6, the data space can be described as a conditional distribution $P(x|z, c)$ via semantic meaning of the latent space z and class-conditioning c. In this distinction, we enforce a joint distribution where variations of z **or** c induce variations in the decoded samples.

The decomposition to a joint distribution using class labels is not the last step to understanding the latent dimension. A further decomposition can be made on the latent Z space, in which we specify the difference between **known** and **unknown** factors of variation. Known factors of variation need not be entirely labelled in the training data, but are simply known to be present and might have some known examples. This assumption comes from the observation that there are stylistic similarities between multiple samples of natural data. However, disentangled learning Locatello et al., 2019 does not allow for partial supervision of the latent space, as the semantic meaning attributed to each dimension must be inferred, not specified.

In the CIFAR-10 data set, the class 'Plane' might include propeller planes, jets, and commercial aircraft. The differences between these known-but-unlabelled subclasses are not explicit in an unregularized latent space, and must be inferred if they are present in a disentangled latent representation. However by decomposing the latent space z into explicit factors of variation v and degrees of freedom , the latent

space is imbued with greater expressivity. Now the condition space c and variation space v may be held constant with explicit semantic meaning, and the unexplained variation space $\tilde{z}$ maybe be randomly sampled. This variation-controlled conditioning gives 'anchoring' to the latent space such that more meaning can be easily encoded, and generated samples have more known factors than the class label condition.

A.2 Model Architectures

A.2.1 Encoding Architecture

Encoders are a family of function mapping an input point to some lower-dimension representation. For this thesis, there are three task-specific encoders with similar architecture.

Conditional Encoders

Conditional encoders are the big-to-small portion of the conditional autoencoder architecture. Though reconstruction and adversarial losses may be backpropogated to the conditional encoders, the primary component of their optimization is the conditional encoding of c given by $f(x) = (v, c)$. This conditional encoding is a supervised task for labelled training data where the loss is taken between the conditional encoding and the target encoding given by the class label. For input sample x, conditional encoders return representation $(v, c)|x$ given by:

```
rgb_32_Conditional_Encoder(
  (conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv2_bn): BatchNorm2d(256, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv3_bn): BatchNorm2d(512, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv4_bn): BatchNorm2d(1024, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv5v): Conv2d(1024, 100, kernel_size=(2, 2), stride=(1, 1))
  (conv5c): Conv2d(1024, 10, kernel_size=(2, 2), stride=(1, 1))
)
```

It may be observed the input to the conditional encoder is a 3-channel image. This will differ from the conditional big-to-small discriminator architectures, which do have access to the condition of the input sample when learning the corresponding representation. Whereas established conditional autoencoders return a latent representation conditioning on the input class $f(x|c) = v$, these conditional encoders double as predictive models: they both predict the condition in a supervised manner, as well as assign a latent representation such that the sample maybe be more easily decoded, enforced by the reconstruction error.

Discriminator Architecture

The discriminator architecture maps from a square 32*32 RGB image to a probability, defined by the composition of functions:

```
rgb_32_Discriminator(
  (conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2_bn): BatchNorm2d(256, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv3_bn): BatchNorm2d(512, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv4_bn): BatchNorm2d(1024, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv5): Conv2d(1024, 1, kernel_size=(2, 2), stride=(1, 1))
)
```

The conditional discriminator employed by CGAN 1.3.8 utilize one learned linear embedding from the code dimension to one channel of the image dimension, then stacks the two into an 'information' point in $\mathbb{R}^{32^4}$:

```
rgb_32_Conditional_Discriminator(
  (fc1): Linear(in_features=10, out_features=1024, bias=False)
  (conv1): Conv2d(4, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2_bn): BatchNorm2d(256, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv3_bn): BatchNorm2d(512, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv4_bn): BatchNorm2d(1024, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv5): Conv2d(1024, 1, kernel_size=(2, 2), stride=(1, 1))
)
```

Further, the ACGAN 1.3.9 discriminator is a conditional discriminator, but without the 1.3.8 information stack. The model must instead perform a multi-headed prediction and classification task given by fc_{adv} and fc_{aux} respectively:

```
rgb_32_Auxiliary_Classifier_Discriminator(
  (conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2_bn): BatchNorm2d(256, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv3_bn): BatchNorm2d(512, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv4_bn): BatchNorm2d(1024, eps=1e−05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv5): Conv2d(1024, 1, kernel_size=(4, 4), stride=(1, 1))
```

```
  (fc_adv): Linear(in_features=4096, out_features=1, bias=True)
  (fc_aux): Linear(in_features=4096, out_features=10, bias=True)
)
```

Predictor Architecture

The predictor architecture used as a baseline supervised classification model is standard across all experiments. The predictor uses identical convolutional layers to other big-to-small models, with the exception of the classification output:

```
rgb_32_P(
  (conv1): Conv2d(3, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv2_bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv3): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv3_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv4): Conv2d(512, 1024, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
      bias=False)
  (conv4_bn): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (conv5c): Conv2d(1024, 10, kernel_size=(2, 2), stride=(1, 1))
)
```

It may be observed that the predictor is identical to the discriminator with a 10-channel output representing a probability distribution over the predicted classes rather than a probability of being sampled from the training distribution.

A.2.2 Decoding Architecture

Decoding architectures are a general term for small-to-big models, which map either encoded points or latent samples to the data space. How this function is learned differs by model.

Unconditional Generation

Used by autoencoders and GANs, unconditional generation maps a low-dimensional latent sample to the 32*32 RGB data space:

```
rgb_32_Decoder(
  (deconv1): ConvTranspose2d(100, 1024, kernel_size=(4, 4), stride=(1, 1), bias=
      False)
  (deconv1_bn): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv2): ConvTranspose2d(1024, 512, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv2_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
```

```
  (deconv3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv3_bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv4): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv4_bn): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv5): ConvTranspose2d(128, 3, kernel_size=(3, 3), stride=(1, 1), padding
      =(1, 1))
)
```

Conditional Generation

Conditional generators use two arguments: a latent or encoded sample (v, c), where v is a random or encoded variation vector, and c is a random or encoded class label. The c vector is embedded to the image dimension 32*32 and the two are stacked into the 'information' point:

```
rgb_32_Conditional_Decoder(
  (deconv1v): ConvTranspose2d(100, 1024, kernel_size=(4, 4), stride=(1, 1), bias=
      False)
  (deconv1c): ConvTranspose2d(10, 1024, kernel_size=(4, 4), stride=(1, 1), bias=
      False)
  (deconv1_bn): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv2): ConvTranspose2d(2048, 512, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv2_bn): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv3_bn): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv4): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
      padding=(1, 1), bias=False)
  (deconv4_bn): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
  (deconv5): ConvTranspose2d(128, 3, kernel_size=(1, 1), stride=(1, 1))
)
```

A.3 Structuring

Structuring refers to inductive biases on the model on the nature of the distribution of a space. For example, the Variational Autoencoder (VAE) Kingma and Welling, 2014 imparts a KL-divergence over the distribution of the encoded points with respect to a multivariate standard normal distribution. A basic autoencoder imparts no such assumptions on the distribution of the encoded training data as chosen by the model, other than the choice of activation which bounds the encoding space.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{n=1} N e^{z_n}} \tag{A.1}$$

A.4 Chemical Sensing

Table demonstrating the cross-validation selected hyperparameters:

	Epochs	Learning Rate
No CGAN, No Aug	45	$1e-4$
No CGAN, with Aug	15	$1e-4$
With CGAN, No Aug	35	$5e-5$
With CGAN and Aug	35	$1e-4$

TABLE A.1: Optimal hyperparameters for each paradigm by random 10-fold cross validation.

A.5 Generated samples per model

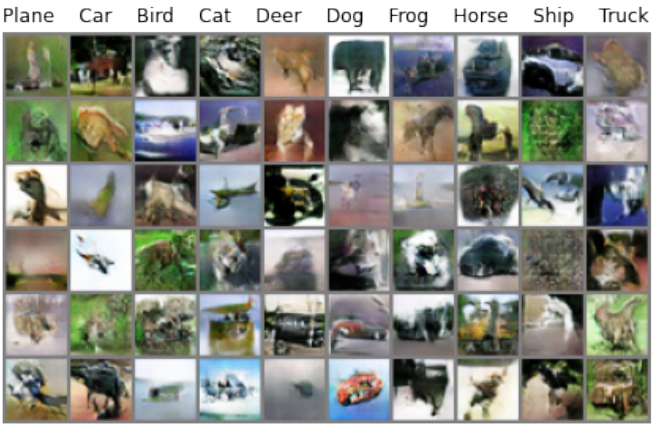


FIGURE A.1: Unconditional generation by the GAN.

A.6 Specific Model Architecture

A.6.1 GAN

A.6.2 Conditional GAN

A.6.3 Autoencoder

A.7 FID Extended

Borrowed from DeVries et al., 2019:

$$\mu = \frac{1}{N}\sum_{i=0}^{N} f(x^{(i)}), \quad \Sigma = \frac{1}{N-1}\sum_{i=0}^{N}(f(x^{(i)}) - \mu)(f(x^{(i)}) - \mu)^T \tag{A.2}$$

FIGURE A.2: Conditional generation by the CGAN.

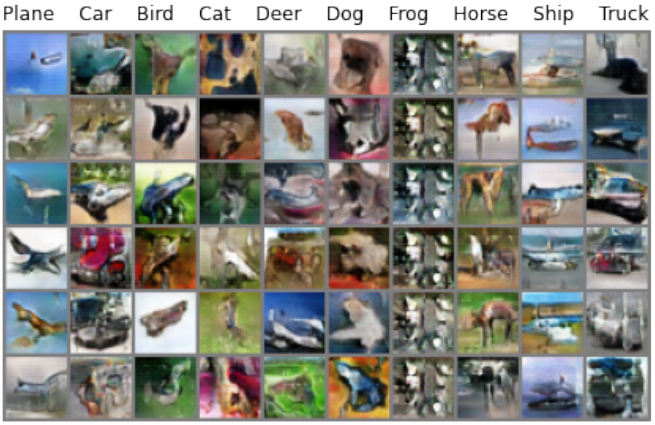


FIGURE A.3: Conditional generation by the ACGAN.

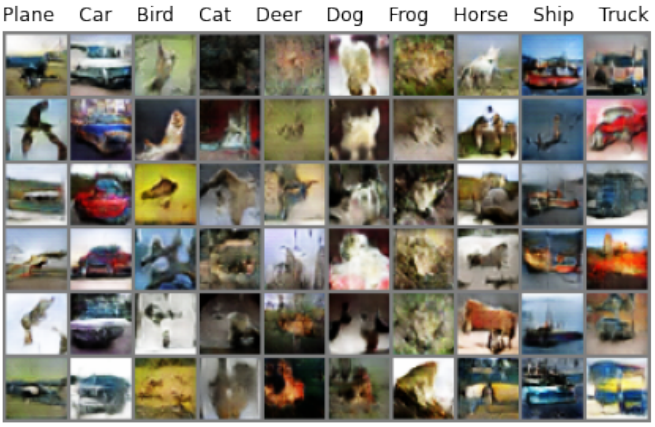


FIGURE A.4: Conditional generation by the CAEGAN.

$$\hat{\mu} = \frac{1}{N}\sum_{i=0}^{N} f(\hat{x}^{(i)}), \quad \hat{\Sigma} = \frac{1}{N-1}\sum_{i=0}^{N}(f(\hat{x}^{(i)}) - \hat{\mu})(f(\hat{x}^{(i)}) - \hat{\mu})^T \tag{A.3}$$

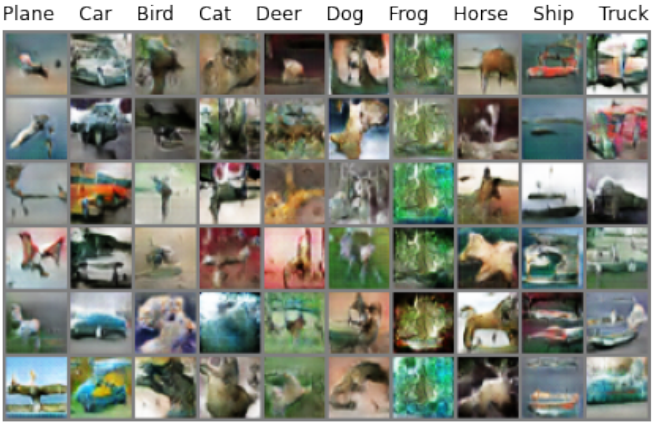


FIGURE A.5: Conditional generation by the ICAEGAN.

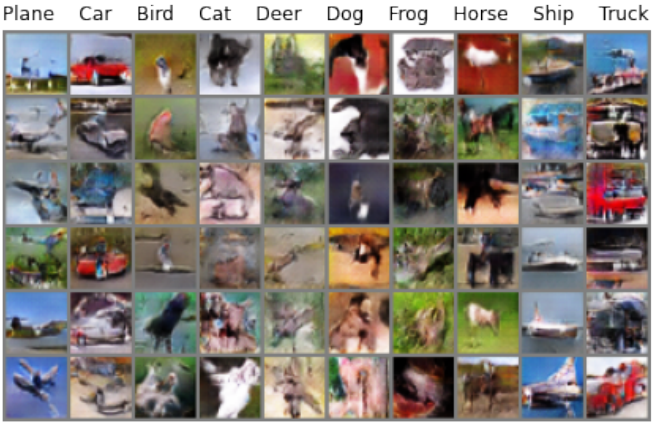


FIGURE A.6: Conditional generation by the Cycle-CAEGAN.

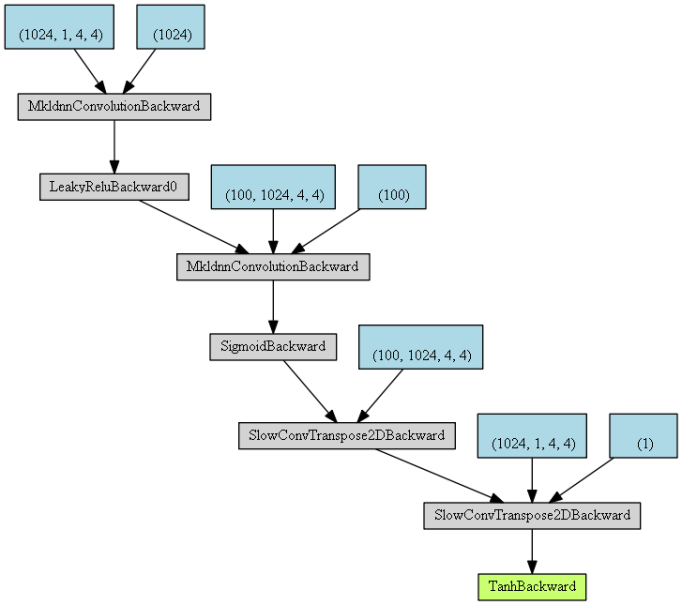


FIGURE A.7: Autoencoder Architecture, simplified

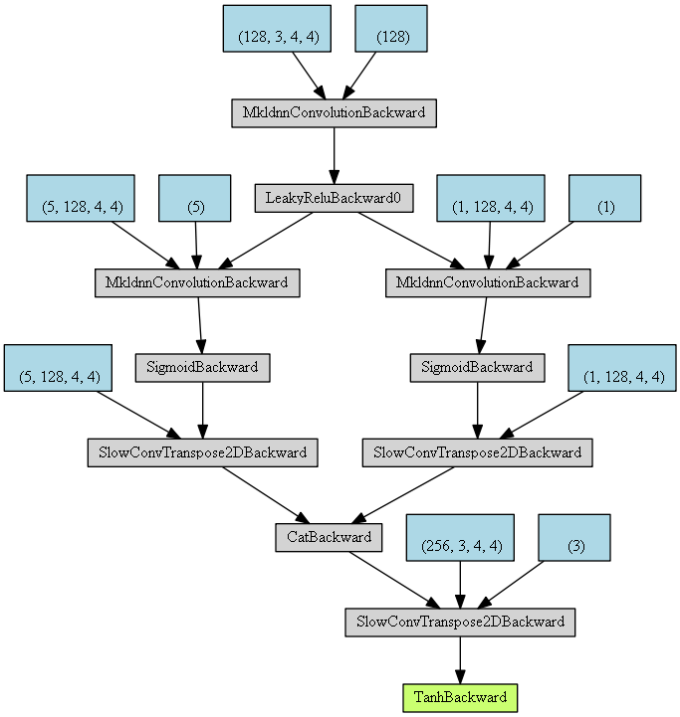


FIGURE A.8: Hidden Conditional Autoencoder Architecture.

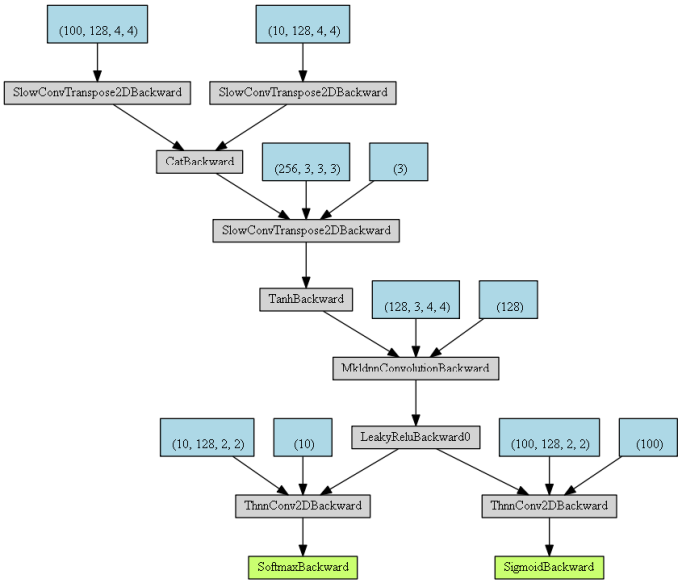
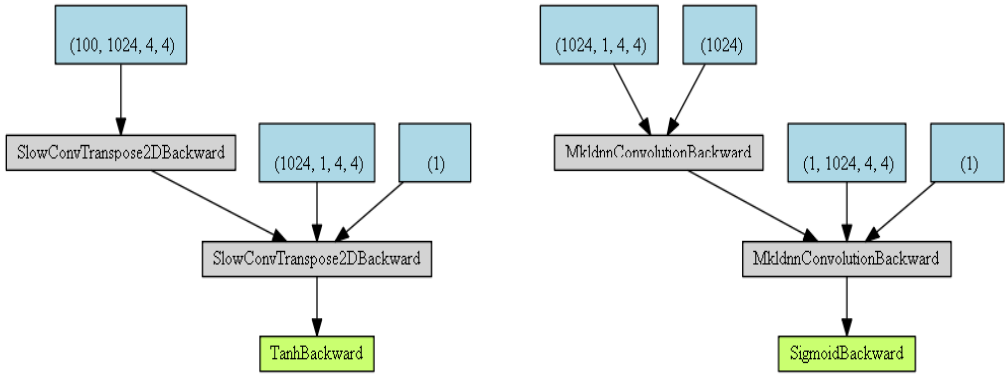
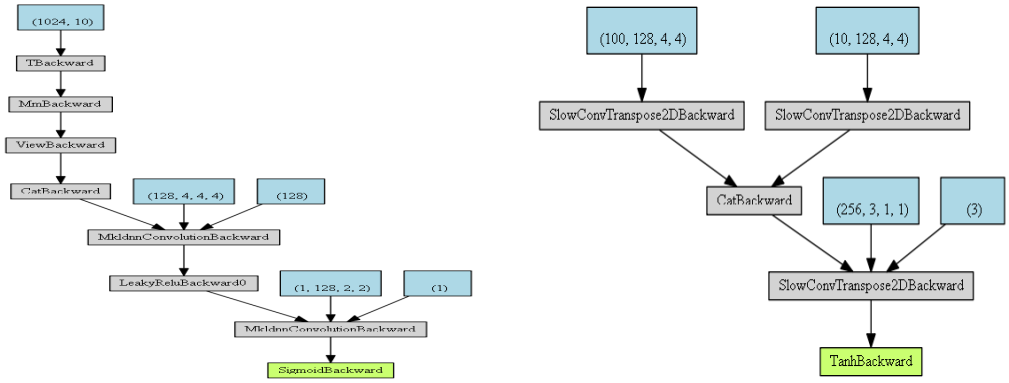


FIGURE A.9: Inverse Conditional Autoencoder Architecture.



(A) Generator architecture. (B) Discriminator architecture.

FIGURE A.10: A minimal GAN architecture diagram.



(A) Generator architecture. (B) Discriminator architecture.

FIGURE A.11: A minimal CGAN architecture diagram.

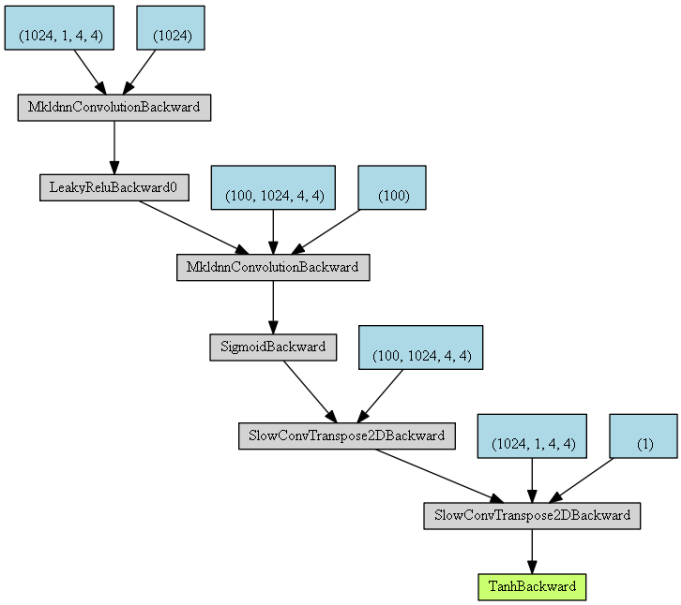


FIGURE A.12: Non-conditional Autoencoder

Bibliography

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR, pp. 214–223.

Bazan, RE et al. (2008). "Adsorption equilibria of natural gas components on activated carbon: pure and mixed gas isotherms". In: *Adsorption Science & Technology* 26.5, pp. 323–332.

Dai, Bo and Dahua Lin (2017). "Contrastive learning for image captioning". In: *arXiv preprint arXiv:1710.02534*.

DeVries, Terrance et al. (2019). "On the evaluation of conditional gans". In: *arXiv preprint arXiv:1907.08175*.

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (2016). "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782*.

Foret, Pierre et al. (2020). *Sharpness-Aware Minimization for Efficiently Improving Generalization*. arXiv: 2010.01412 [cs.LG].

Goodfellow, Ian J et al. (2014). "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661*.

He, Tong et al. (2019). "Bag of tricks for image classification with convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 558–567.

Heusel, Martin et al. (2017). "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *arXiv preprint arXiv:1706.08500*.

Higgins, Irina et al. (2016). "beta-vae: Learning basic visual concepts with a constrained variational framework". In:

Jolicoeur-Martineau, Alexia (2018). "The relativistic discriminator: a key element missing from standard GAN". In: *arXiv preprint arXiv:1807.00734*.

Karras, Tero et al. (2018). "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Hk99zCeAb.

Karras, Tero et al. (2020a). *Analyzing and Improving the Image Quality of StyleGAN*. arXiv: 1912.04958 [cs.CV].

Karras, Tero et al. (2020b). "Training generative adversarial networks with limited data". In: *arXiv preprint arXiv:2006.06676*.

Kingma, Diederik P and Max Welling (2014). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [stat.ML].

Korotin, Alexander et al. (2020). *Wasserstein-2 Generative Networks*. arXiv: 1909.13082 [cs.LG].

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In:

Larsen, Anders Boesen Lindbo et al. (2016). *Autoencoding beyond pixels using a learned similarity metric*. arXiv: 1512.09300 [cs.LG].

Locatello, Francesco et al. (2019). "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *international conference on machine learning*. PMLR, pp. 4114–4124.

Mathiasen, Alexander and Frederik Hvilshøj (2020). *Fast Fréchet Inception Distance*. arXiv: 2009.14075 [cs.LG].

Mescheder, Lars, Andreas Geiger, and Sebastian Nowozin (2018). "Which training methods for GANs do actually converge?" In: *International conference on machine learning*. PMLR, pp. 3481–3490.

Mishra, Ashish et al. (2018). "A generative model for zero shot learning using conditional variational autoencoders". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2188–2196.

Odena, Augustus, Christopher Olah, and Jonathon Shlens (2017). *Conditional Image Synthesis With Auxiliary Classifier GANs*. arXiv: 1610.09585 [stat.ML].

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434*.

Romera-Paredes, Bernardino and Philip Torr (2015). "An embarrassingly simple approach to zero-shot learning". In: *International conference on machine learning*. PMLR, pp. 2152–2161.

Salimans, Tim et al. (2016). "Improved techniques for training gans". In: *arXiv preprint arXiv:1606.03498*.

Szegedy, Christian et al. (2015). *Rethinking the Inception Architecture for Computer Vision*. arXiv: 1512.00567 [cs.CV].

Touvron, Hugo et al. (2021). "Going deeper with Image Transformers". In: *arXiv preprint arXiv:2103.17239*.

Weiss, Matthew et al. (2018). "Applications of the kalman filter to chemical sensors for downstream machine learning". In: *IEEE Sensors Journal* 18.13, pp. 5455–5463.

Wieting, John and Douwe Kiela (2019). *No Training Required: Exploring Random Encoders for Sentence Classification*. arXiv: 1901.10444 [cs.CL].

Zadorozhnyy, Vasily, Qiang Cheng, and Qiang Ye (2020). "Adaptive Weighted Discriminator for Training Generative Adversarial Networks". In: *arXiv preprint arXiv:2012.03149*.

Zhang, Han et al. (2019). "Consistency regularization for generative adversarial networks". In: *arXiv preprint arXiv:1910.12027*.

Zhu, Jun-Yan et al. (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232.