Adversarial Autoencoders

Authors - Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Brendan Frey

Conference - International Conference on Learning Representations (ICLR), 2016

Literature survey done by -

1. Akshay Shukla - 2022A7PS0087P
2. Deepan Roy - 2022A7PS0004P

Table Of Contents

1) Background

Generative modeling tries to learn and generalise to data distributions for tasks such as classification, synthesis and unsupervised learning. Traditional deep generative models like RBMs (Restricted Boltzmann Machines) and DBNs (Deep Belief Networks) relied heavily on MCMC (Markov Chain Monte Carlo) sampling, which was computationally expensive and often unstable as it required many steps to generate good samples, especially for high-dimensional data. Also, if the Markov chains don't mix well (i.e. fail to explore the full space), the samples become biased, leading to poor model updates and unreliable learning.

The rise of Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) introduced backpropagation friendly alternatives. However, VAEs enforce latent priors using KL divergence, which restricts flexibility, while GANs lack an explicit encoder and suffer from mode collapse. Hence using any of these models individually is not of great use.

The authors propose a method to bridge this gap with the help of the Adversarial Autoencoders (AAEs). These models combine the reconstruction capabilities of autoencoders with the distribution-shaping power of GANs to regularize the latent space. It does this by introducing a discriminator into a typical VAE implementation.

2) Model Overview

The adversarial autoencoder is composed of

- Encoder $q(z|x)$: Maps the data(x) to the latent code (z)
- Decoder $p(x|z)$: Reconstructs data from z
- Discriminator $D(z)$: Trained to distinguish between latent codes from $q(z|x)$ and samples from a chosen prior $p(z)$

The training consists of two phases per minibatch - Reconstruction Phase and Regularisation Phase. The reconstruction phase optimizes the autoencoder to minimize reconstruction loss. The regularization phase uses GAN-style training to match the aggregated posterior $q(z)$ to $p(z)$.

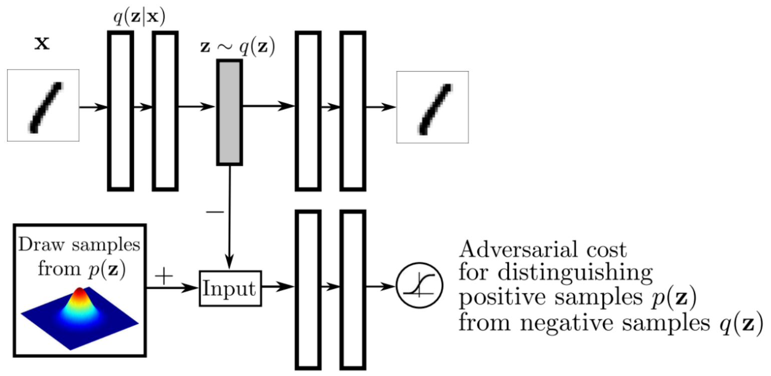

$q(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
$\mathbf{z} \sim q(\mathbf{z})$
Draw samples from $p(\mathbf{z})$
$+$
$-$
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

Figure 1: Architecture of an adversarial autoencoder. The top row is a standard autoencoder that reconstructs an image $\mathbf{x}$ from a latent code $\mathbf{z}$. The bottom row diagrams a second network trained to discriminatively predict whether a sample arises from the hidden code of the autoencoder or from a sampled distribution specified by the user.

3) Technical Contributions

- Aggregated Posterior Matching - Instead of matching individual posteriors q(z|x) to a prior (as in VAEs), AAEs match the aggregated posterior q(z), which allows for more flexible and expressive latent distributions.
- Latent Distribution Flexibility - The encoder can be deterministic (output is a fixed vector), Gaussian (output consists of a mean and variance which are sampled with the help of reparameterization) and a universal approximator (encodes data using a function of input and noise, allowing arbitrary posteriors).
- Semi-Supervised Extension - AAEs incorporate label information using a one-hot vector during adversarial training, associating each class with a specific mode of a mixture prior (e.g. mixture of Gaussians). This helps regularize the latent space to become more discriminative.

4) Comparison to Related Models

VAEs regularize using KL divergence, requiring a closed-form prior. AAEs use adversarial training, requiring only sample access to the prior and provide sharper latent manifolds with better interpolation results.

GANs generate data from noise directly and do not deal with training an encoder. AAEs on the other hand encode data and can generate by sampling from a structured latent space.

GMMNs (Generative Moment Matching Network) use maximum mean discrepancy (MMD) to match distributions. AAEs use adversarial loss for tighter distribution matching and are trained end to end.

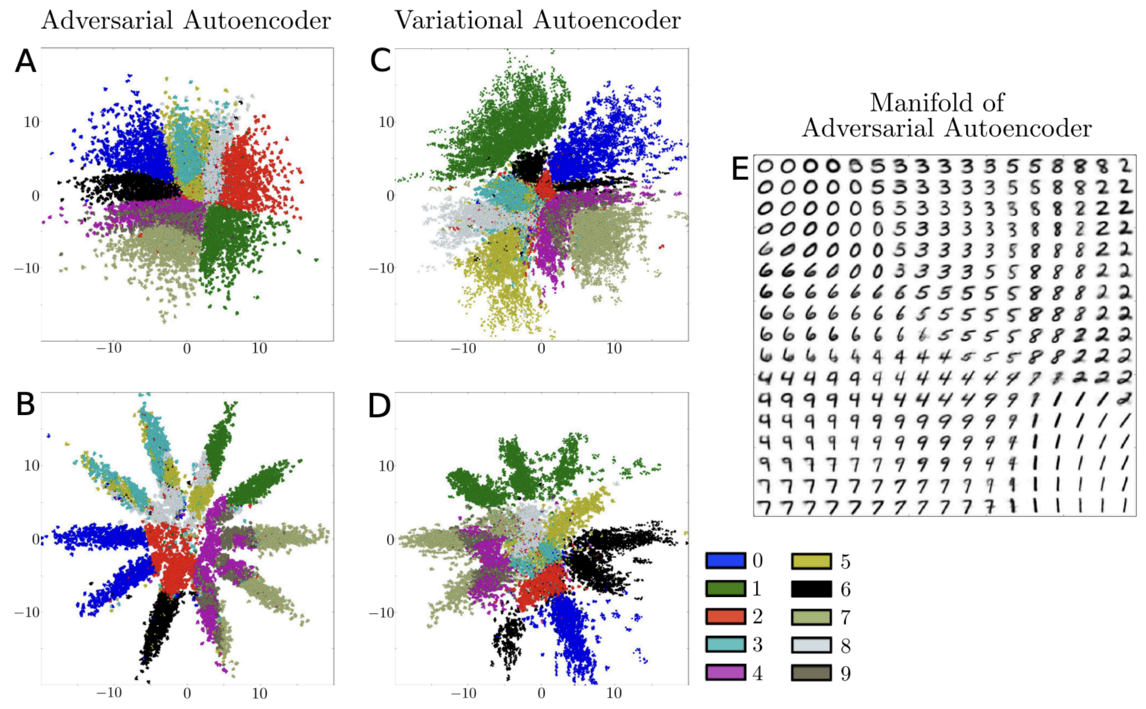
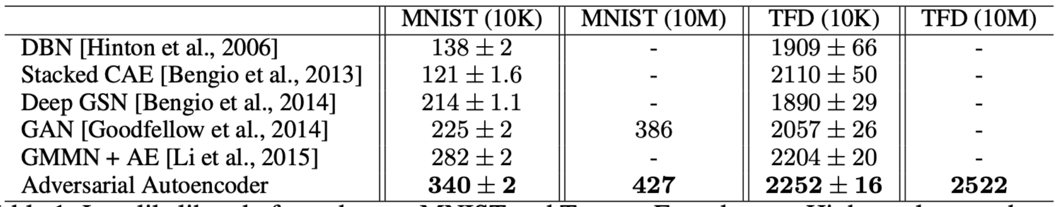


Figure 2: Comparison of adversarial and variational autoencoder on MNIST. The hidden code $\mathbf{z}$ of the *hold-out* images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) a 2-D gaussian and (d) a mixture of 10 2-D Gaussians. (e) Images generated by uniformly sampling the Gaussian percentiles along each hidden code dimension $\mathbf{z}$ in the 2-D Gaussian adversarial autoencoder.

5) Applications And Results

Generative Modeling

AAEs outperform GANs and VAEs in Parzen-window estimated log-likelihood on MNIST and TFD datasets. The results of experiments are as follows -



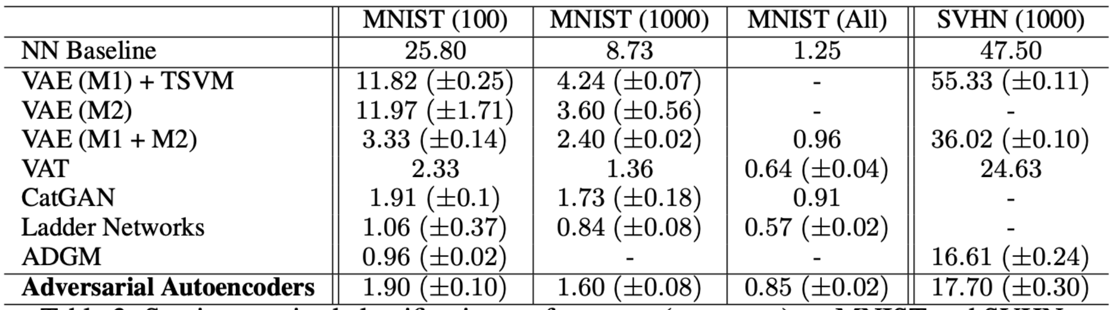
	MNIST (10K)	MNIST (10M)	TFD (10K)	TFD (10M)
DBN [Hinton et al., 2006]	138 ± 2	-	1909 ± 66	-
Stacked CAE [Bengio et al., 2013]	121 ± 1.6	-	2110 ± 50	-
Deep GSN [Bengio et al., 2014]	214 ± 1.1	-	1890 ± 29	-
GAN [Goodfellow et al., 2014]	225 ± 2	386	2057 ± 26	-
GMMN + AE [Li et al., 2015]	282 ± 2	-	2204 ± 20	-
Adversarial Autoencoder	$\mathbf{340 \pm 2}$	**427**	$\mathbf{2252 \pm 16}$	**2522**

Table 1: Log-likelihood of test data on MNIST and Toronto Face dataset. Higher values are better. On both datasets we report the Parzen window estimate of the log-likelihood obtained by drawing 10K or 10M samples from the trained model. For MNIST, we compare against other models on the real-valued version of the dataset.

Semi-Supervised Learning

In settings with limited labels:

- 100 labeled MNIST samples: AAE achieves 1.90% error (better than VAE-based M1/M2 and comparable to CatGAN).
- SVHN with 1000 labels: AAE achieves 17.7% error.

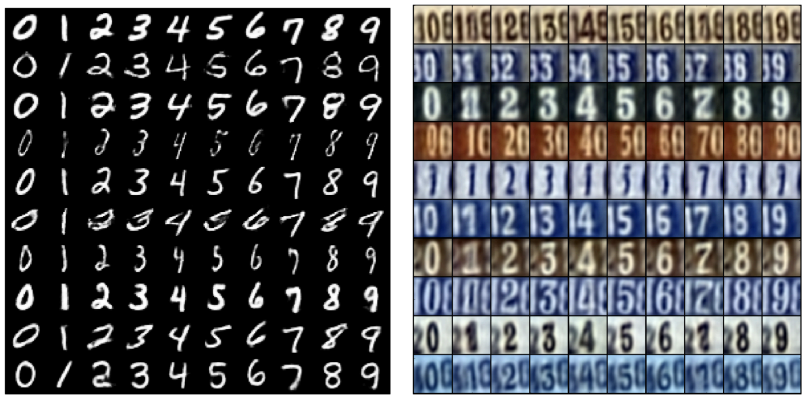


	MNIST (100)	MNIST (1000)	MNIST (All)	SVHN (1000)
NN Baseline	25.80	8.73	1.25	47.50
VAE (M1) + TSVM	11.82 ($\pm$0.25)	4.24 ($\pm$0.07)	-	55.33 ($\pm$0.11)
VAE (M2)	11.97 ($\pm$1.71)	3.60 ($\pm$0.56)	-	-
VAE (M1 + M2)	3.33 ($\pm$0.14)	2.40 ($\pm$0.02)	0.96	36.02 ($\pm$0.10)
VAT	2.33	1.36	0.64 ($\pm$0.04)	24.63
CatGAN	1.91 ($\pm$0.1)	1.73 ($\pm$0.18)	0.91	-
Ladder Networks	1.06 ($\pm$0.37)	0.84 ($\pm$0.08)	0.57 ($\pm$0.02)	-
ADGM	0.96 ($\pm$0.02)	-	-	16.61 ($\pm$0.24)
Adversarial Autoencoders	1.90 ($\pm$0.10)	1.60 ($\pm$0.08)	0.85 ($\pm$0.02)	17.70 ($\pm$0.30)

Table 2: Semi-supervised classification performance (error-rate) on MNIST and SVHN.

Disentangled Representations

AAEs can separate different factors of variation in data such as digit identity (content) and handwriting techniques(style) by feeding class labels as one-hot vectors into the decoder during training. This setup forces the hidden code to represent style-related variations while the label encodes content. As a result, changing the label alters the digit identity, while keeping the hidden code constant preserves the writing style. This disentanglement is useful for generating controlled variations of data (e.g. different digits in the same style).



(a) MNIST (b) SVHN

Figure 7: Disentangling content and style (15-D Gaussian) on MNIST and SVHN datasets.

Unsupervised Clustering

Even without labeled data, AAEs can discover meaningful clusters in the latent space. The model treats part of the latent code as a categorical variable (e.g. one-hot vector) and uses adversarial training to match it to a categorical prior. When applied to MNIST with 30 latent clusters, the AAE groups digits based on subtle differences (like slant or loop presence), achieving a classification error of 4.10%, which is significantly better than other models such as CatGAN's 9.70%. This demonstrates the AAE's capacity to learn class-like groupings in an unsupervised way.

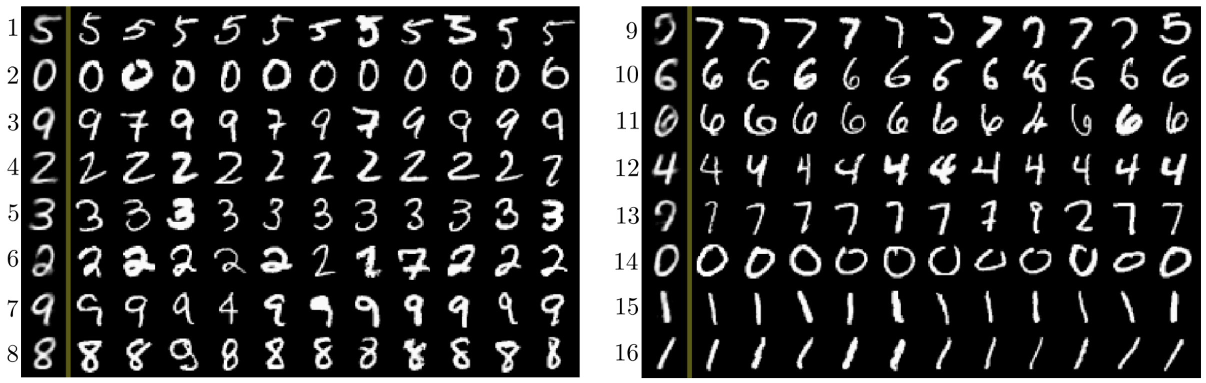


Figure 9: Unsupervised clustering of MNIST using the AAE with 16 clusters. Each row corresponds to one cluster with the first image being the cluster head. (see text)

Dimensionality Reduction

AAEs can perform nonlinear dimensionality reduction while maintaining semantic meaning in the reduced space. By imposing structured priors (e.g. Gaussian for style, Categorical for class) and combining them in a compact latent code, AAEs produce low-dimensional embeddings that visually separate different data clusters. When trained with a 2D latent space on MNIST, the network cleanly separates digits into clusters and achieves a semi-supervised classification error of 3.90% demonstrating both visualization quality and practical utility.

6) Strengths and Limitations

Strength of AAE are -

- Combines strengths of autoencoders and GANs.
- Supports flexible priors and complex latent spaces.
- Capable of semi-supervised learning, clustering, and visualization.
- High sample quality and meaningful latent interpolations.

The limitations of AAE are -

- Training involves alternating between reconstruction and adversarial steps, which can be complex to tune.
- Parzen window log-likelihood estimates are known to be an imperfect metric for generative performance.
- Requires a higher degree of computational power as against similar models like VAEs and GANs.

7) Conclusion

This paper introduces a powerful and versatile generative framework that regularizes latent spaces through adversarial training. AAEs unify representation learning, generative modeling, and semi-supervised classification under a single architecture and show superior performance across multiple tasks and datasets.