# Langevin Autoencoders for Learning Deep Latent Variable Models

IE 506 Course Project

**Team Name: Supernova**
Members
**Tarun Bisht**
**Indian Institute of Technology Bombay**
IEOR

April 27, 2023

# Presentation Outline

# Problem Description

# Problem Description



- One of the goals of unsupervised learning is to model the distribution of a given dataset $x \sim D$, i.e., model distribution $p(x)$ such that $p(x) \sim p_D$. Here $x$ contains samples from unknown distributions $D$. These models are called generative models.
- After we have learned the distribution, we can
  - find probability of arbitrary data point $x$, $p(x)$
  - sample point $x$ from distribution, $x \sim p(x)$.
- This project focuses on one of the types of generative models called Latent Variable models. These models compute the dataset's exact or approximate distribution functions.

# Problem Description



- We train these models using Maximum Likelihood estimation over some training dataset.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \; -\sum_{i=1}^{m} \log p_\theta(x^i)$$

- To solve this optimization problem, gradient descent-based methods can be applied for which the gradient of the objective function needs to be calculated.
- While calculating gradient expression we get

$$\nabla_\theta \log p_\theta(x) = \int p_\theta(z|x) \nabla_\theta \log p_\theta(x, z) dz$$

- We need to compute posterior $p(z|x)$ to compute the gradient.

# Problem Description



- The posterior calculation is intractable due to the lack of an analytic solution to the integral.
- We try to approximate the posterior distribution. There are two classes of methods for this:
  - **Variational Inference** approximate the posterior with a tractable distribution. Ex. Variational Autoencoder (VAE)
  - **Markov Chain Monte Carlo(MCMC)** provide sample based approximation of posterior distribution. Ex. Langevin Autoencoder (LAE)
- This paper uses MCMC based method (Langevin Dynamics) to approximate posterior.
- We want stationary distribution of Langevin Dynamics equation to be same as target posterior distribution, enabling sampling from posterior by simulating above equation.

# Langevin Autoencoder



**Algorithm 2** Langevin Autoencoders

$\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{\psi} \leftarrow$ Initialize parameters
**repeat**
    $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \sim \hat{p}(\mathbf{x})$ ▷ Sample a minibatch of $n$ examples from the training data.
    **for** $t = 0, \ldots, T-1$ **do** ▷ Run ALD iterations.
        $V_t = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi} g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
        $\boldsymbol{\Phi}' \sim q\left(\boldsymbol{\Phi}' \mid \boldsymbol{\Phi}\right) := \mathcal{N}\left(\boldsymbol{\Phi}'; \boldsymbol{\Phi} - \eta \nabla_{\boldsymbol{\Phi}} V_t, 2\eta \boldsymbol{I}\right)$
        $V_t' = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi}' g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
        $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}'$ with probability $\min\left\{1, \frac{\exp(-V_t') q(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}')}{\exp(-V_t) q(\boldsymbol{\Phi}' \mid \boldsymbol{\Phi})}\right\}$. ▷ MH rejection step.
    **end for**
    $V_T = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi} g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^{T} V_t$ ▷ Update the decoder.
    $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \frac{1}{T} \sum_{t=1}^{T} V_t$ ▷ Update the encoder.
**until** convergence of parameters
**return** $\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{\psi}$

# Work done before Midterm and Comments

# Work Done before Midterm Review



- Reading overview of generative models, assigned paper, relevant referenced materials etc.
- Implemented paper code using PyTorch and Torch distributions
- Benchmarking was done between VAE and proposed LAE on their capability to reconstruct given input for MNIST dataset.

# Comments on Midterm Review

- Check generation capability comparing VAE and LAE latent space interpolation results.
- Analyze the representation learning capability of VAE and LAE.

# Addressing Comments during mid-term



- Model was trained for some longer epochs, and generation capability is checked by sampling from latent space and reconstructing input. Along with interpolation in latent space was performed.
- To analyze the representation learning capability of VAE and LAE, latent space representation of input data was used to train a single-layer neural network for both VAE and LAE, and accuracy was used as the measure.

# Work Done after Midterm

# Work done after Midterm



- Image generation through random sampling from Gaussian noise was done.
- Some comparison of the effect of different parameters in LAE that are not in the paper.
- Analyze the representation learning capability of VAE and LAE was done
- A new training method is proposed and experiments were done regarding it.
- Adversarial robustness of the proposed method is compared.

# Experiments by Team

# Effect of Number of Steps in LAE



latent dim z=2, no mh, step size 0.001, batch size 512, early stopping patience 10 and loss BCE

| Model | Reconstruction Loss (MSE) | Val Loss (BCE) |
|---|---|---|
| LAE\| ns=2 | 0.00000743449 | 136.040208 |
| LAE\| ns=10 | 0.00000716265 | 132.89959 |
| LAE\| ns=50 | 0.00000729323 | 134.36413 |

# Effect of MH in LAE



LAE z=2, num_steps 2, step_size 0.001, batch_size 512, early stopping patience 10, loss BCE

| Model | Reconstruction Loss (MSE) | Val Loss (BCE) |
|---|---|---|
| LAE\| no_mh | 0.00000743449 | 136.040208 |
| LAE\| mh | 0.00000748080 | 136.238208 |

# Learned encoded representations for classification



LAE num step = 2 and no mh step size 0.001, single layer with ReLU activation and linear output activation

| Model | Train Loss | Train Accuracy | Val Loss | Val Accuracy |
|---|---|---|---|---|
| VAE$\|z = 2$ | 0.001352 | 0.0005407 | 0.001338 | 0.0005586 |
| LAE$\|z = 2$ | 0.00173053 | 0.0003460 | 0.00166865 | 0.00037973 |
| VAE$\|z = 8$ | 0.00074552 | 0.00071939 | 0.00074213 | 0.0007355 |
| LAE$\|z = 8$ | 0.00085344 | 0.00078394 | 0.00061837 | 0.00080696 |
| VAE$\|z = 16$ | 0.00082766 | 0.00069669 | 0.00080636 | 0.00072118 |
| LAE$\|z = 16$ | 0.00149354 | 0.00074028 | 0.00086874 | 0.00081385 |

- More number of steps leads to better accuracy.
- Increasing latent dimension also helps in both VAE and LAE.
- Using mh increases validation accuracy and decreases validation loss.

# Testing classifier training with latent sampling method

LAE: num step = 2 and no mh step size 0.001, epochs = 10

| Model | Val Loss | Val Accuracy (%) |
|---|---|---|
| None$\|ns = 0$ | 0.001153 | 95.66 |
| LAE$\|ns = 2$ | 0.001747 | 94.21 |
| VAE$\|ns = 2$ | 0.001770 | 93.99 |
| LAE$\|ns = 5$ | 0.002130 | 92.91 |
| VAE$\|ns = 5$ | 0.002134 | 92.61 |

# Adversarial Attacks on trained models



| Model | Accuracy | Epsilon = 0.01 | Epsilon = 0.1 | Epsilon = 0.2 | Epsilon = 0.3 |
|---|---|---|---|---|---|
| None \| ns = 0 | 95.66 | 94.69 | 70.85 | 13.18 | 0.16 |
| LAE \| ns = 2 | 94.21 | 93.03 | 66.42 | 11.31 | 0.33 |
| VAE \| ns = 2 | 93.99 | 92.66 | 63.40 | 11.16 | 0.35 |
| LAE \| ns = 5 | 92.91 | 91.31 | 62.62 | 10.60 | 0.41 |
| VAE \| ns = 5 | 92.61 | 90.83 | 60.89 | 10.32 | 0.41 |

Accuracy of different models for different values of epsilon

# Adversarial Attacks on trained models



| Model | Val Loss | $\epsilon = 0.01$ | $\epsilon = 0.1$ | $\epsilon = 0.2$ | $\epsilon = 0.3$ |
|---|---|---|---|---|---|
| None \| ns=0 | 0.001153 | 0.1702 | 0.8187 | 3.2136 | 6.6067 |
| LAE \| ns=2 | 0.001747 | 0.2523 | 0.9044 | 2.9268 | 5.5999 |
| VAE \| ns=2 | 0.001770 | 0.2630 | 0.9968 | 3.2381 | 6.1211 |
| LAE \| ns=5 | 0.002130 | 0.3044 | 0.9958 | 3.0319 | 5.6184 |
| VAE \| ns=5 | 0.002134 | 0.3114 | 1.0703 | 3.3443 | 6.1619 |

Validation loss adversarial attack

# Conclusions

# Conclusion



- Although LAE was better than VAE, the factor is not too big.
- Latent space learned by VAE has a better structure and control than LAE.
- Training method proposed has shown some positive results towards adversarial robustness, and these generative models can be paired up with the training of ML models.

# Future Direction

# Future Direction



- More tests towards the adversarial robustness of the proposed method can be done by using different datasets and training for longer epochs.
- This training idea can be extended and its usefulness can be measured on datasets having class imbalance problems, and data scarcity problems.

# References




- https://theaisummer.com/latent-variable-models
- https://abdulfatir.com/blog/2020/Langevin-Monte-Carlo/
- https://www.youtube.com/watch?v=FMuvUZXMzKM
- https://arxiv.org/abs/2209.07036

# Thank You!