Anomaly Detection using VAEs, Normalizing Flows for New Physics Searches

Introduction:

- Pratik Jawahar
- MS. in Robotics Engineering & AI at Worcester Polytechnic Institute (MA, USA)
- Research Focus: Deep Learning, Fault Detection, Computer Vision
- DIANA-HEP Fellowship 2020; Supervisor: Dr. Maurizio Pierini
- Previously worked as a short-term intern with EP-CMX group (GEM Detectors) in 2018

Objectives:

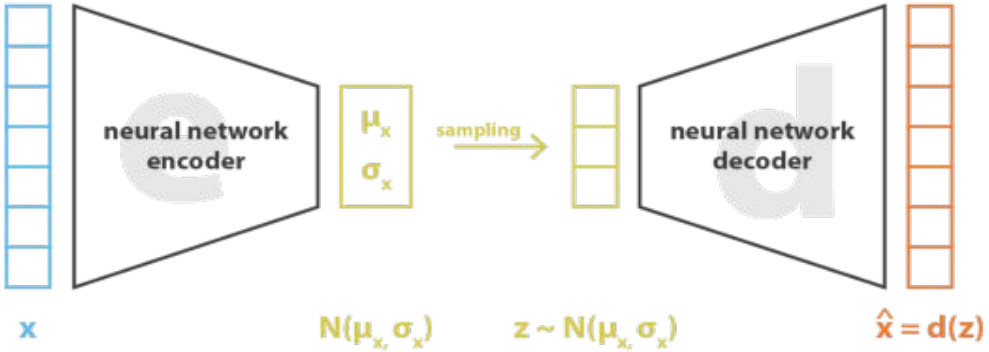
- Develop Anomaly Detection Algorithms to aid in New Physics Searches at the LHC
- Explore Deep Learning based Algorithms to classify BSM events as Anomalies

Scope:

- Probe the use of Variational Autoencoders (VAEs) as potential Anomaly Detectors
- Improve standard VAE performance through latent space optimization

Variational Autoencoders as Anomaly Detectors:

- Train the network to reconstruct the standard class (SM events)
- Feed anomalies to the trained network; Ideally leads to poor reconstruction
- Thresholding strategy on reconstruction loss to detect anomalies (BSM events)



- 2D ConvVAE
- Custom Sparse Loss Function

Potential Issue with VAEs (Scope for improvement):

- Thresholding on loss requires a significant separation in loss values for SM and BSM events.
- Which can be facilitated by better reconstruction of SM events (implies worse reconstruction of anomalies)
- Latent distribution as a Gaussian!
- Might not be the best distribution for good reconstruction
- Finding a better latent distribution is key!

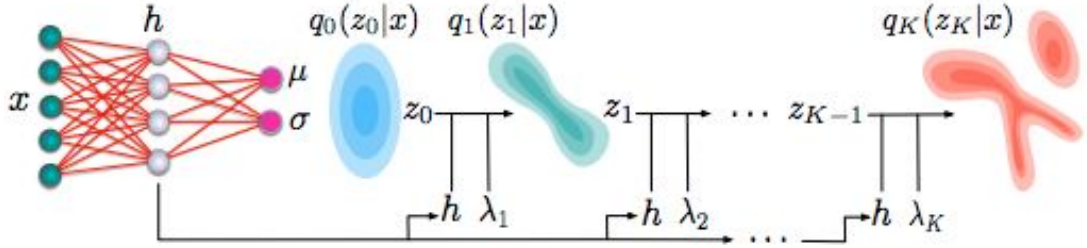
Normalizing Flows:

- Invertible transformations that convert a given distribution (*z*) to a target distribution (*z'*)
- State of the art algorithms in Variational Inference currently implement different types of flows
- https://arxiv.org/pdf/1505.05770.pdf

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\top \mathbf{z} + b),$$

$$p_1(\mathbf{z}') = p_0(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$$

$$\mathbf{z}_K = f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z})$$



Project Status:

Built and tested following models:

1. ConvVAE (Custom sparse loss function)
2. ConvVAE + Planar Flows
3. ConvVAE + Orthogonal Sylvester Flows
4. ConvVAE + Householder Sylvester Flows
5. ConvVAE + Triangular Sylvester Flows
6. ConvVAE + Inverse Autoregressive Flows (Linear)
7. ConvVAE + ConvFlow

Sylvester Normalizing Flows:

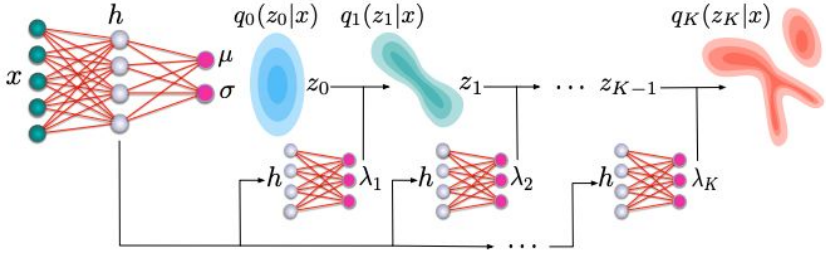
- Uses Sylvester Identity to calculate determinant of Jacobian in the flow algorithm
- Relaxes constraints on flow parameters A, B.
- Reduces computation intensiveness
- Three types: Orthogonal, Householder, Triangular
- https://arxiv.org/pdf/1803.05649.pdf

$$\mathbf{z}' = \mathbf{z} + \mathbf{A}h(\mathbf{B}\mathbf{z} + \mathbf{b})$$

Theorem 1 (Sylvester's determinant identity). *For all* $\mathbf{A} \in \mathbb{R}^{D\times M}, \mathbf{B} \in \mathbb{R}^{M\times D}$,

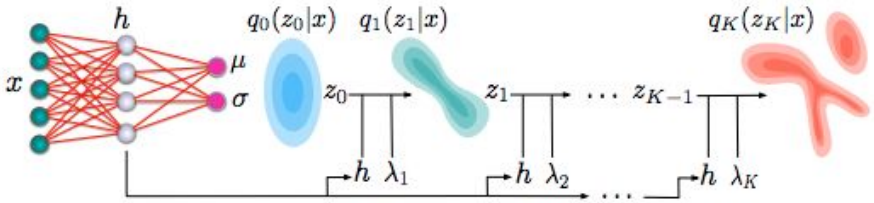
$$\det(\mathbf{I}_D + \mathbf{AB}) = \det(\mathbf{I}_M + \mathbf{BA}), \qquad (11)$$

where $\mathbf{I}_M$ *and* $\mathbf{I}_D$ *are* M *and* D*-dimensional identity matrices, respectively.*



Inverse Autoregressive Flows:

- Autoregressive formulations (invertible transformations) require sequential steps to sample a single vector 'z'
- The inverse however is easier to sample from and can be computed in parallel, hence the name IAF
- Flow parameters in the inverse transformation are independent of input 'x'
- https://arxiv.org/pdf/1606.04934.pdf



$$z_0 = \bar{\mu}_0 + \bar{\sigma}_0 \cdot \epsilon_0$$
$$z_i = \bar{\mu}_i(\mathbf{z}_{1:i-1}) + \bar{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot \epsilon_i, \quad i = 1, \ldots, D$$

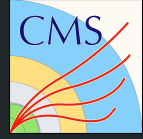
Convolutional Flows:

- Makes use of 1D convolutions to reduces net number of parameters per flow layer
- Better time complexity since jacobian calculations are faster
- Permits stacking multiple transformations per layer
- Better suited for Deep Learning applications since the flow itself involves Conv Layers and doesn't need to maintain a parallel hypernetwork
- https://arxiv.org/pdf/1711.02255.pdf

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u} \odot h(\mathrm{conv}(\boldsymbol{z}, \boldsymbol{w}))$$

Future Work:

- Look at efficient thresholding strategies
- Explore more efficient neural network architectures for the Encoder-Decoder setup
- Explore more robust loss functions for this particular case
- Look into inference time optimization



Thank You!