STATISTICAL DATA DEPTH IN DEEP LEARNING: APPLICATION TO OUT OF DISTRIBUTION DETECTION

Ananya Kumar
Mentor : Spencer Szabados
Department of Mathematics

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

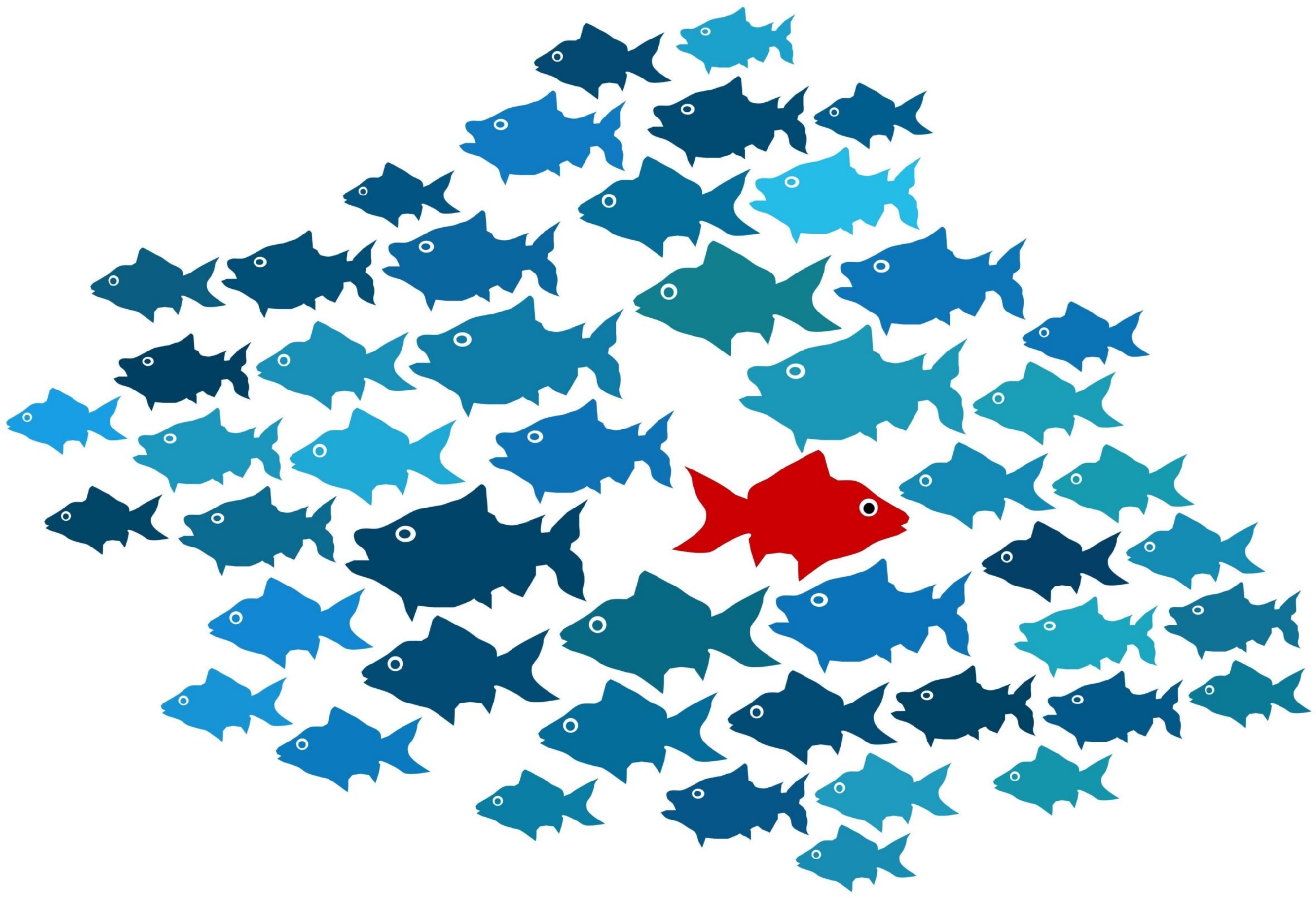
INTRODUCTION

OUT OF DISTRIBUTION DETECTION

A model's ability to recognize and appropriately handle data that deviates significantly from its training set.

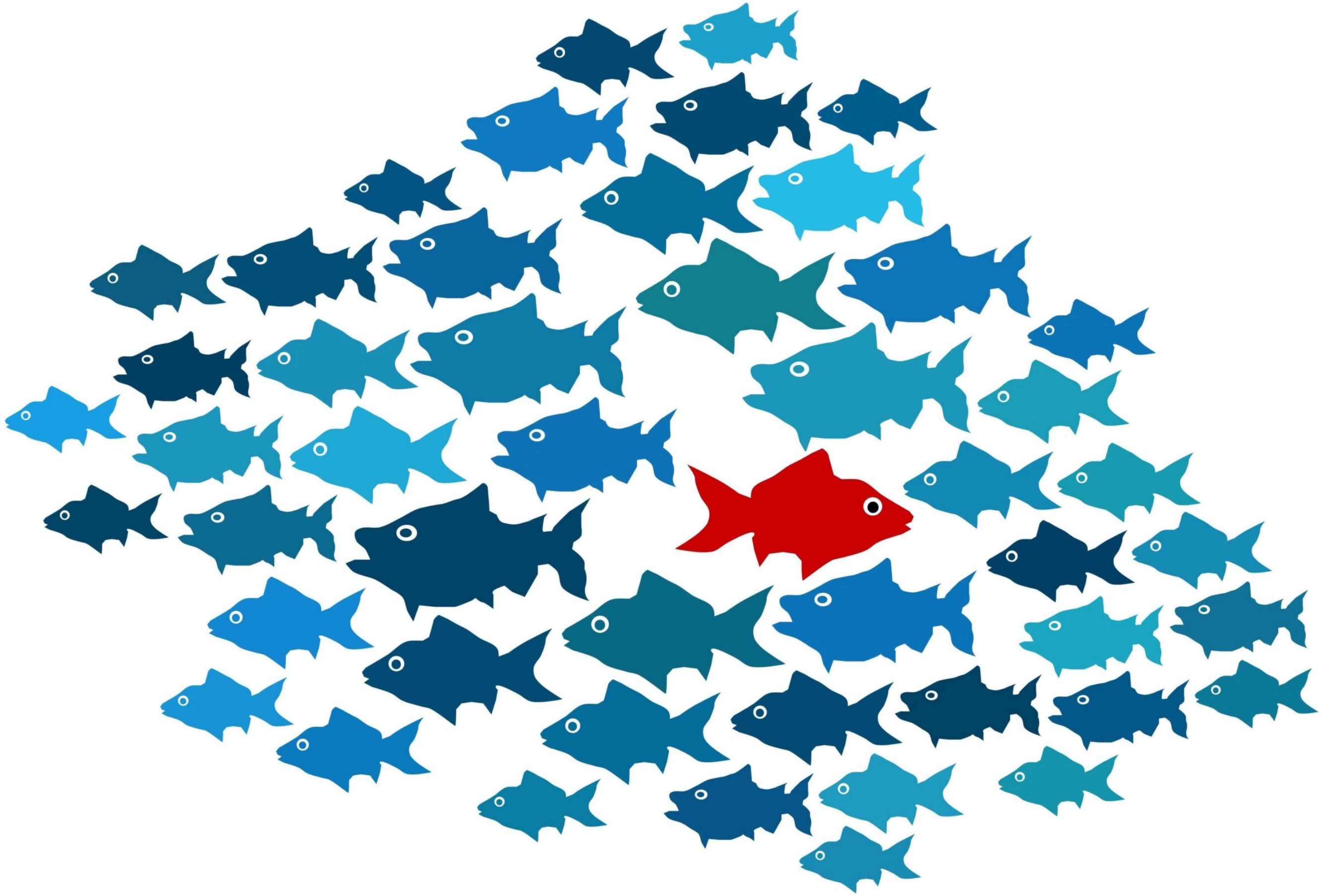
Importance:

- Error Reduction
- Data Quality Control
- Model Robustness
- Safety and Reliability



OUT OF DISTRIBUTION DETECTION – CHALLENGES

- Defining “Out-of-Distribution”
- High Dimensionality
- Computation Costs
- Domain Shift
- Model Calibration
- Noise and Outliers
- Transferability
- Evaluation Metrics



STATISTICAL DATA DEPTH

A **data depth** measures how close a given point is located to the center of a distribution. For $x \in \mathbb{R}^p$ and a p-variate random vector X distributed as $P \in \mathcal{P}$, a data depth is a function

$$D : \mathbb{R}^p \times \mathcal{P} \to [0,1], \qquad (\boldsymbol{x}, P) \longmapsto D(\boldsymbol{x}|P)$$

that is :

D1 **translation invariant:** $D(\boldsymbol{x} + b \,|X + b) = D(\boldsymbol{x}|X)$ for any $b \in \mathbb{R}^p$;

D2 **linear invariant:** $D(A\boldsymbol{x} \,|AX) = D(\boldsymbol{x}|X)$ for any $p \times p$ non-singular matrix A;

D3 **vanishing at infinity:** $\lim_{\|\boldsymbol{x}\| \to \infty} D(\boldsymbol{x}|X) = 0$;

D4 **monotone on rays:** for any $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x} \in \mathbb{R}^p} D(\boldsymbol{x}|X)$, any $x \in \mathbb{R}^p$, and any $0 \leq \alpha \leq 1$ it holds: $D(\boldsymbol{x}|X) \leq D(\boldsymbol{x}^* + \alpha(\boldsymbol{x} - \boldsymbol{x}^*)|X)$;

D5 **upper semicontinuous in $\boldsymbol{x}$:** the upper-level sets $D_\alpha(\boldsymbol{x}|X) \leq \{\boldsymbol{x} \in \mathbb{R}^p : D(\boldsymbol{x}|X) \geq \alpha\}$ are closed for all α.

STATISTICAL DATA DEPTH

There are many definitions of data depth:

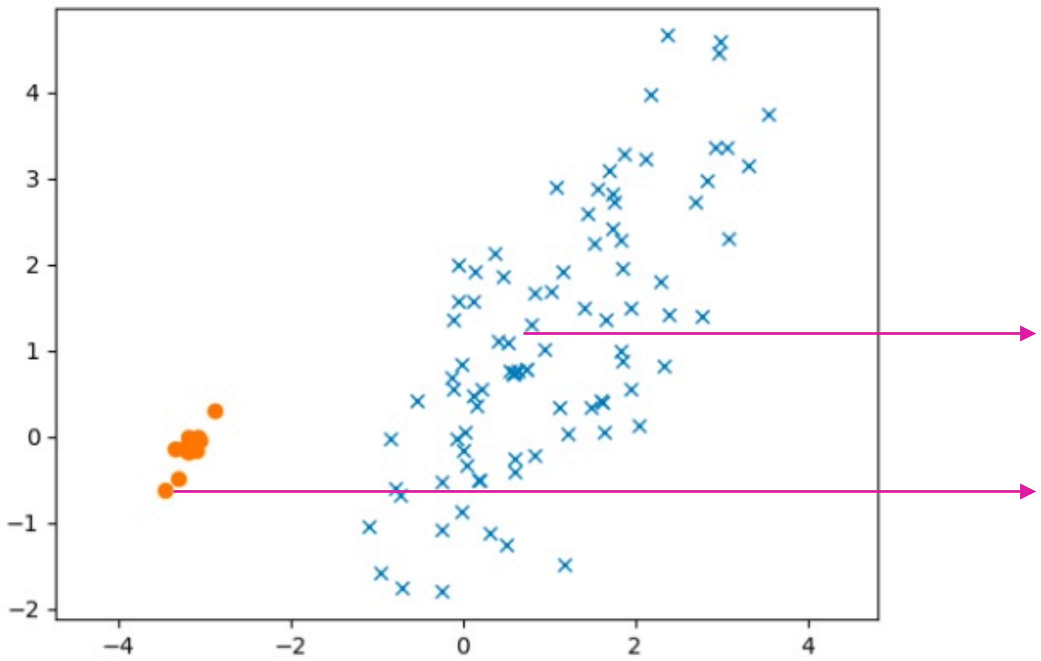
- Mahalanobis depth (Mahalanobis, 1936)
- Convex hull peeling depth (Barnett, 1976; Eddy, 1981)
- Projection depth (Stahel, 1981; Donoho, 1982)
- Simplicial volume depth (Oja, 1983)
- Simplicial depth (Liu, 1990)
- Majority depth (Singh, 1991)
- Zonoid depth (Koshevoy and Mosler, 1997)
- Regression Depth (Rousseeuw and Hubert, 1999)
- L_p-depth (Zuo and Serfling, 2000)
- Spatial depth (Serfling, 2002)
- Expected convex hull depth (Cascos, 2007)
- Geometrical depth (Dyckerhoff and Mosler, 2011)
- Lens depth (Liu and Modarres, 2011)

STATISTICAL DATA DEPTH

There are many definitions of data depth:

- Mahalanobis depth (Mahalanobis, 1936)
- Convex hull peeling depth (Barnett, 1976; Eddy, 1981)
- Projection depth (Stahel, 1981; Donoho, 1982)
- Simplicial volume depth (Oja, 1983)
- Simplicial depth (Liu, 1990)
- Majority depth (Singh, 1991)
- Zonoid depth (Koshevoy and Mosler, 1997)
- Regression Depth (Rousseeuw and Hubert, 1999)
- L_p-depth (Zuo and Serfling, 2000)
- Spatial depth (Serfling, 2002)
- Expected convex hull depth (Cascos, 2007)
- Geometrical depth (Dyckerhoff and Mosler, 2011)
- Lens depth (Liu and Modarres, 2011)

STATISTICAL DATA DEPTH – BIVARIATE DEMO



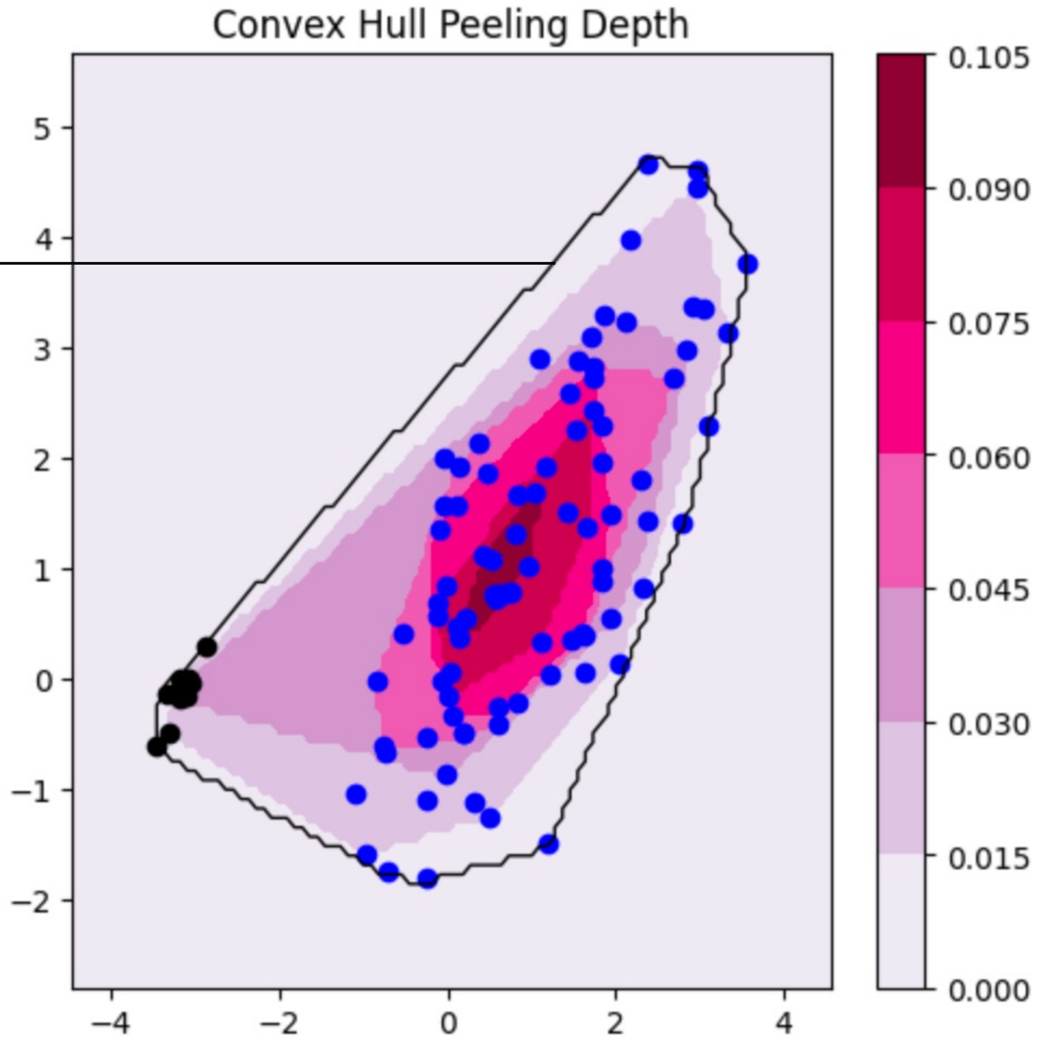
$$\sim \text{Norm}\left(\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}1 & 1\\1 & 2\end{pmatrix}\right)$$

$$\sim \text{Norm}\left(\begin{pmatrix}-3.181\\-0.222\end{pmatrix}, \frac{1}{36}\begin{pmatrix}1 & 1\\1 & 2\end{pmatrix}\right)$$

Synthetic data generated using two different multivariate normal distributions.

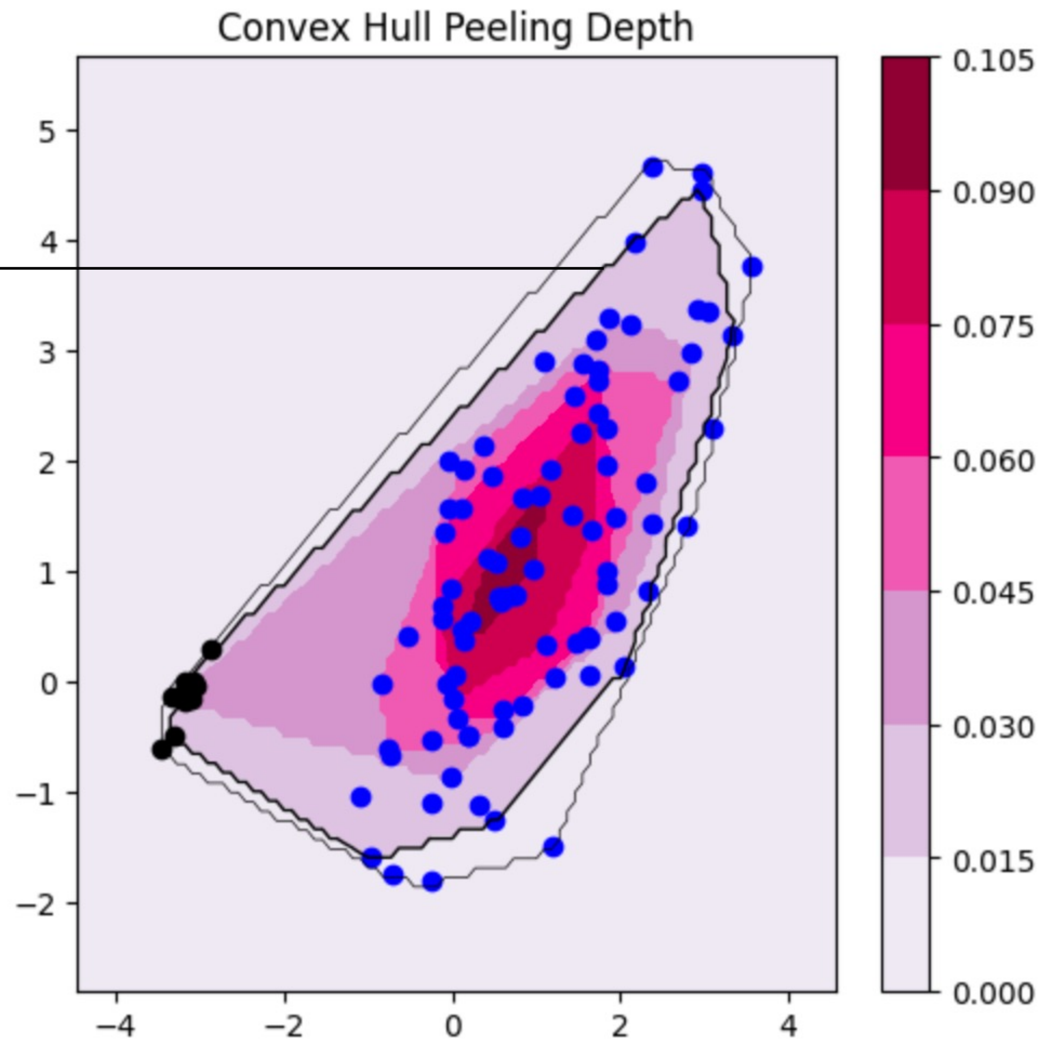
STATISTICAL DATA DEPTH – BIVARIATE DEMO

Outermost convex hull;
Corresponds to data depth ~ 0



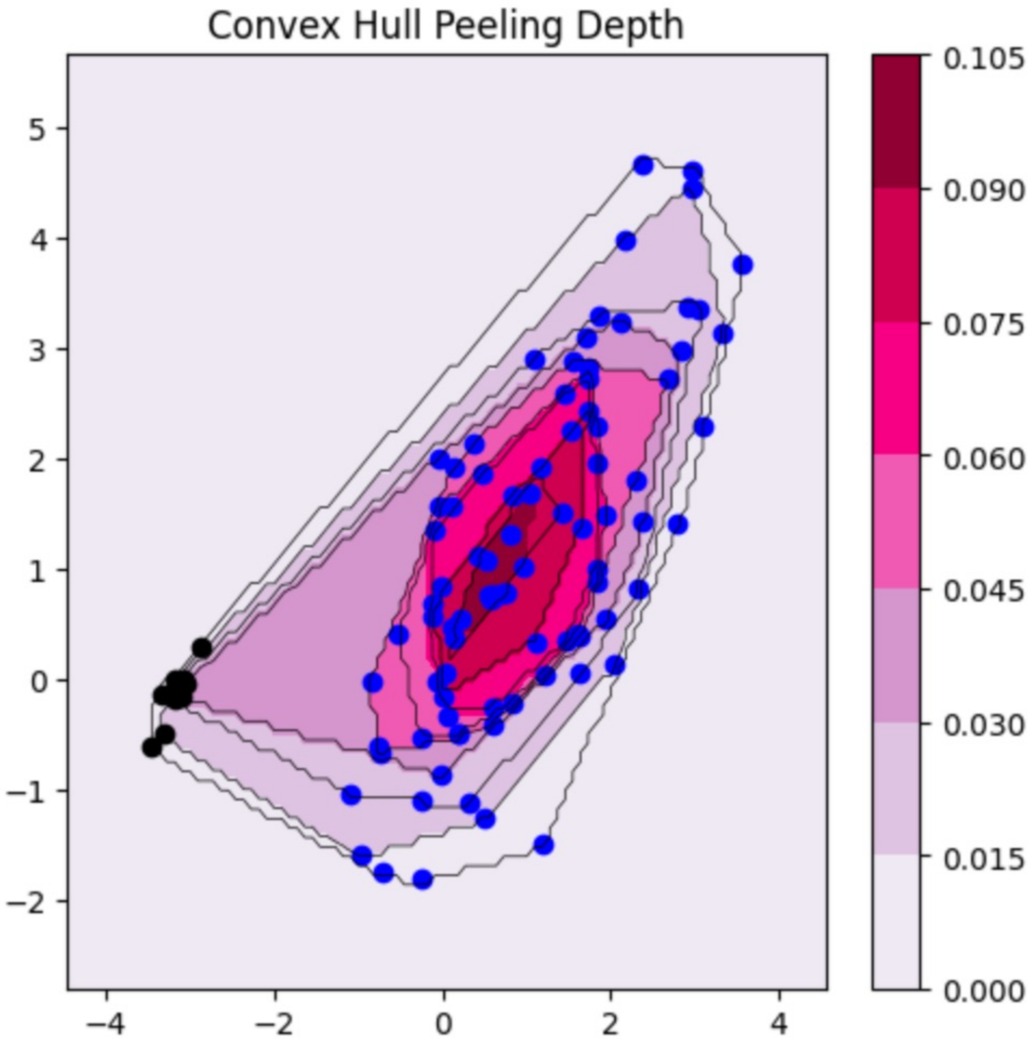
STATISTICAL DATA DEPTH – BIVARIATE DEMO

Next convex hull;
Data depth is now ~ 0.15



STATISTICAL DATA DEPTH - BIVARIATE DEMO

At each level, we construct a new convex hull and assign data depth values accordingly.



STATISTICAL DATA DEPTH

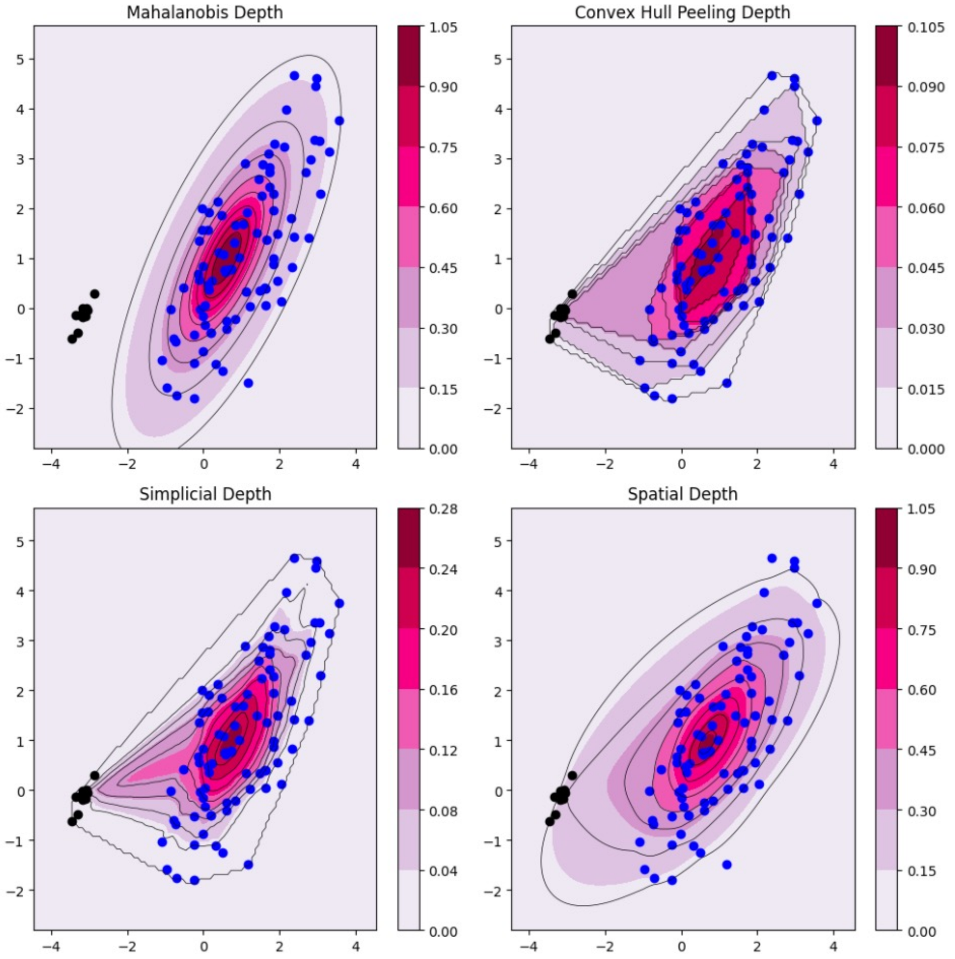
There are many definitions of data depth:

- Mahalanobis depth (Mahalanobis, 1936)
- Convex hull peeling depth (Barnett, 1976; Eddy, 1981)
- Projection depth (Stahel, 1981; Donoho, 1982)
- Simplicial volume depth (Oja, 1983)
- Simplicial depth (Liu, 1990)
- Majority depth (Singh, 1991)
- Zonoid depth (Koshevoy and Mosler, 1997)
- Regression Depth (Rousseeuw and Hubert, 1999)
- L_p-depth (Zuo and Serfling, 2000)
- Spatial depth (Serfling, 2002)
- Expected convex hull depth (Cascos, 2007)
- Geometrical depth (Dyckerhoff and Mosler, 2011)
- Lens depth (Liu and Modarres, 2011)

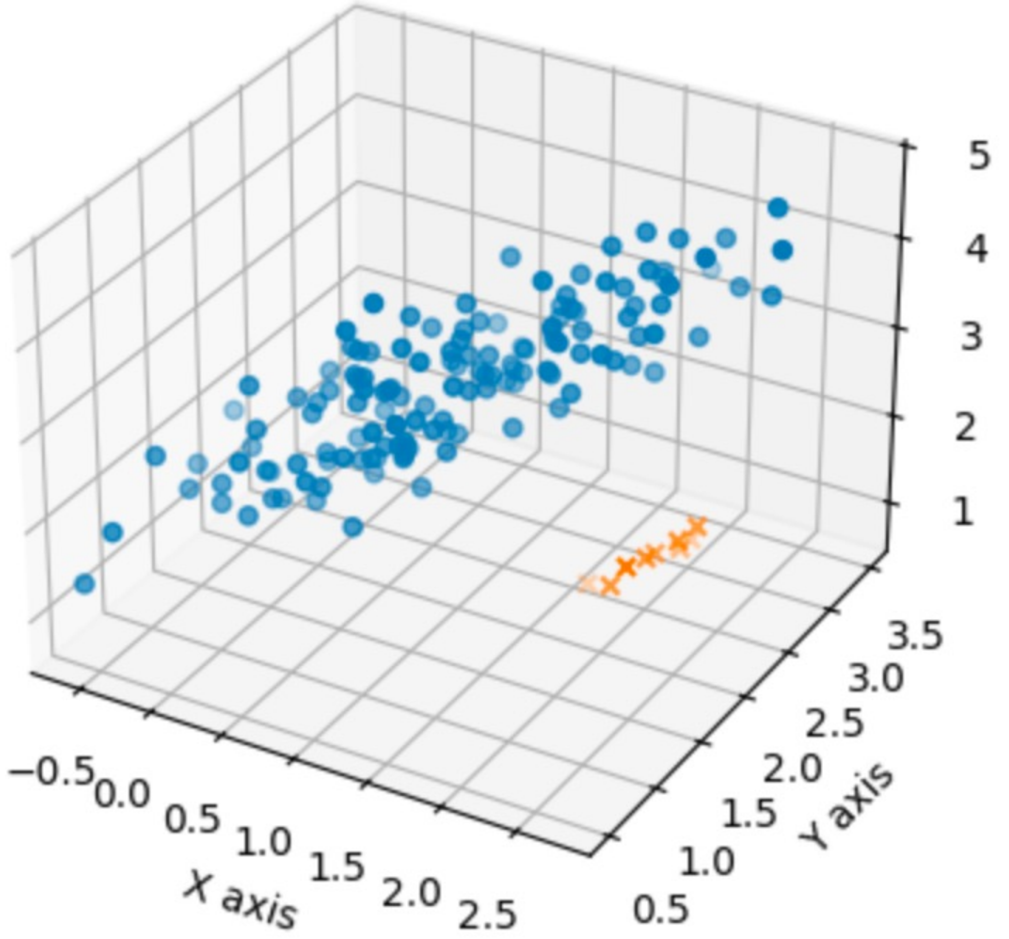
STATISTICAL DATA DEPTH

There are many definitions of data depth:

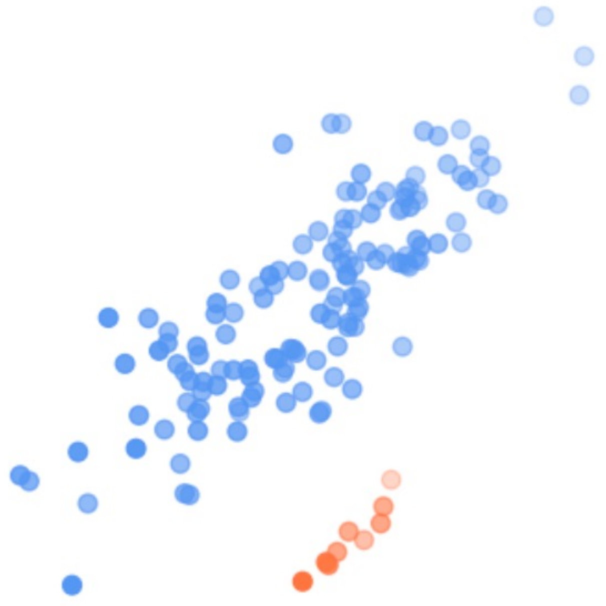
- Mahalanobis depth (Mahalanobis, 1936)
- Convex hull peeling depth (Barnett, 1976; Eddy, 1981)
- Projection depth (Stahel, 1981; Donoho, 1982)
- Simplicial volume depth (Oja, 1983)
- Simplicial depth (Liu, 1990)
- Majority depth (Singh, 1991)
- Zonoid depth (Koshevoy and Mosler, 1997)
- Regression Depth (Rousseeuw and Hubert, 1999)
- L_p-depth (Zuo and Serfling, 2000)
- Spatial depth (Serfling, 2002)
- Expected convex hull depth (Cascos, 2007)
- Geometrical depth (Dyckerhoff and Mosler, 2011)
- Lens depth (Liu and Modarres, 2011)



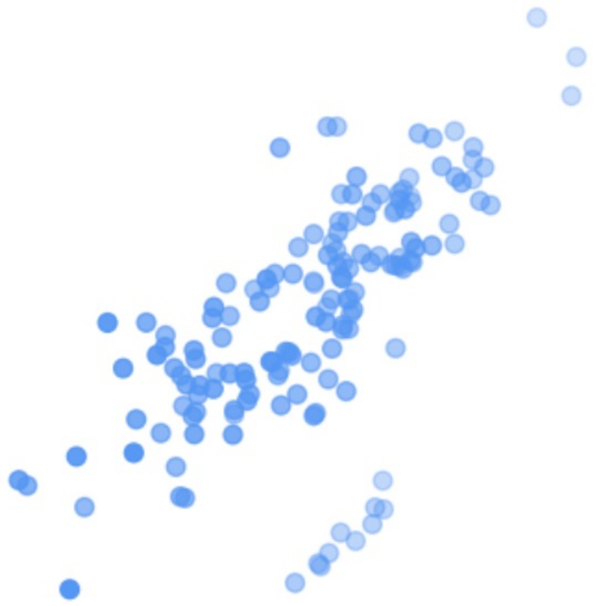
STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



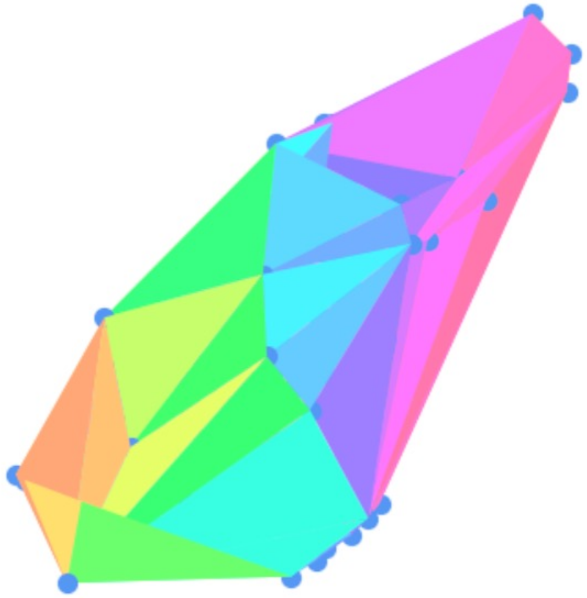
STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

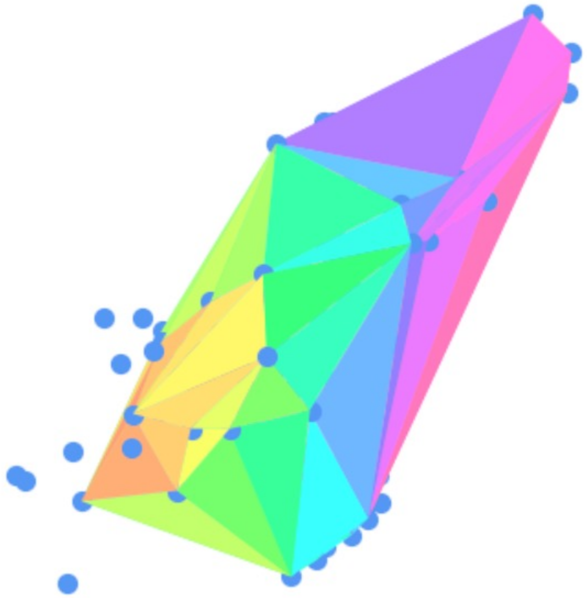


STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

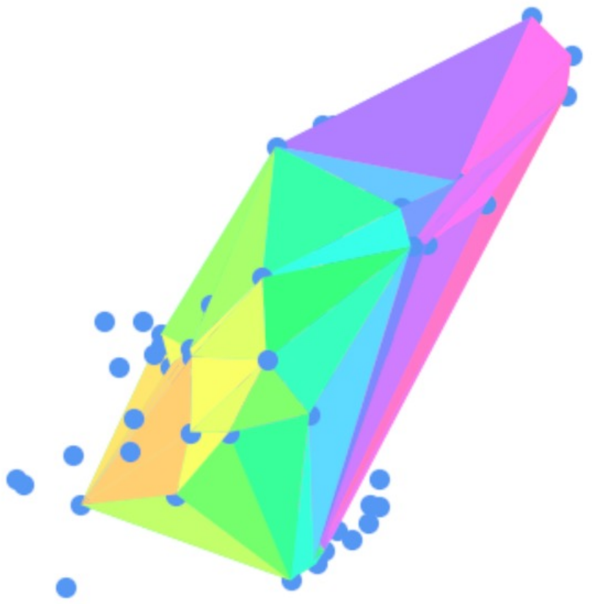


STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

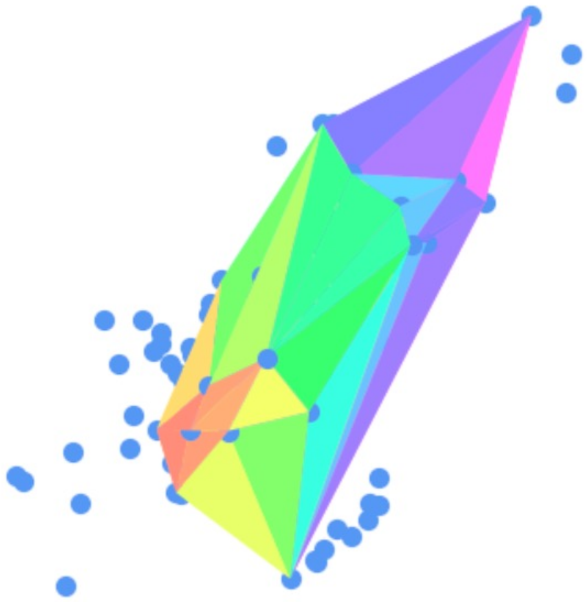


STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

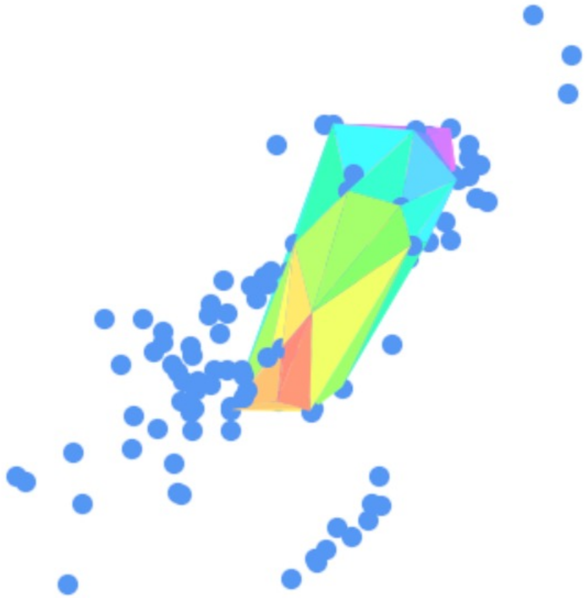
STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



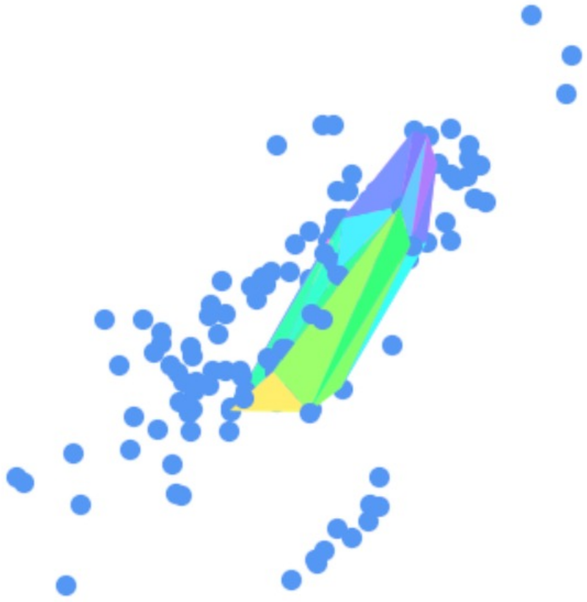
STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

STATISTICAL DATA DEPTH – MULTIVARIATE DEMO

STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



STATISTICAL DATA DEPTH – MULTIVARIATE DEMO



STATISTICAL DATA DEPTH - CHALLENGES

- High Dimensionality
- Scalability
- Robustness
- Choice of Depth Measure
- Non-Euclidean Data
- Interpretability
- Computation of Depth Regions
- Integration with Machine Learning Models

CURRENT CHALLENGES

Out of Distribution Detection

- Defining “Out-of-Distribution”
- High Dimensionality
- Computation Costs
- Domain Shift
- Model Calibration
- Noise and Outliers
- Transferability
- Evaluation Metrics

Statistical Data Depth

- High Dimensionality
- Scalability
- Robustness
- Choice of Depth Measure
- Non-Euclidean Data
- Interpretability
- Computation of Depth Regions
- Integration with Machine Learning Models

CURRENT CHALLENGES

Out of Distribution Detection

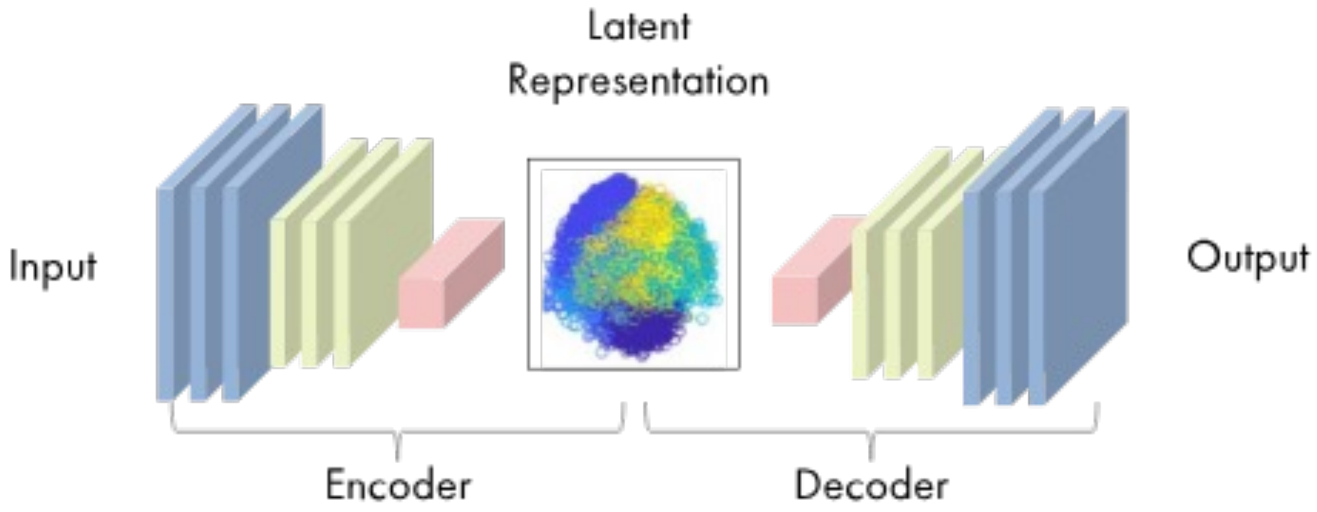
- Defining “Out-of-Distribution”
- High Dimensionality
- Computation Costs
- Domain Shift
- Model Calibration
- Noise and Outliers
- Transferability
- Evaluation Metrics

Statistical Data Depth

- High Dimensionality
- Scalability
- Robustness
- Choice of Depth Measure
- Non-Euclidean Data
- Interpretability
- Computation of Depth Regions
- Integration with Machine Learning Models

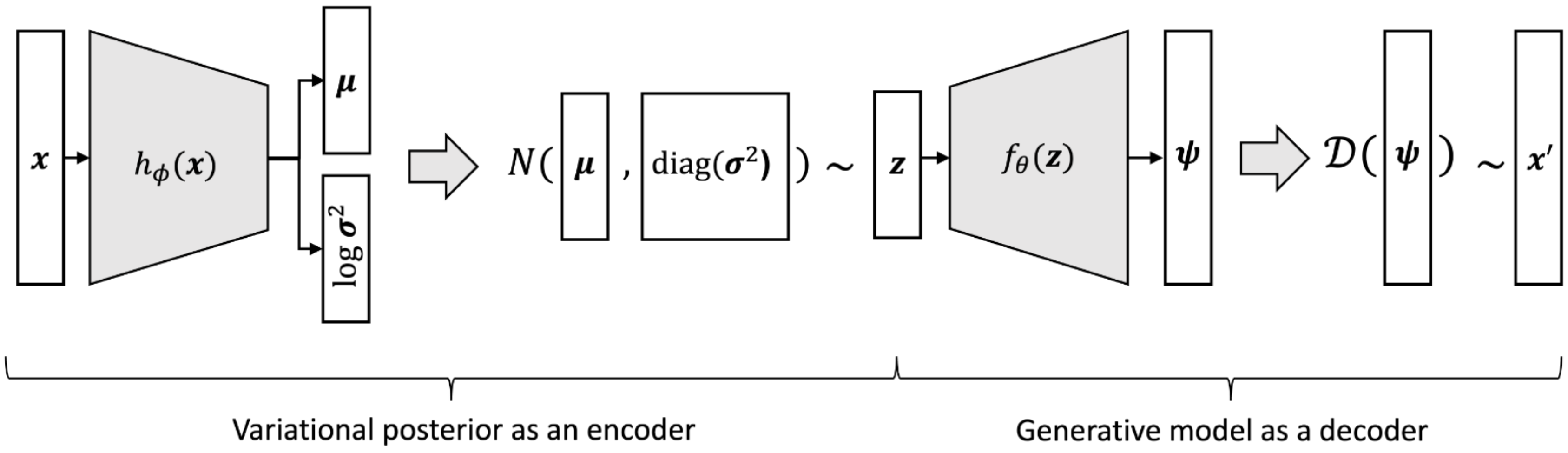
AUTOENCODERS

- A representation learning algorithm
- Learn to map examples to low-dimensional representation



VARIATIONAL AUTOENCODERS

- Variational autoencoders (VAEs), introduced by Kingma and Welling (2013), are a class of probabilistic models that find latent, low-dimensional representations of data.
- VAEs are thus a method for performing dimensionality reduction to reduce data down to their intrinsic dimensionality.

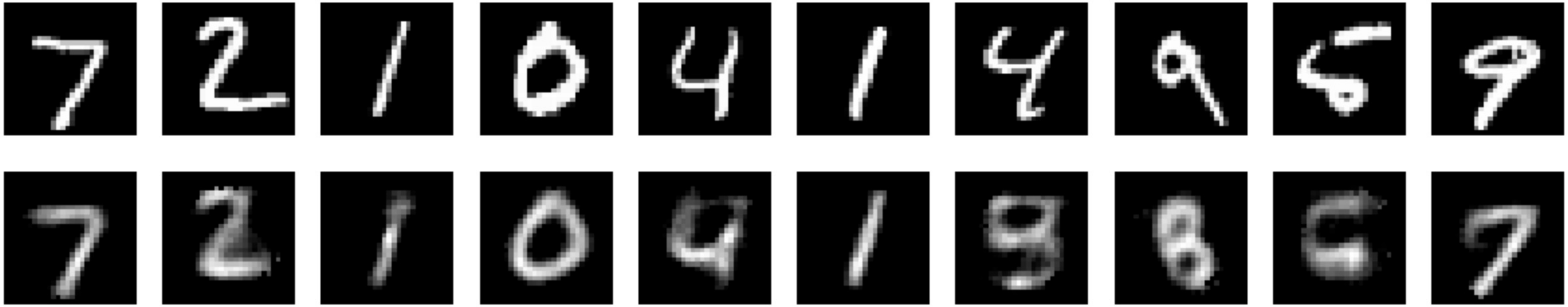


VARIATIONAL AUTOENCODERS – DEMO

- Encoder with two linear layers that produce the mean and log-variance of the latent variables
- Reparameterization trick to ensure differentiability when sampling latent variables.

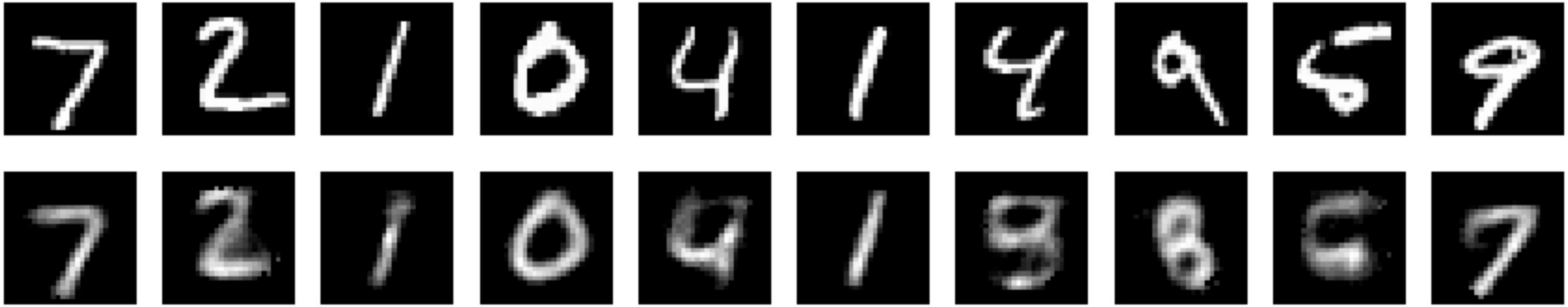
VARIATIONAL AUTOENCODERS – DEMO

- Encoder with two linear layers that produce the mean and log-variance of the latent variables
- Reparameterization trick to ensure differentiability when sampling latent variables.



VARIATIONAL AUTOENCODERS – DEMO

- Encoder with two linear layers that produce the mean and log-variance of the latent variables
- Reparameterization trick to ensure differentiability when sampling latent variables.

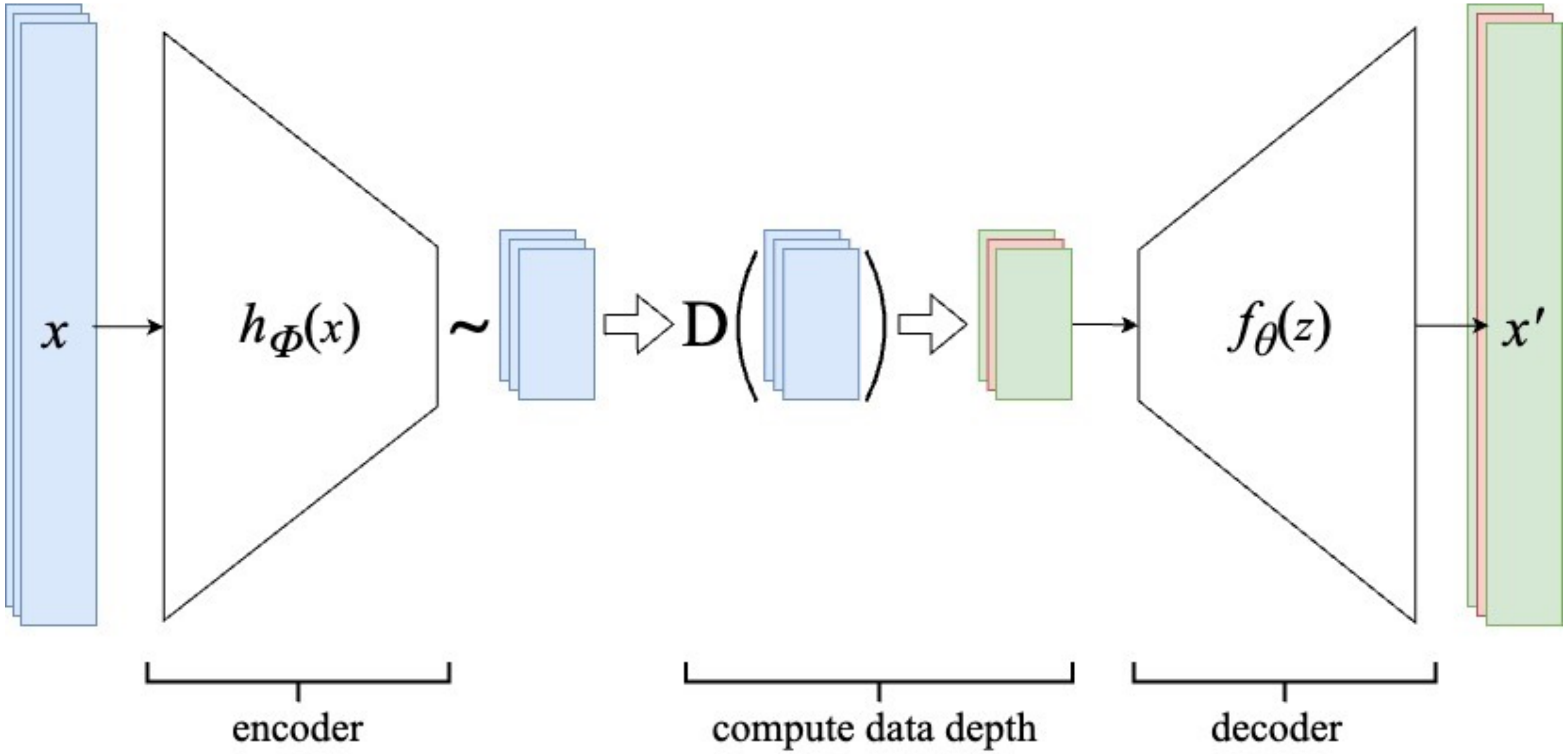


- A combination of reconstruction loss (MSE) and KL divergence to regularize the latent space
- VAE's effectiveness in data compression and latent space representation

VARIATIONAL AUTOENCODERS – KEY ADVANTAGES

- Data generation
- Control of Latent Space
- Modelling Complex Distributions

SOLUTION OVERVIEW



METHODOLOGY

DATASET REVIEW

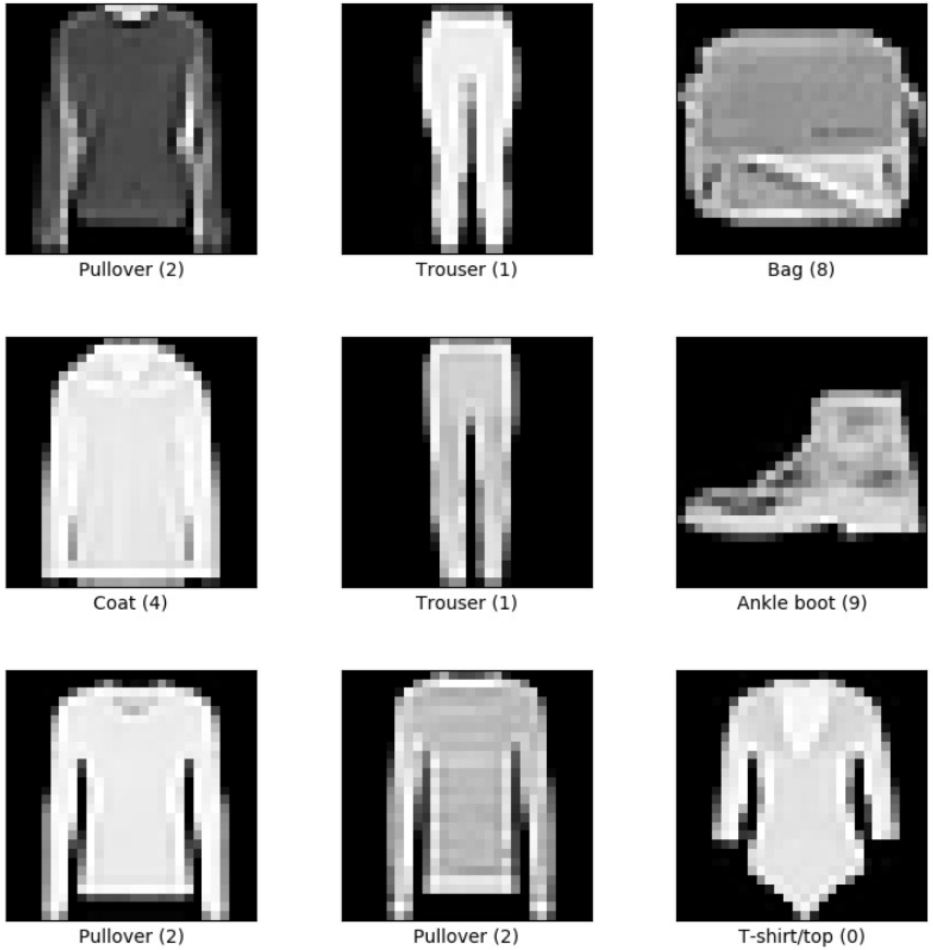
Fashion MNIST Dataset with

Split	Examples
test	10,000
train	60,000

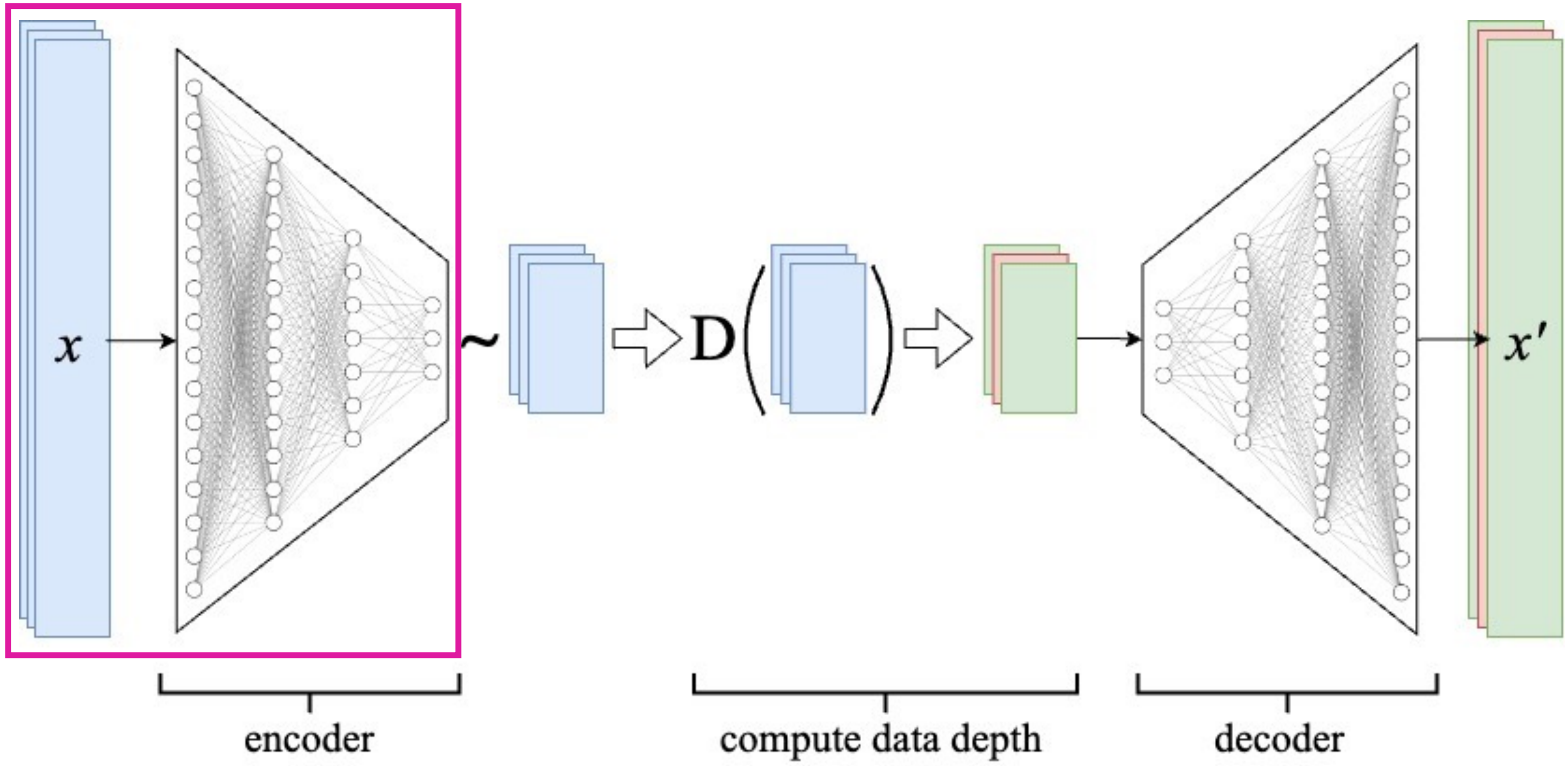
where each example has

```
FeaturesDict({
    'image': Image(shape=(28, 28, 1), dtype=uint8),
    'label': ClassLabel(shape=(), dtype=int64, num_classes=10),
})
```

and each label value corresponds to:



SOLUTION OVERVIEW



We begin by training an encoder on the training dataset of 60000 points to reduce dimensionality.

VARIATIONAL AUTOENCODERS – DEMO

```
class VAE(nn.Module):
    def __init__(self, x_dim, hidden_dim1, hidden_dim2, z_dim=10):
        super(VAE, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(x_dim, hidden_dim1),
            nn.ReLU(),
            nn.Linear(hidden_dim1, hidden_dim2),
            nn.ReLU(),
            nn.Linear(hidden_dim2, z_dim * 2)
        )

        self.decoder = nn.Sequential(
            nn.Linear(z_dim, hidden_dim2),
            nn.ReLU(),
            nn.Linear(hidden_dim2, hidden_dim1),
            nn.ReLU(),
            nn.Linear(hidden_dim1, x_dim),
            nn.Sigmoid()
        )
```

VARIATIONAL AUTOENCODERS – DEMO

- Reparametrize function to sample from the latent space by introducing stochasticity.

```
class VAE(nn.Module):
    ...
    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        h = self.encoder(x)
        mu, logvar = torch.chunk(h, 2, dim=1)
        z = self.reparameterize(mu, logvar)
        return self.decoder(z), z, mu, logvar

# Loss function
def loss_function(reconx, x, mu, logvar):
    BCE = nn.functional.binary_cross_entropy(reconx, x, reduction='sum')
    KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
    return BCE + KLD
```

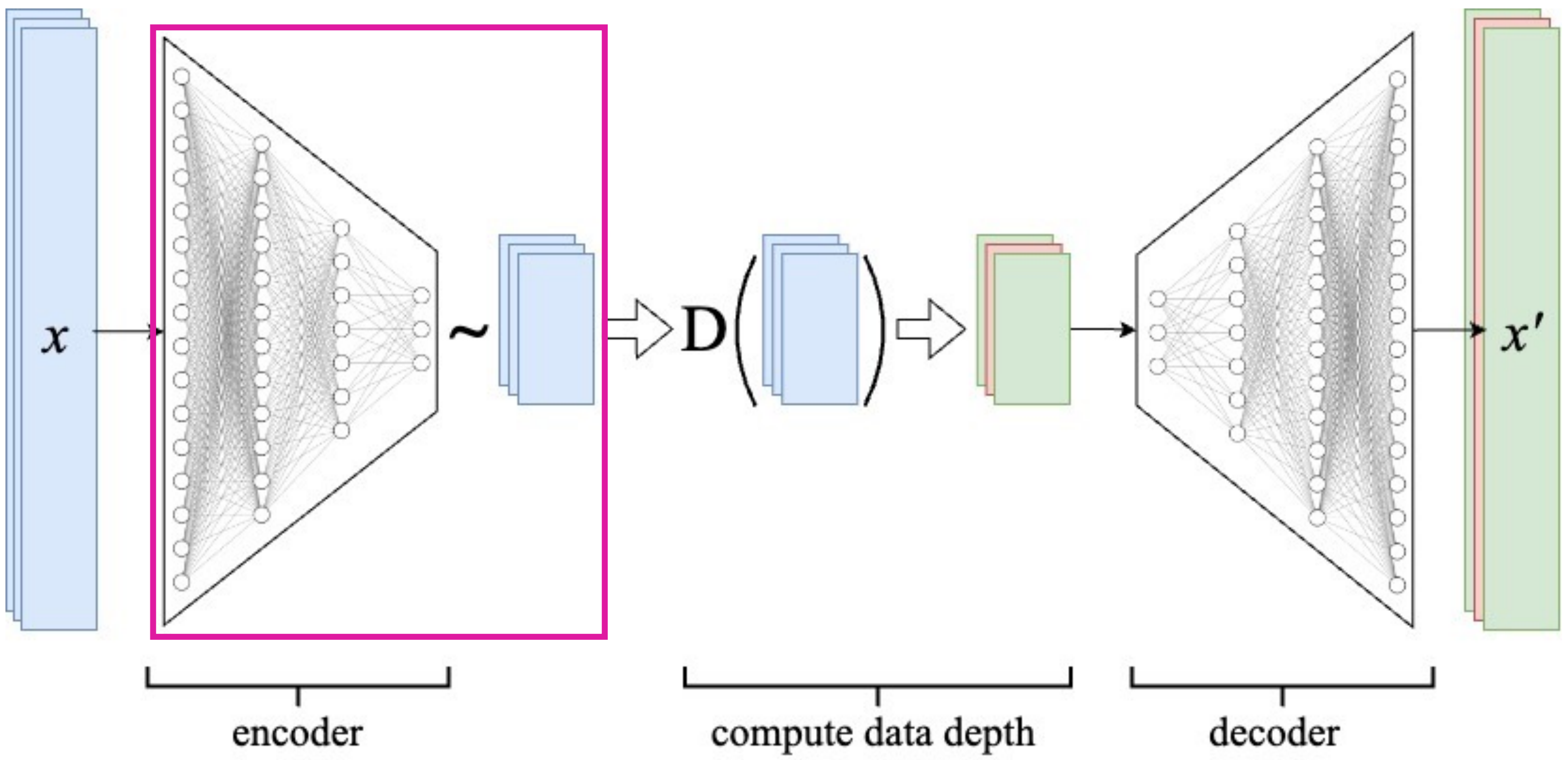
VARIATIONAL AUTOENCODERS – DEMO

- Reparametrize function to sample from the latent space by introducing stochasticity.
- The loss function uses a combination of the standard reconstruction loss and KL Divergence loss.

$$\text{loss}_{\text{VAE}}(\varphi, \theta) = -\sum_{i=1}^{n} \mathrm{E}_{z_i \sim q_\varphi(z_i \mid x_i)} \left[\log p_\theta(x_i \mid z_i)\right] - \mathrm{KL}(q_\varphi(z_i \mid x_i) \,||\, p(z_i))$$

```python
class VAE(nn.Module):
    ...
    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def forward(self, x):
        h = self.encoder(x)
        mu, logvar = torch.chunk(h, 2, dim=1)
        z = self.reparameterize(mu, logvar)
        return self.decoder(z), z, mu, logvar

# Loss function
def loss_function(reconx, x, mu, logvar):
    BCE = nn.functional.binary_cross_entropy(reconx, x, reduction='sum')
    KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
    return BCE + KLD
```

SOLUTION OVERVIEW



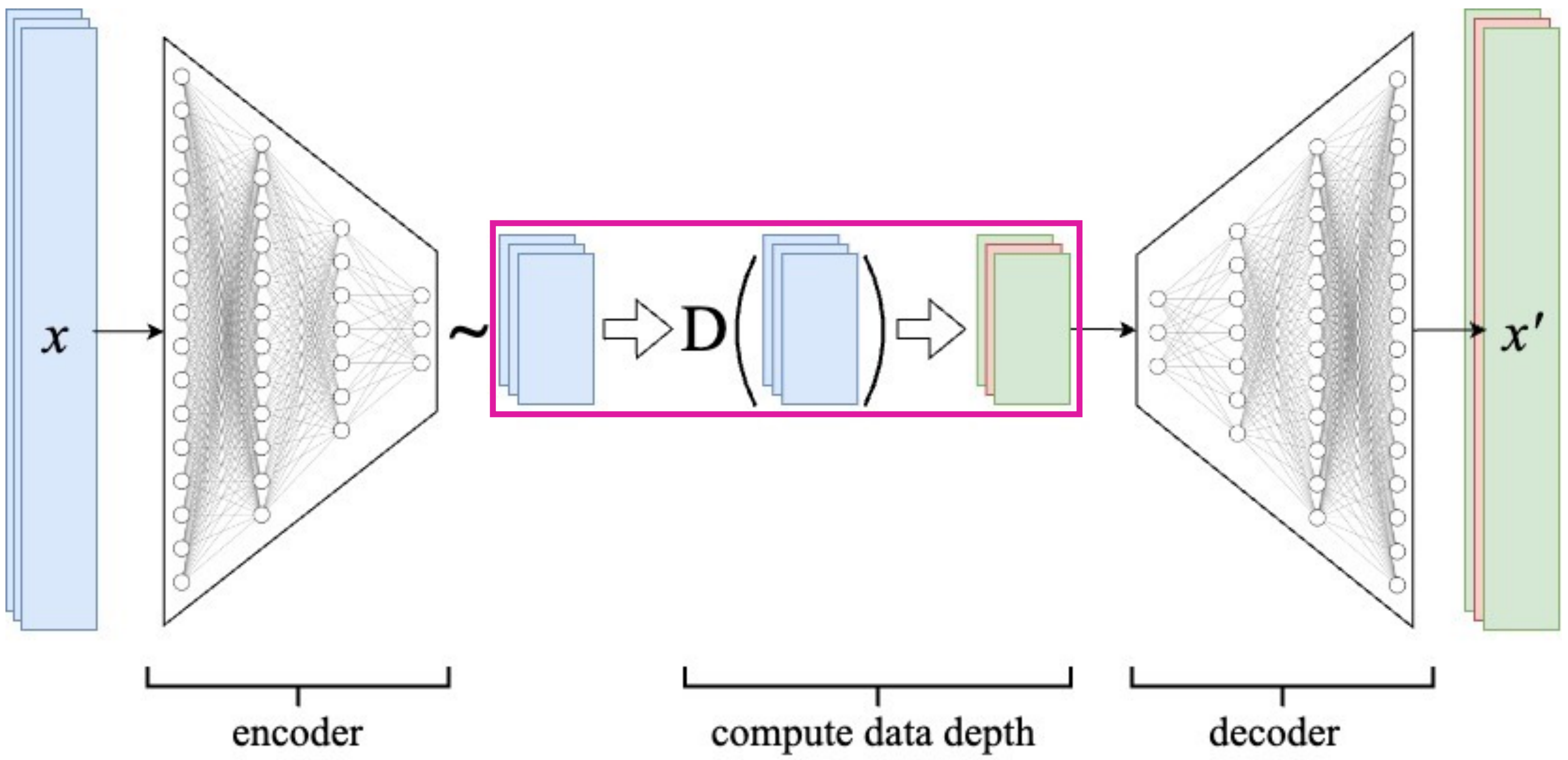
Using the trained encoder, we obtain the latent space representation for each point in the test set.

VARIATIONAL AUTOENCODERS – DEMO

- Evaluate the trained model on the test dataset to obtain the latent representations and store them in a numpy array.

```
trained_model.eval()

# initialize lists to store latent space and their true labels
latent_space = []
labels = []
sample_no = 1 # in case MAX_STEPS is defined
with torch.no_grad():
    for batch in fashion_mnist_test:
        if MAX_STEPS != None and sample_no > MAX_STEPS :
            break
        x, y = batch
        x = x.view(-1, x.size(0))
        _, z, _, _ = trained_model(x)
        latent_space.append(z.cpu().numpy())
        labels.append(torch.tensor(y).cpu().numpy())
        sample_no += 1

latent_space = np.concatenate(latent_space, axis=0)
labels = np.concatenate([labels], axis=0)
```

SOLUTION OVERVIEW



Finally, use the depth function to compute data depth for each point.

VARIATIONAL AUTOENCODERS – DEMO

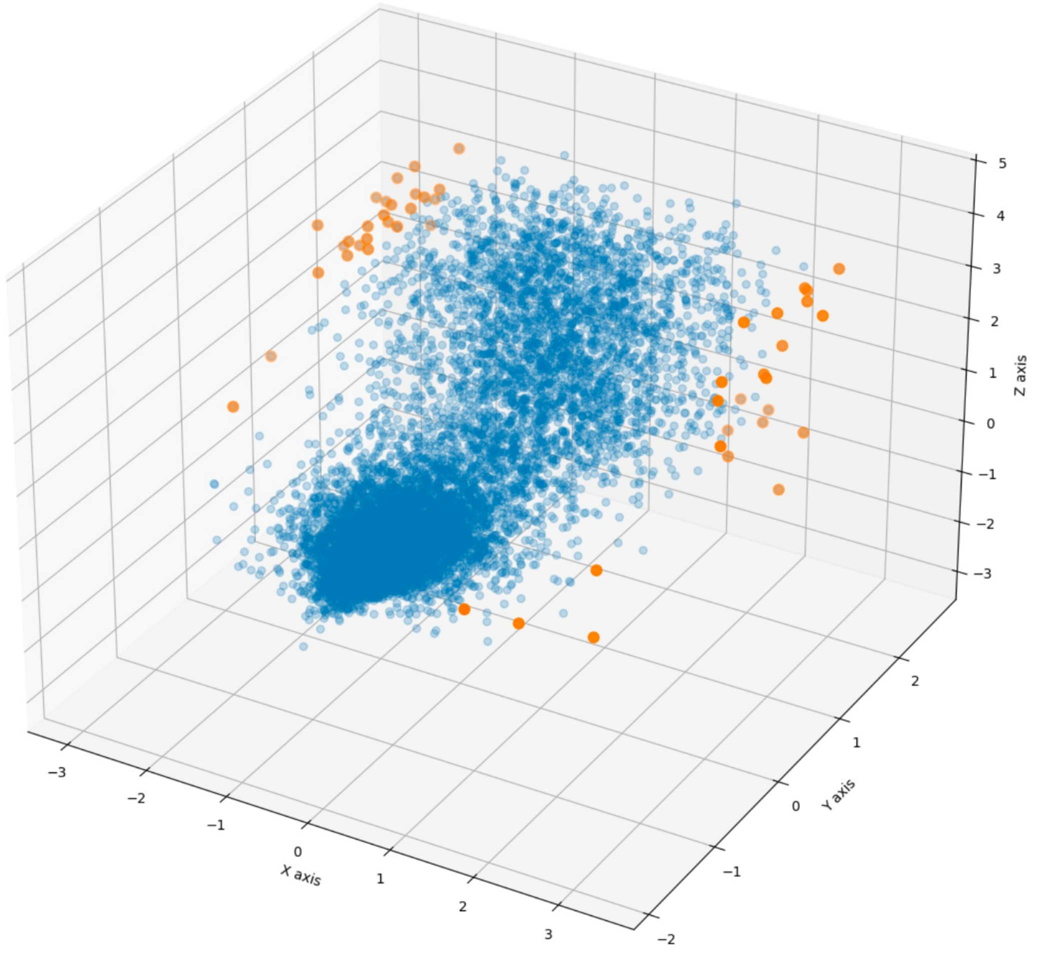
- Multivariate library to compute data depth using spatial depth (polynomial time).
- Set a threshold to classify points as normal or anomalous.

```
# use multivariate library to compute data depth
depths = spatial(latent_space, latent_space)

# set a threshold for anomalies (e.g., top 0.5% furthest points)
threshold = np.percentile(depths, 0.5)

# get anomalies
anomalies = depths ⩽ threshold
```

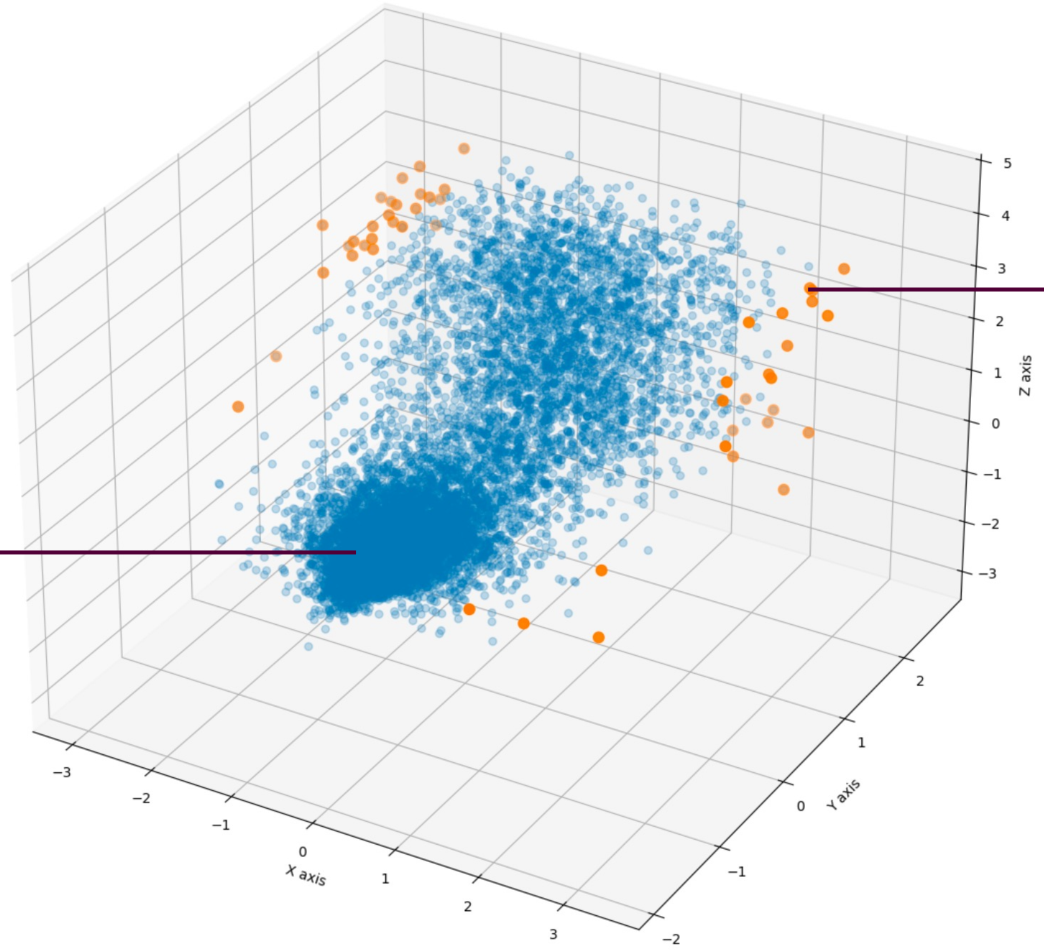
```
Number of points : 10000
Min Data Depth   : 0.03976266539252871
Max Data Depth   : 0.9528853364958756
Std Deviation    : 0.17370466079234456
Threshold        : 0.07019434229670497
```

RESULTS AND ANALYSIS

ANOMALY DISTRIBUTION



ANOMALY DISTRIBUTION



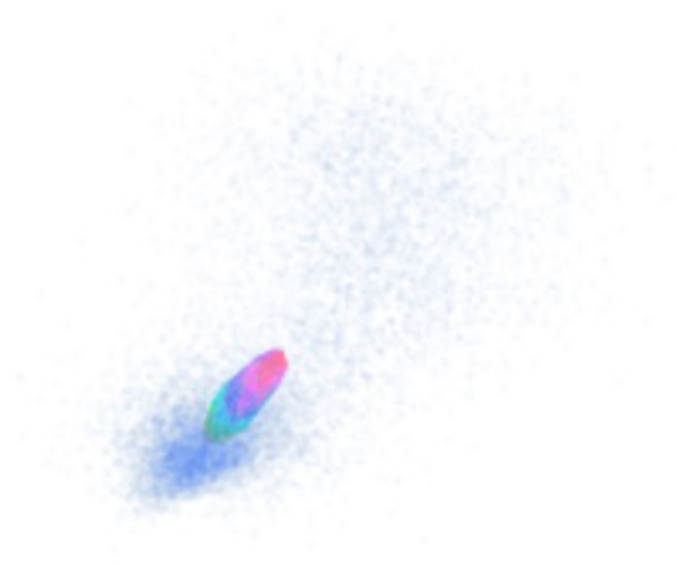
Most anomalies seem to away from the median of the data

The median would be somewhere around here

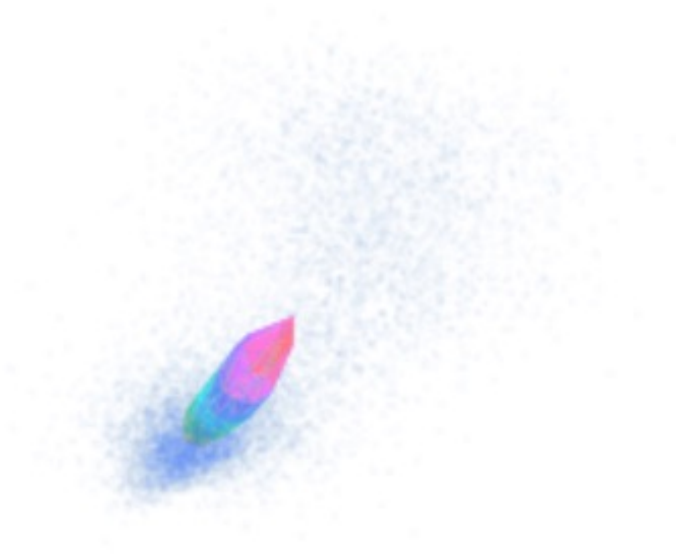
CONVEX HULL PROGRESSION

CONVEX HULL PROGRESSION

CONVEX HULL PROGRESSION

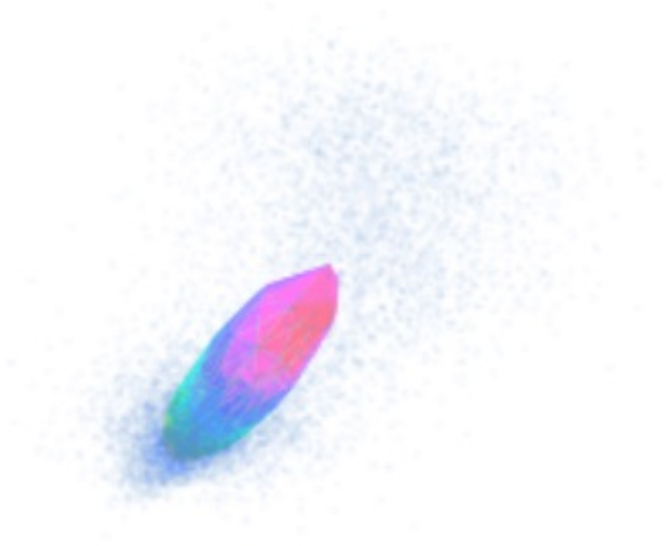


CONVEX HULL PROGRESSION

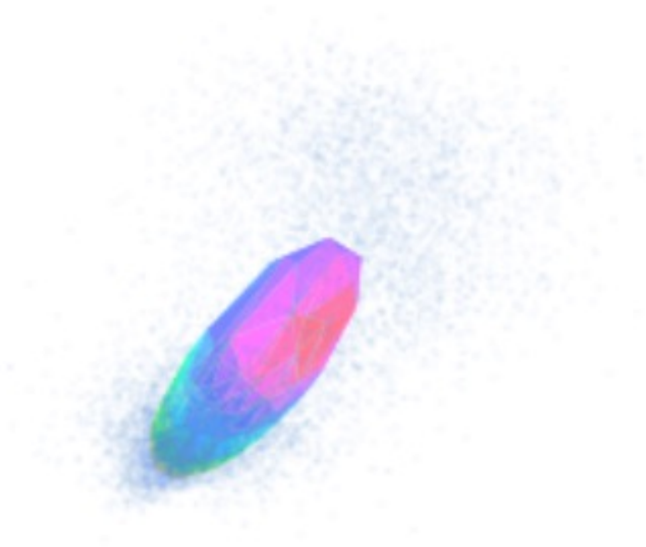


CONVEX HULL PROGRESSION

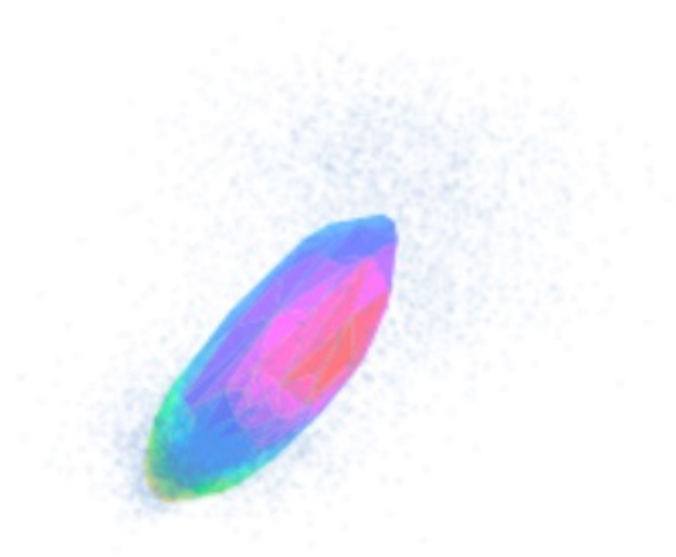
CONVEX HULL PROGRESSION



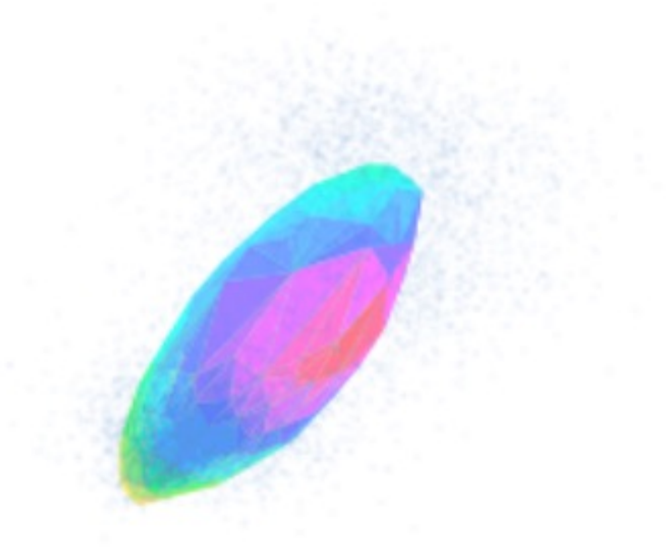
CONVEX HULL PROGRESSION



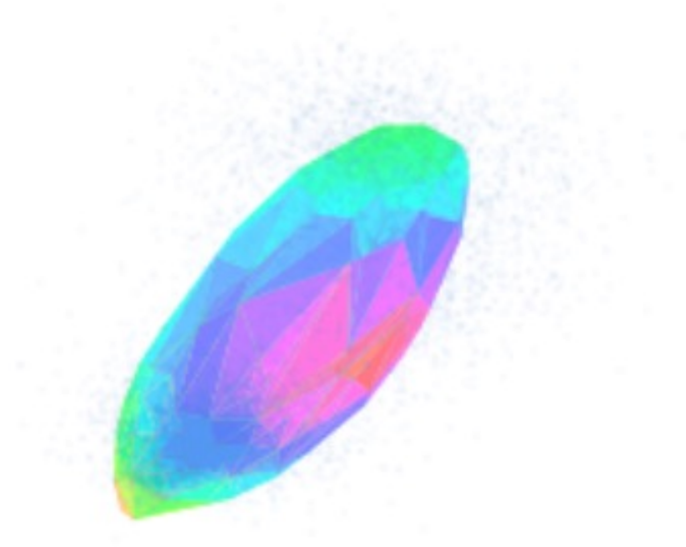
CONVEX HULL PROGRESSION



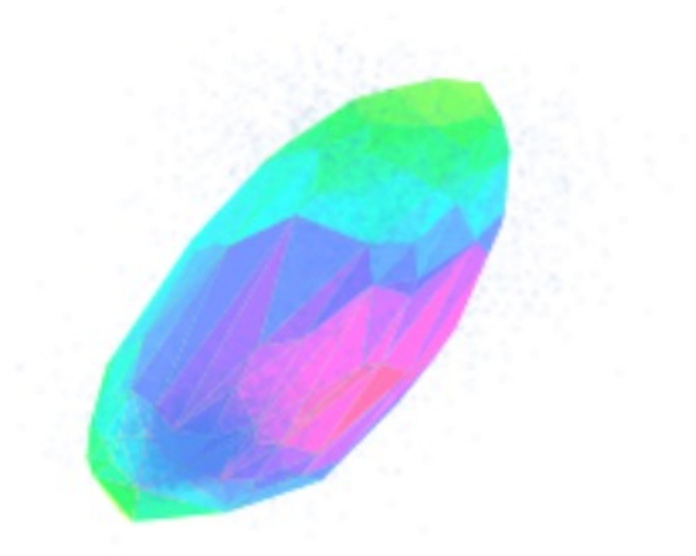
CONVEX HULL PROGRESSION



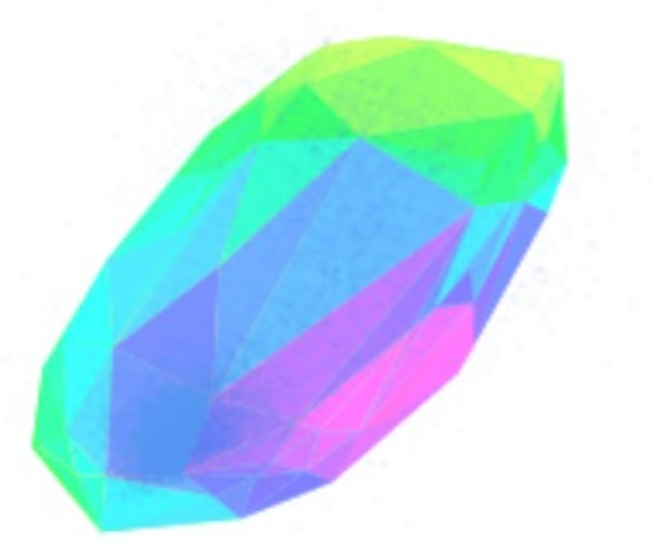
CONVEX HULL PROGRESSION



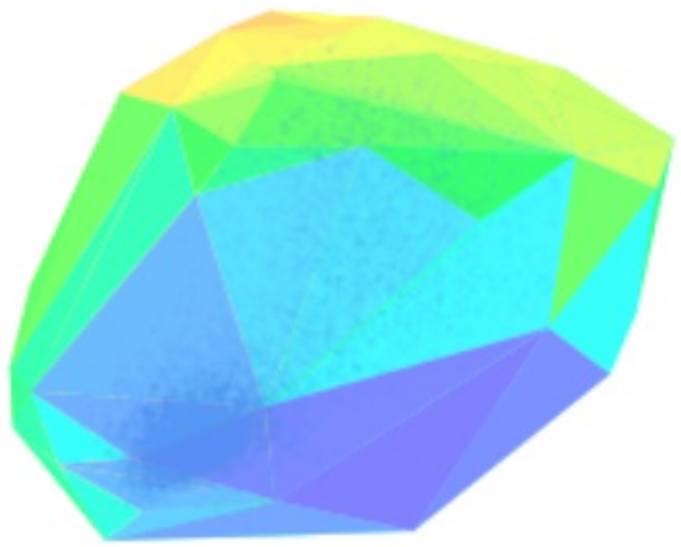
CONVEX HULL PROGRESSION



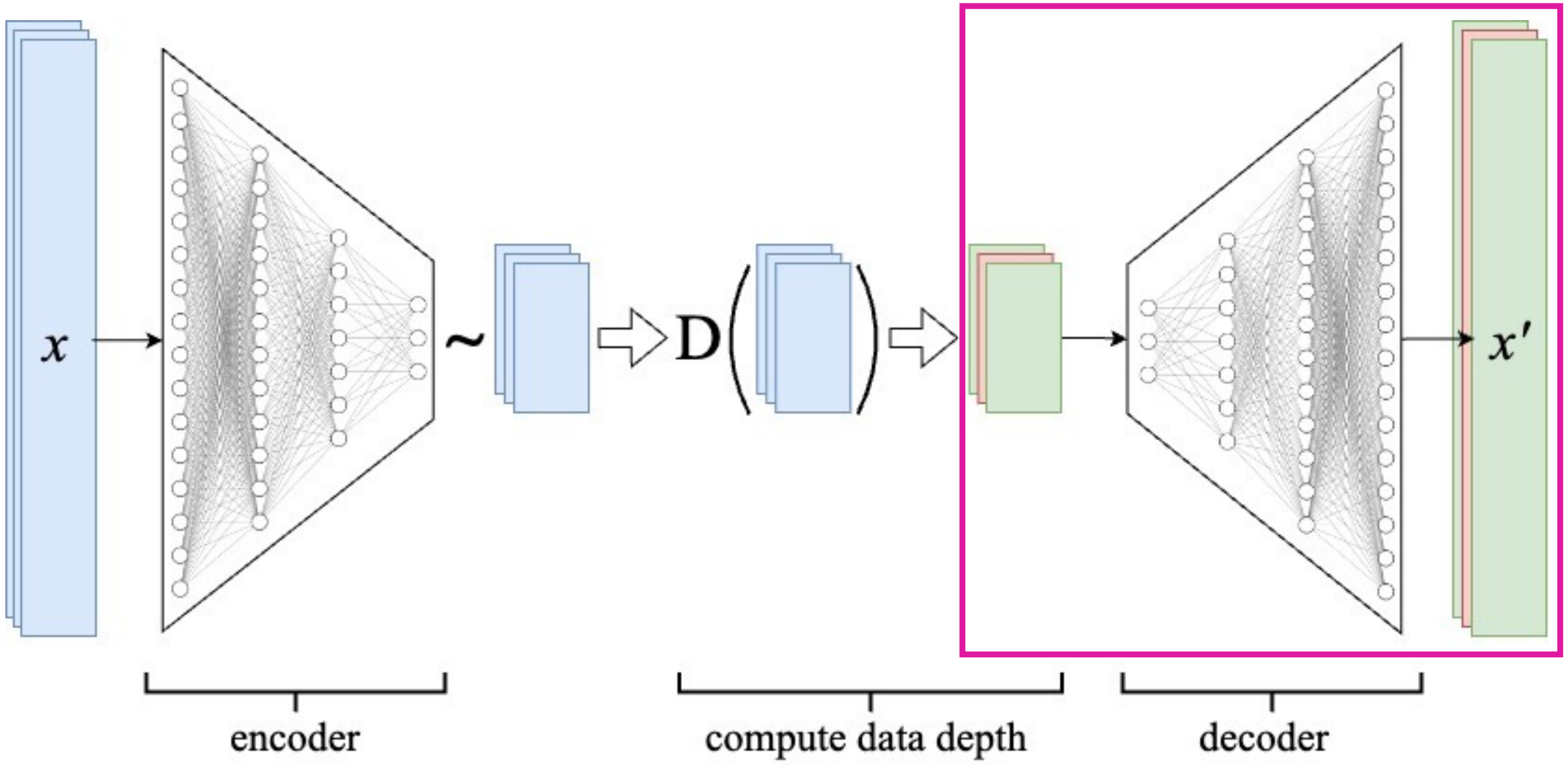
CONVEX HULL PROGRESSION



CONVEX HULL PROGRESSION

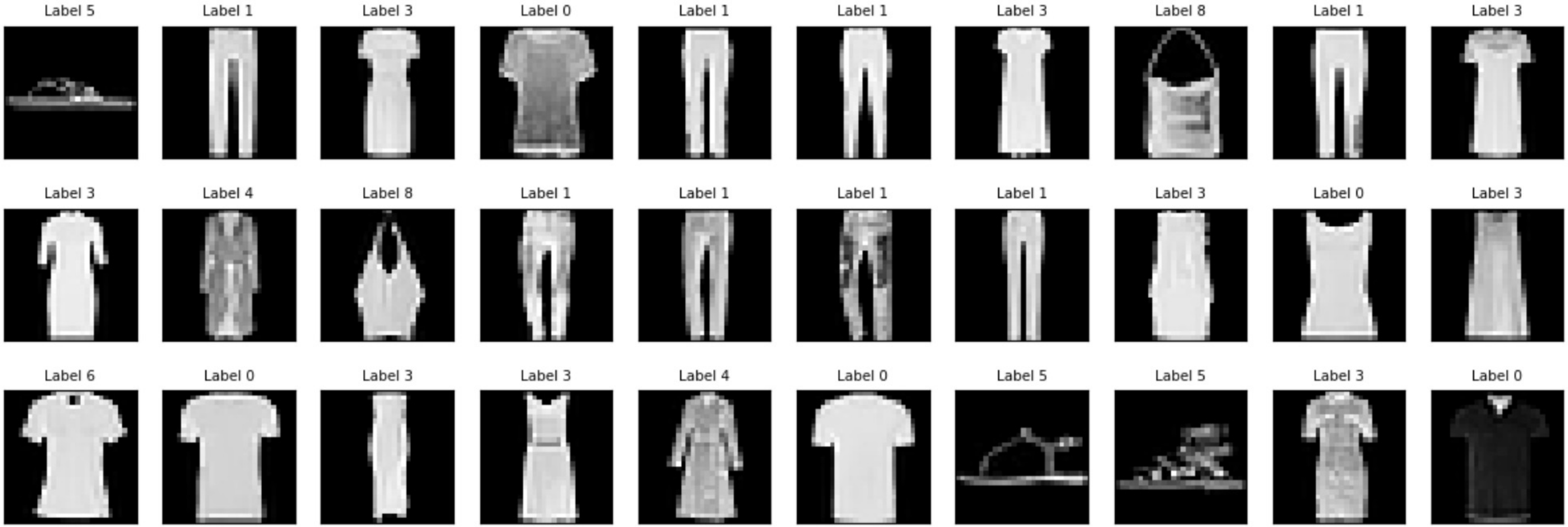


SOLUTION OVERVIEW



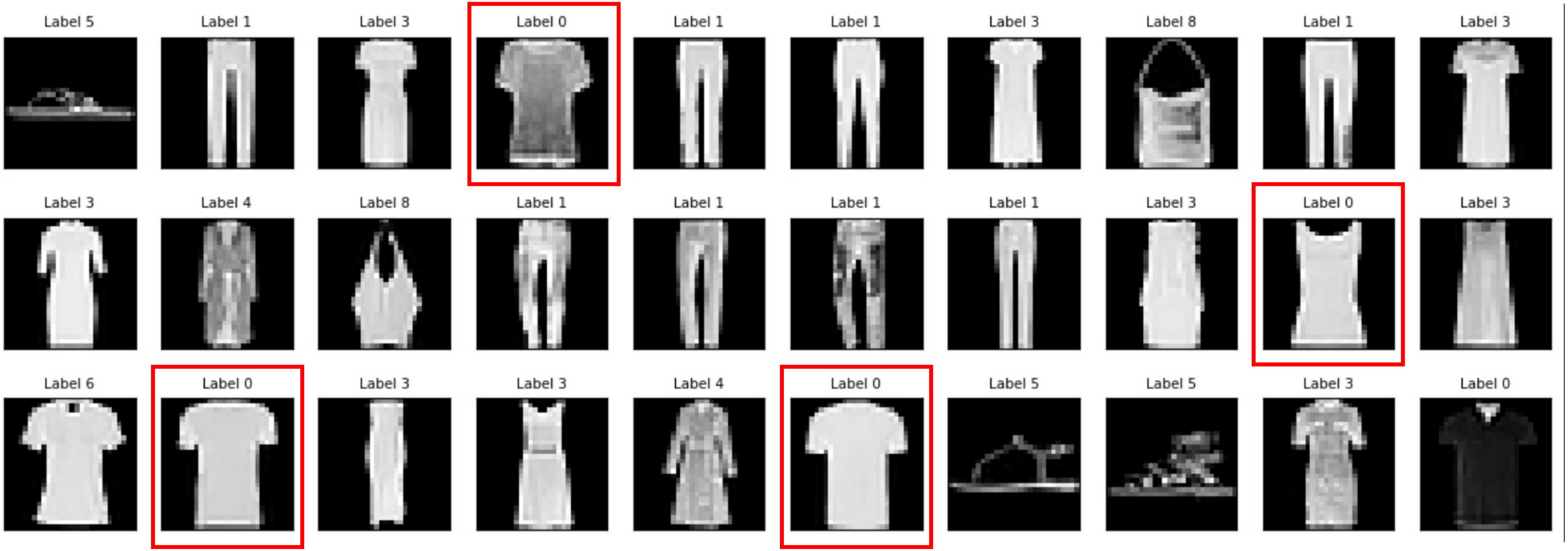
We can also use the decoder to reconstruct the images and check whether they are actually anomalies.

VISUALLY CHECKING ANOMALIES



CONCLUSION

VISUALLY CHECKING ANOMALIES



IMPROVEMENTS

- Hyperparameter Tuning
- Advanced VAE Architectures
- Regularization Techniques
- Alternative Depth Measures
- Hybrid Approaches
- Threshold Optimization
- Quantitative Evaluation
- Error Analysis
- Experiment Tracking

Q/A

APPENDIX

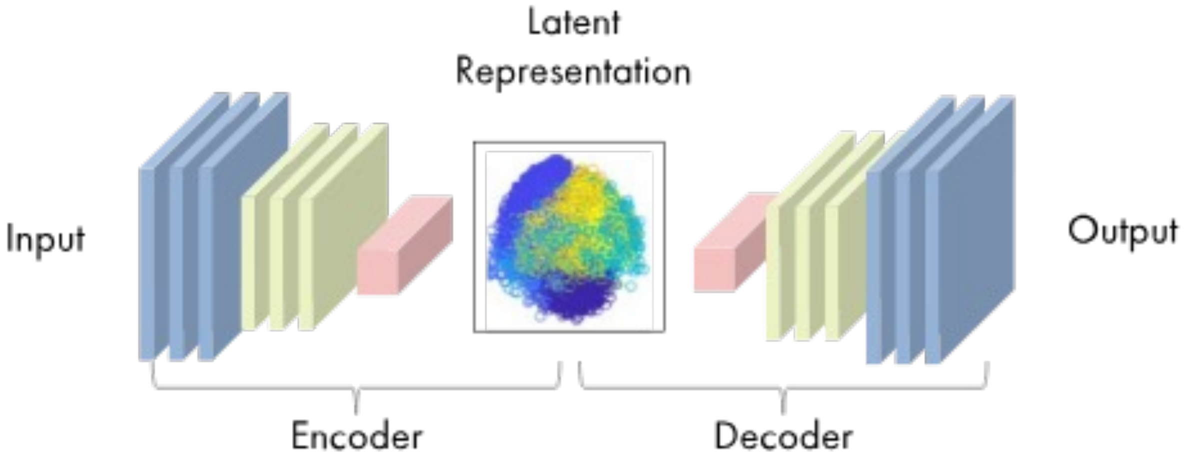
AUTOENCODERS

2 main components :

- Encoder *e(x)*: maps x to low-dimensional representation $\hat{z}$
- Decoder *d($\hat{z}$):* maps $\hat{z}$ to its original representation x

Autoencoder implements $\hat{x} = d(e(x))$

- $\hat{x}$ is the reconstruction of original input x.
- Encoder and decoder learned such that $\hat{z}$ contains as much information about x as needed to reconstruct it.



- Minimize sum of squares of differences between input and prediction: $E = \sum_i (x_i - d(e(x_i)))^2$

GITHUB REPOSITORY

- Please find all details of the implementations and the visualizations here. [https://github.com/ananya-k15/data-depth].
- In case of any issues, contact Ananya Kumar (a327kuma@uwaterloo.ca).

REFERENCES

- A. Varol, M. Salzmann, P. Fua and R. Urtasun, "A constrained latent variable model," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 2248-2255, doi: 10.1109/CVPR.2012.6247934.
- Bernstein, Matthew. Variational Autoencoders - Matthew N. Bernstein. 14 Mar. 2023, https://mbernste.github.io/posts/vae/.
- Mozharovskyi, Pavlo, and Romain Valla. “[2210.02851] Anomaly Detection Using Data Depth: Multivariate Case.” ArXiv.Org, https://arxiv.org/abs/2210.02851.
- Neto, Maria Raquel. “The Concept of Depth in Statistics.” University of Lisbon, fenix.tecnico.ulisboa.pt/downloadFile/395137801290/paper.pdf.
- Xia, Linhan, et al. “[2403.01370] Depth Estimation Algorithm Based on Transformer-Encoder and Feature Fusion.” ArXiv.Org, https://arxiv.org/abs/2403.01370.

UNIVERSITY OF
WATERLOO

FACULTY OF MATHEMATICS