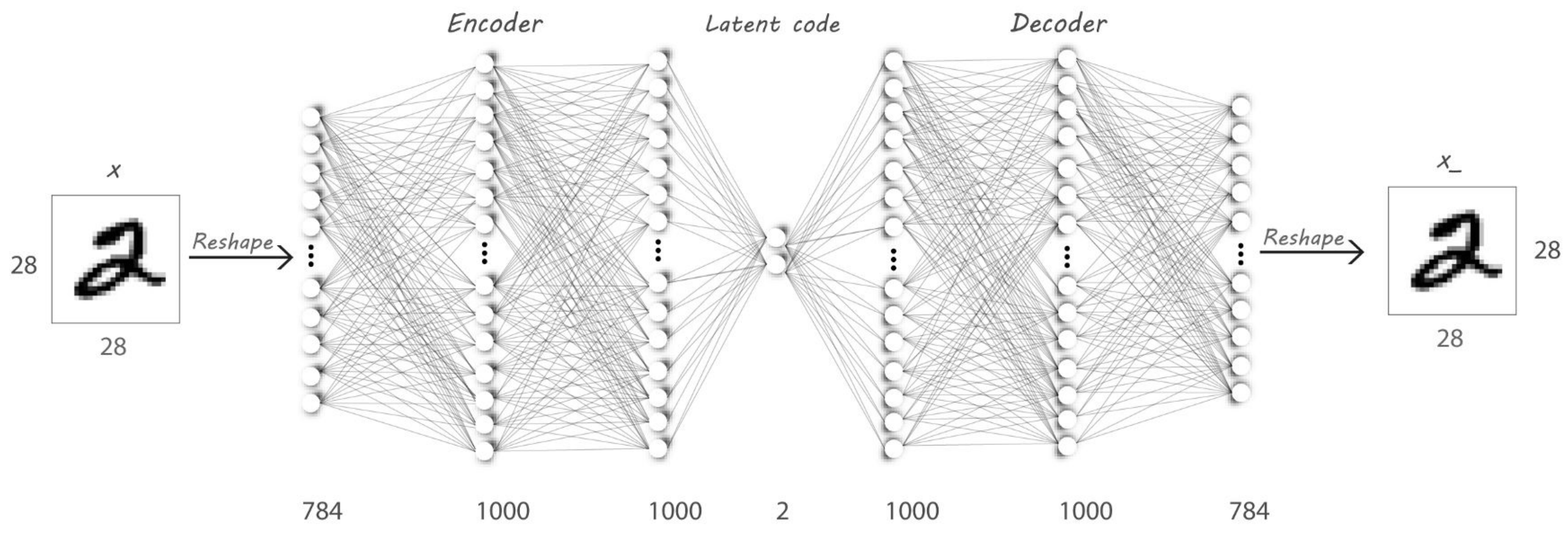
Comparing Autoencoders: Performance Impact of Variational Parameters

***Authors**: Jonas Wacker, Alberto Ibarrondo*
***Tutor**: Maurizio Filippone*

INDEX

1. Autoencoders

Autoencoders

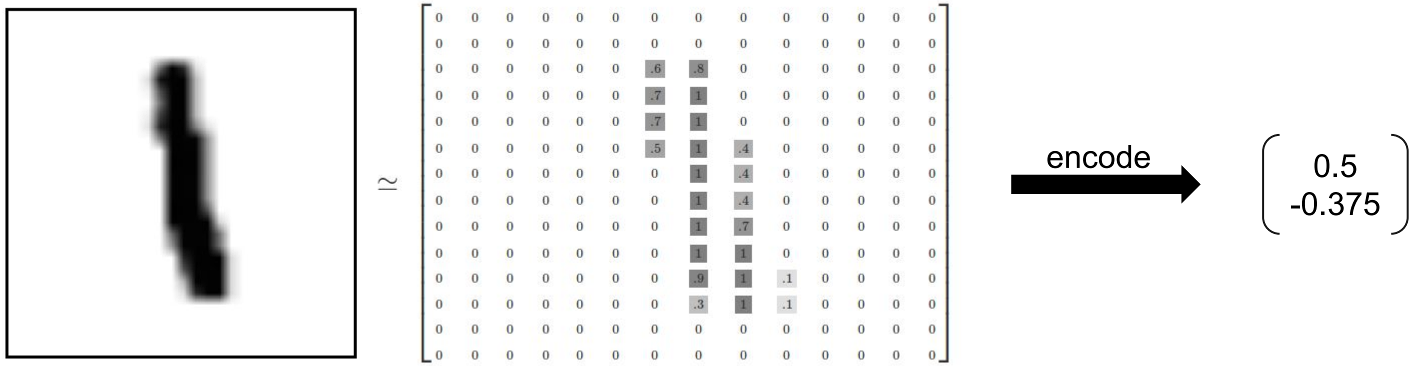


Graphic from
https://towardsdatascience.com/a-wizards-guide-to-adversarial-autoencoders-part-1-autoencoder-d9a5f8795af4

Our dataset: **MNIST** handwritten digits

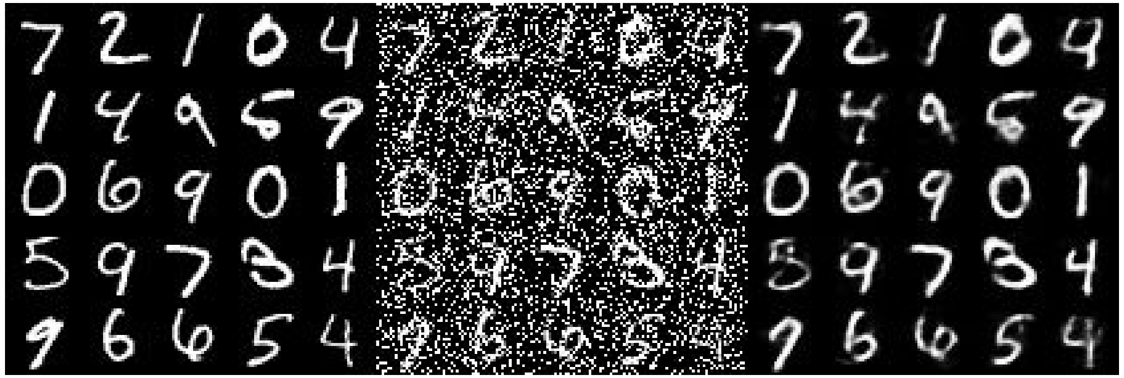
Autoencoders - Applications

Data Compression



Graphic from https://tensorflow.rstudio.com/tensorflow/articles/tutorial_mnist_beginners.html

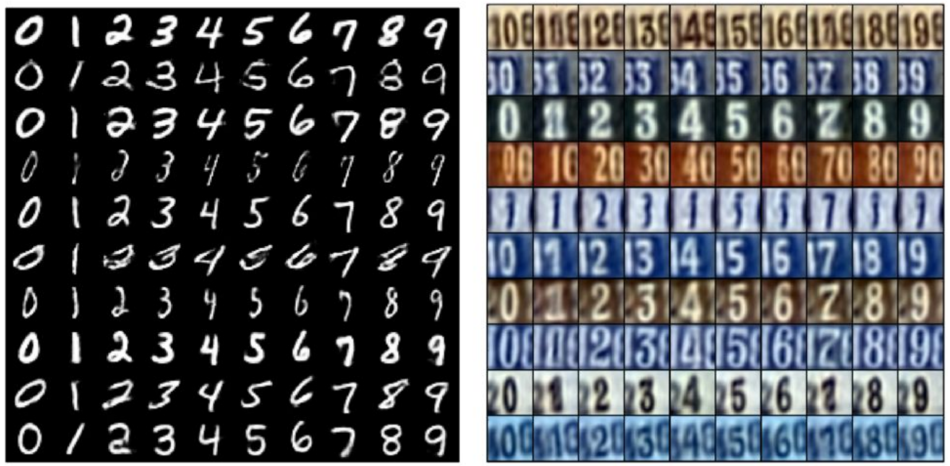
Input Denoising



Graphic from http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model

Autoencoders - Applications

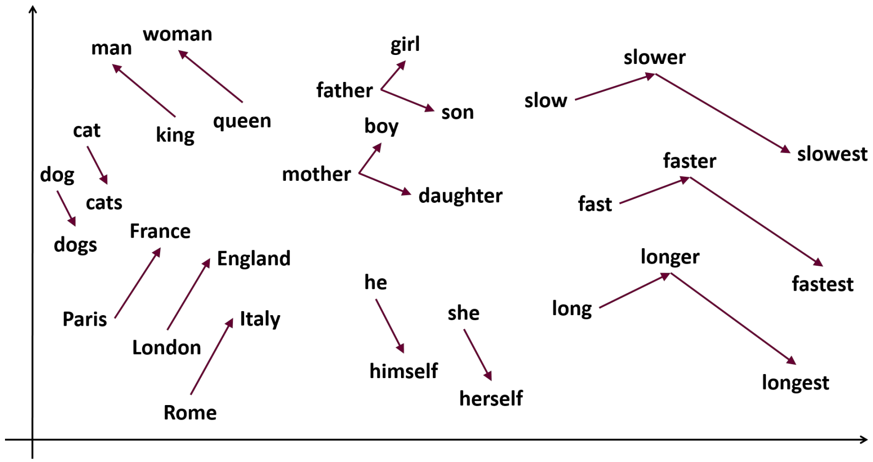
Disentanglement of Content and Style



(a) MNIST (b) SVHN

Alireza Makhzani et al., Adversarial Autoencoders, International Conference on Learning Representations, 2016

Word Embeddings



Graphic from http://www.samyzaf.com

Autoencoders - Training

- Let's treat neural networks as probabilistic models: $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$
- Distribution depends on the data:
  - For regression: Gaussian with least square optimization
  - For **MNIST**, y is between 0 and 1:
    Approximated by **Bernoulli Likelihood** (cross entropy optimization) on pixel level

$$\sum_{n=1}^{N} \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right]$$

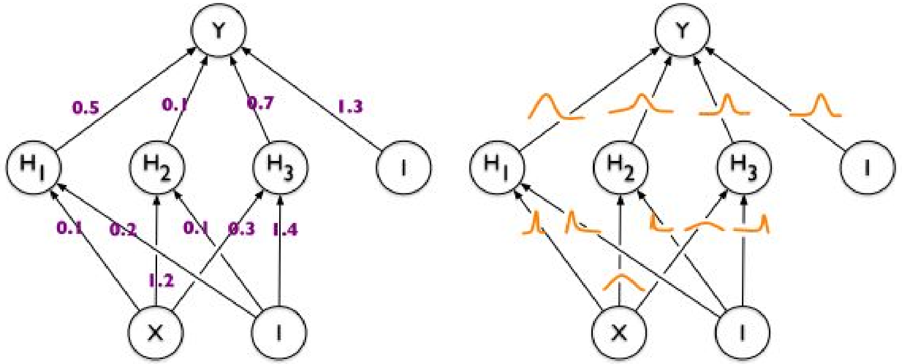
- In classical training we generally find a point estimate:

$$\begin{aligned} \mathbf{w}^{\text{MLE}} &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}). \end{aligned}$$

2. Bayesian Autoencoders

Bayesian Neural Networks

- Posterior distribution over the **weights**: $P(\mathbf{w}|\mathcal{D})$

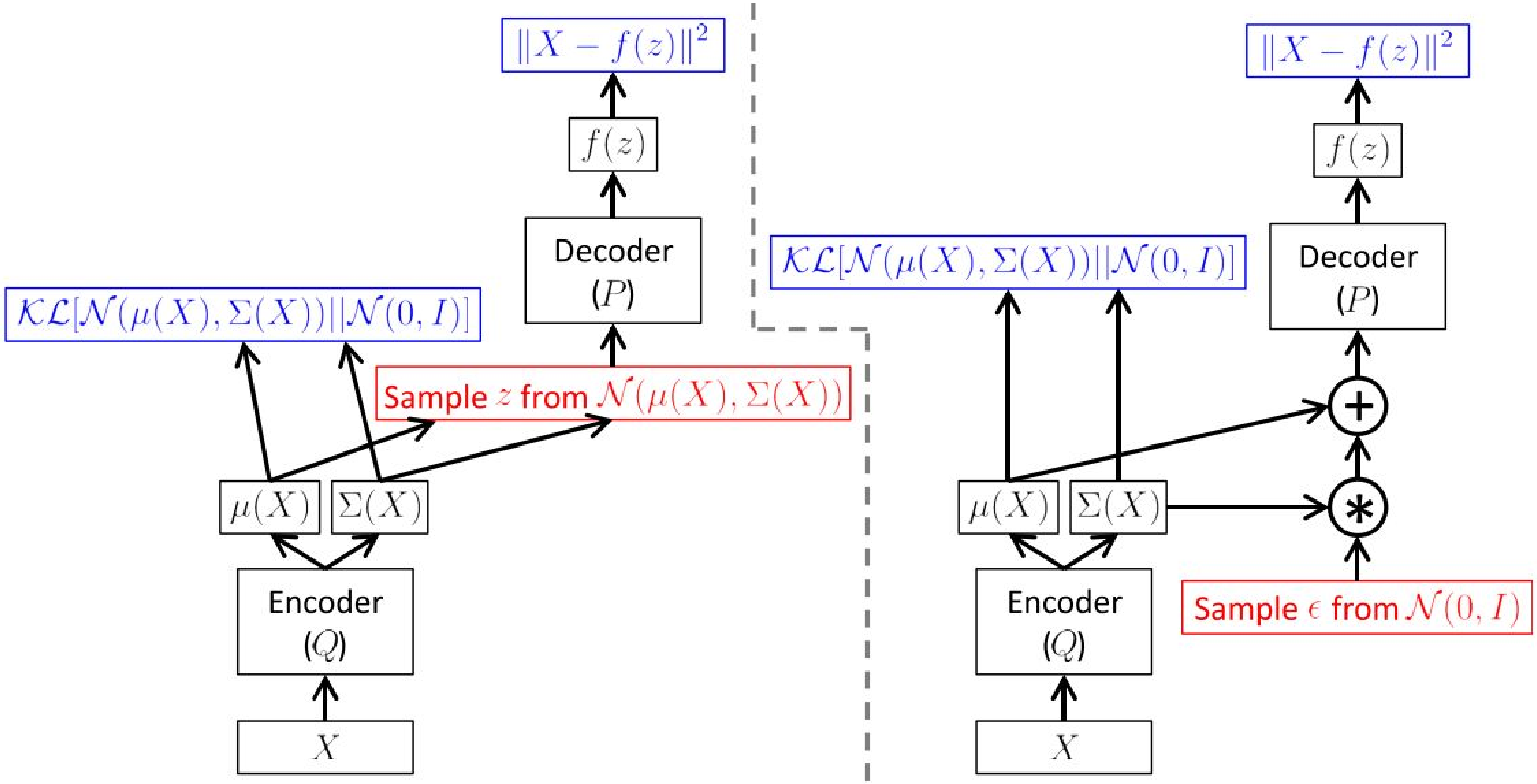


- Minimizing the KL-Divergence between the true posterior and its approximation (minimizing the negative ELBO):

$$
\begin{aligned}
\theta^\star &= \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\
&= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathrm{d}\mathbf{w} \\
&= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \;||\; P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right].
\end{aligned}
$$

Graphic and derivation from C. Blundell et al.: Weight Uncertainty in Neural Networks, Proceedings of the 32nd ICML 2015

Variational Autoencoders



Graphic from Diederik P. Kingma and Max Welling: Auto-Encoding Variational Bayes, ICLR 2014

Variational Autoencoders

- Posterior distribution over the **hidden output z**: $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)}))$
  - We have such a distribution for **each** input sample x(i)!
  - Distribution of **intermediate outputs** instead of weights!

➔ Marginal likelihood + KL-Divergence for each sample!

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

➔ This leads to a **sample lower bound**:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

➔ … which we can sum up to be used in batch optimization:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M}\sum_{i=1}^{M}\widetilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

Formulas from Diederik P. Kingma and Max Welling: Auto-Encoding Variational Bayes, ICLR 2014

3. Evaluation

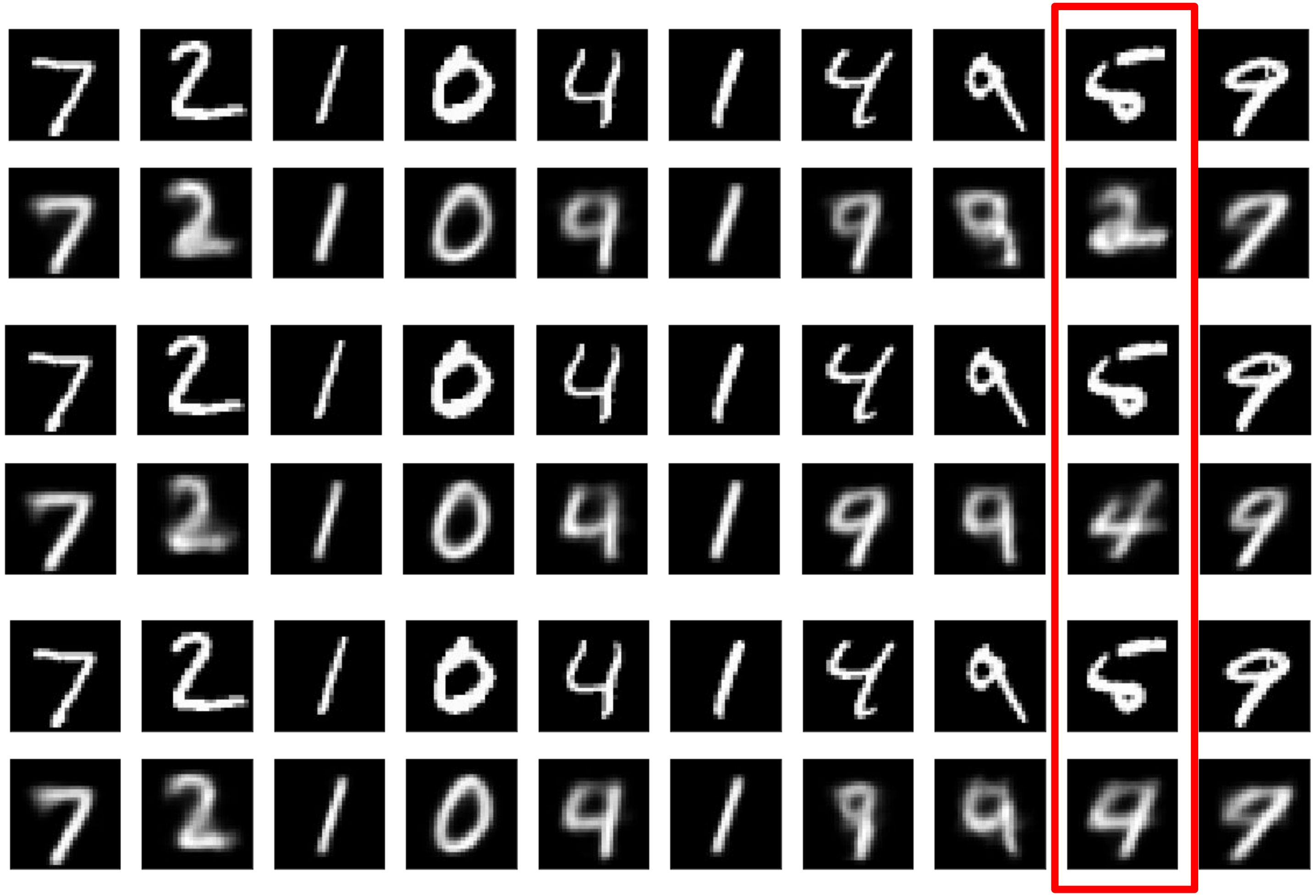
Visual Reconstruction

Neurons in hidden layers: 128x64x2x64x128

Bayesian Neural Network Autoencoder

Variational Autoencoder

Adversarial Autoencoder



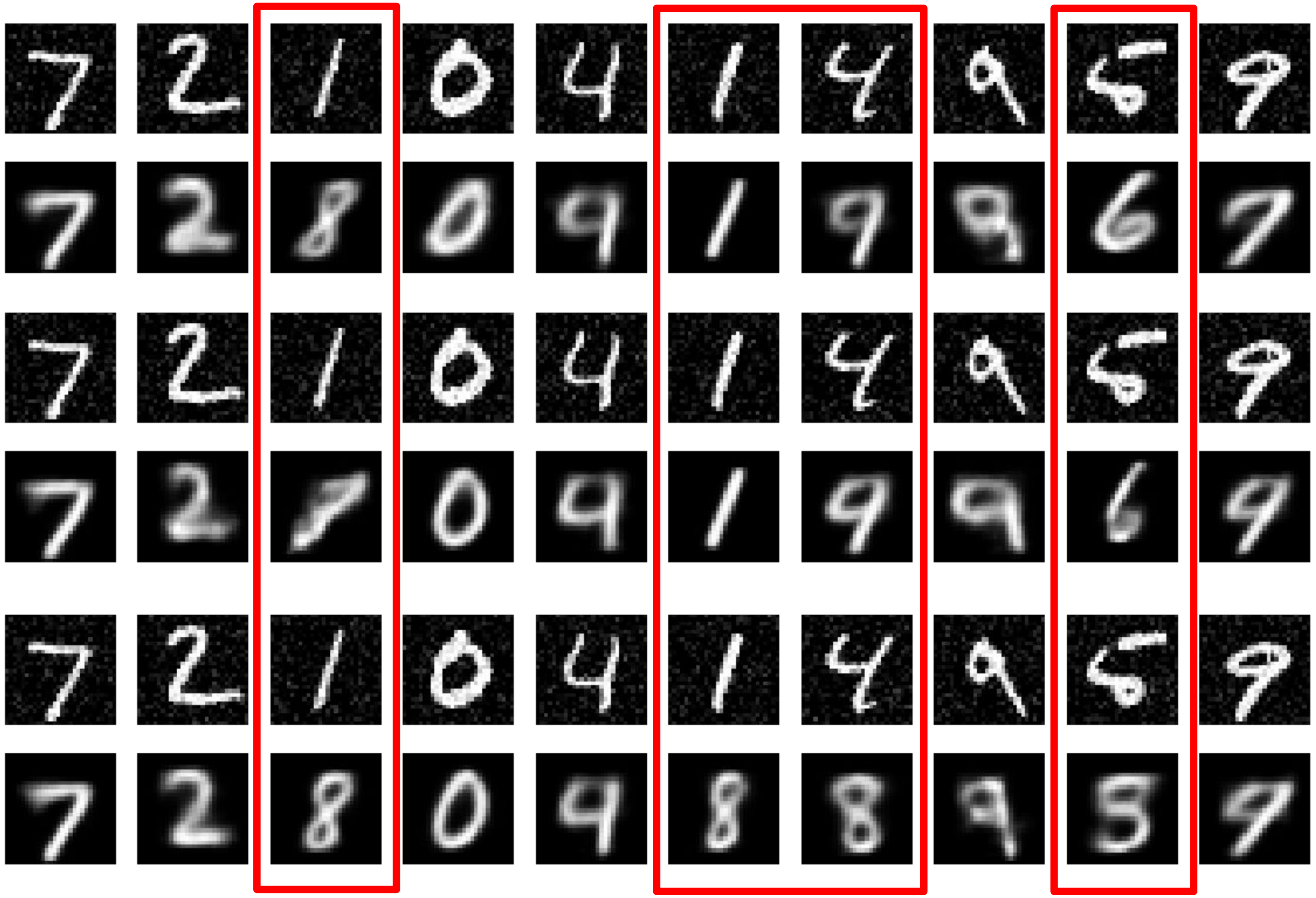
Visual Reconstruction - Noisy

Neurons in hidden layers: 128x64x2x64x128

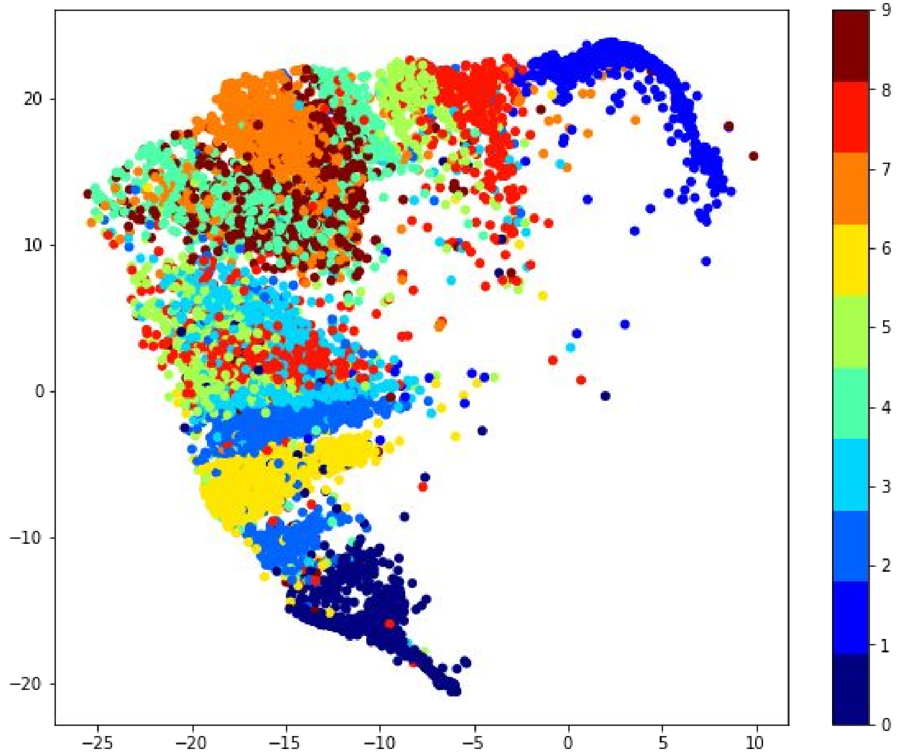
Bayesian Neural Network Autoencoder

Variational Autoencoder

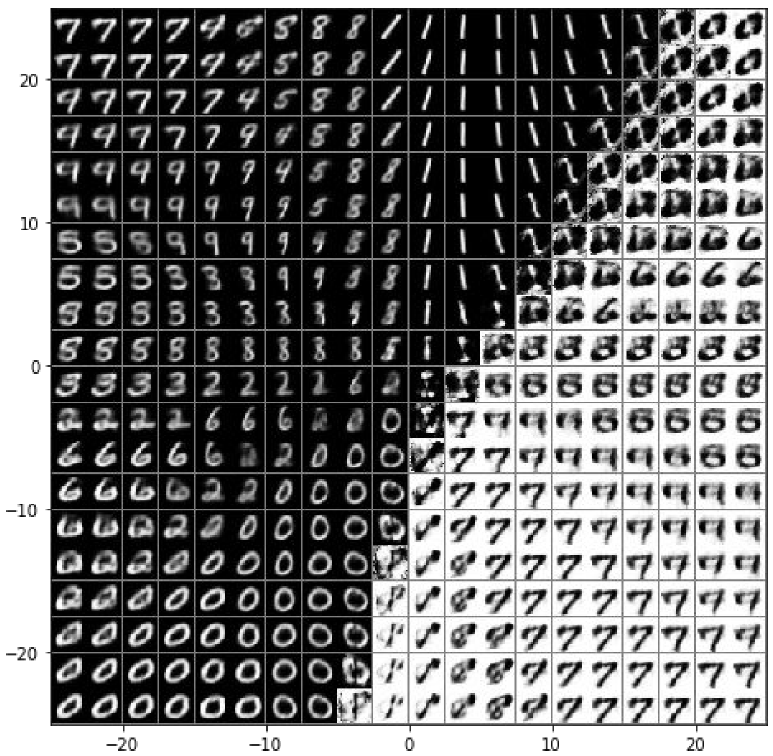
Adversarial Autoencoder



Latent Space - Bayesian NN AE

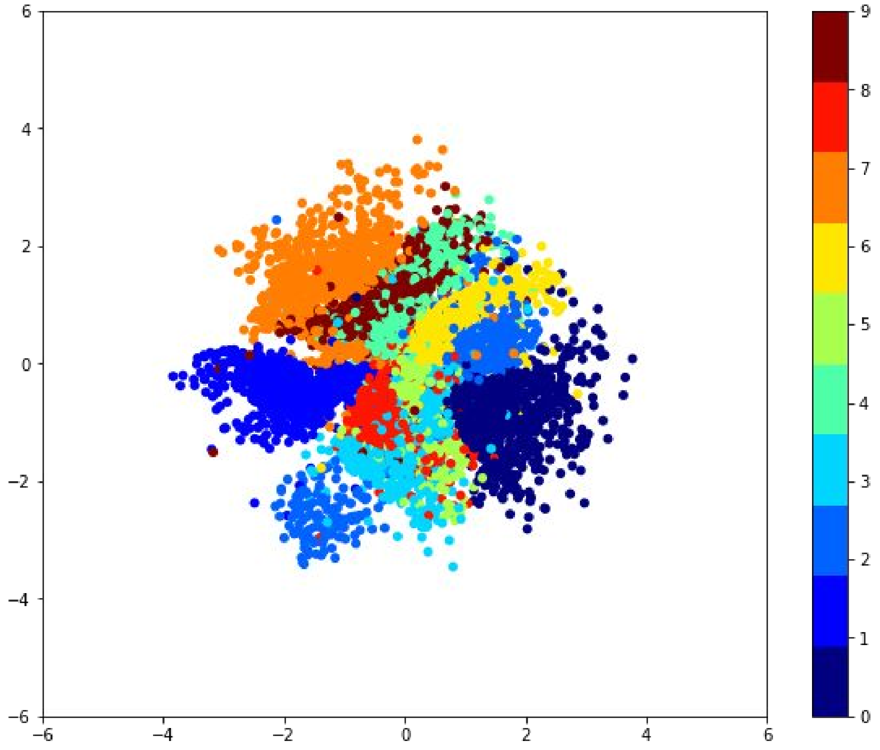


Test data encoded into 2 dimensions

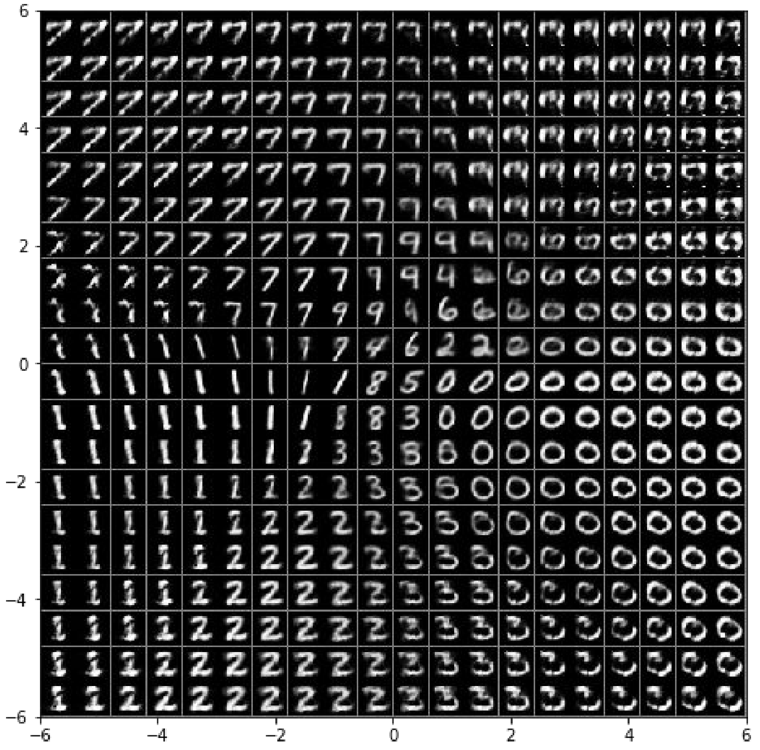


Decoded Reconstruction
(uniform samples)

Latent Space - Variational AE



Test data encoded into 2 dimensions



Decoded Reconstruction
(uniform samples)

1. Autoencoders

2. Bayesian AEs

3. Evaluation

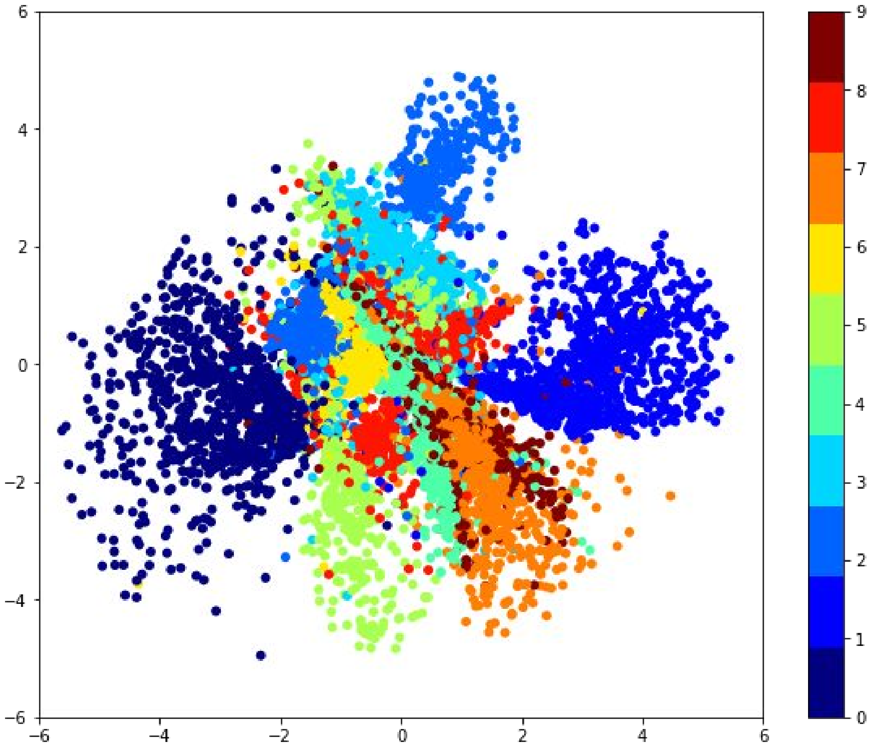
Visual Reconstruction

Latent Space

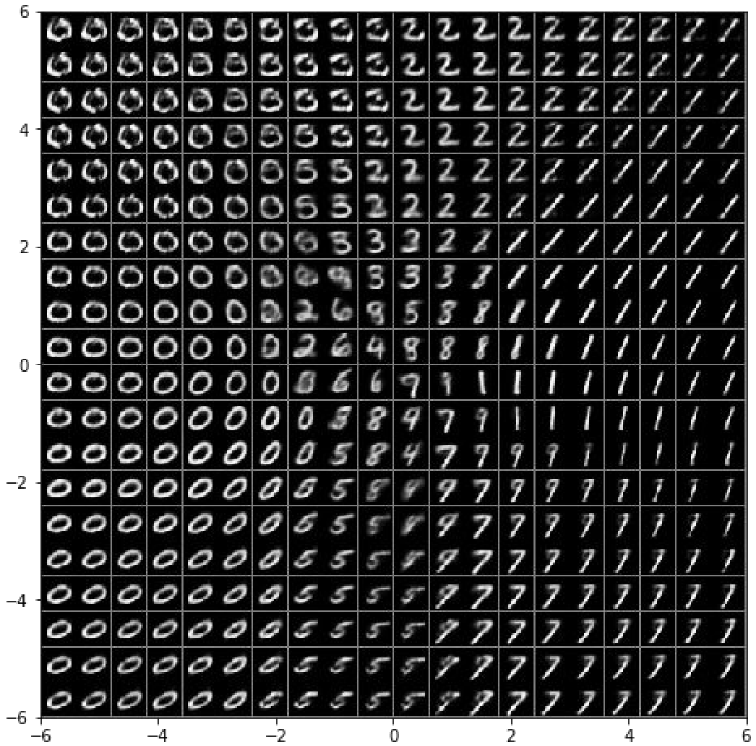
Numerical Results

4. Conclusions

Latent Space - Adversarial AE

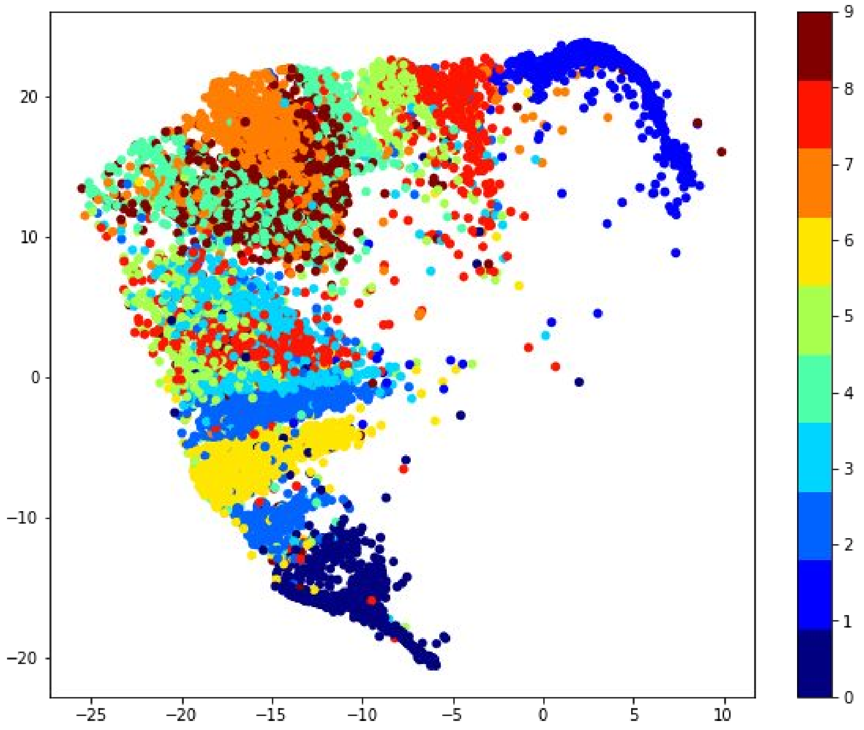


Test data encoded into 2 dimensions

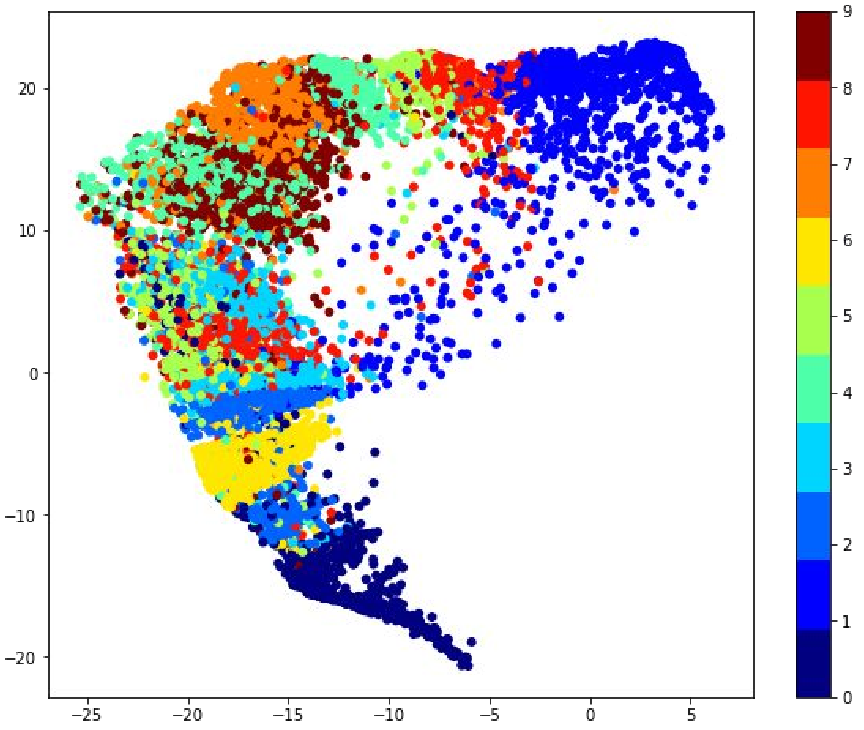


Decoded Reconstruction (uniform samples)

Noisy Latent Space - Bayesian NN AE

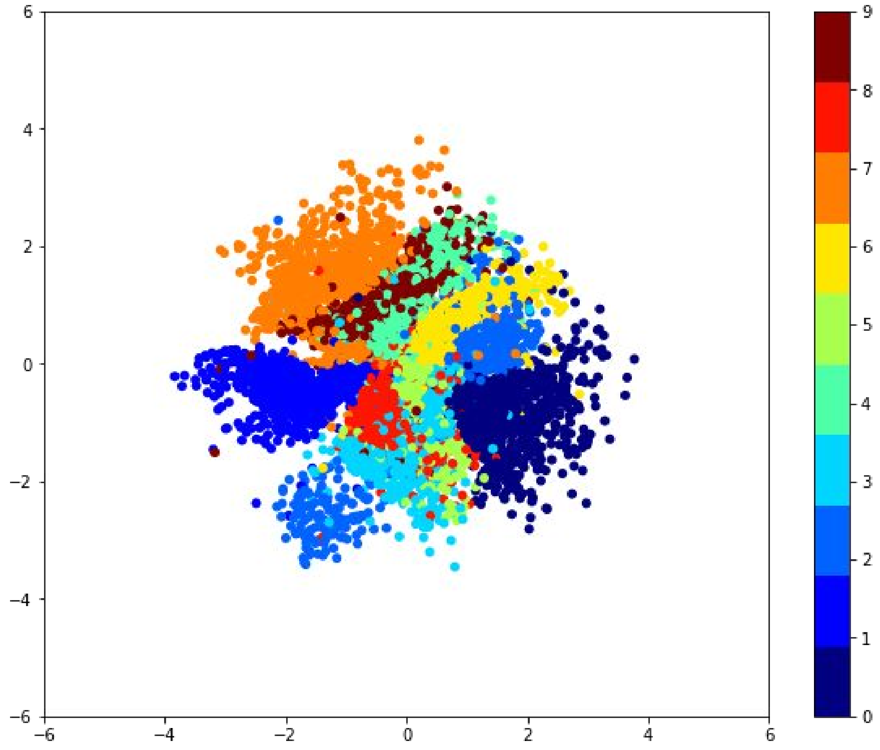


Test data encoded into 2 dimensions

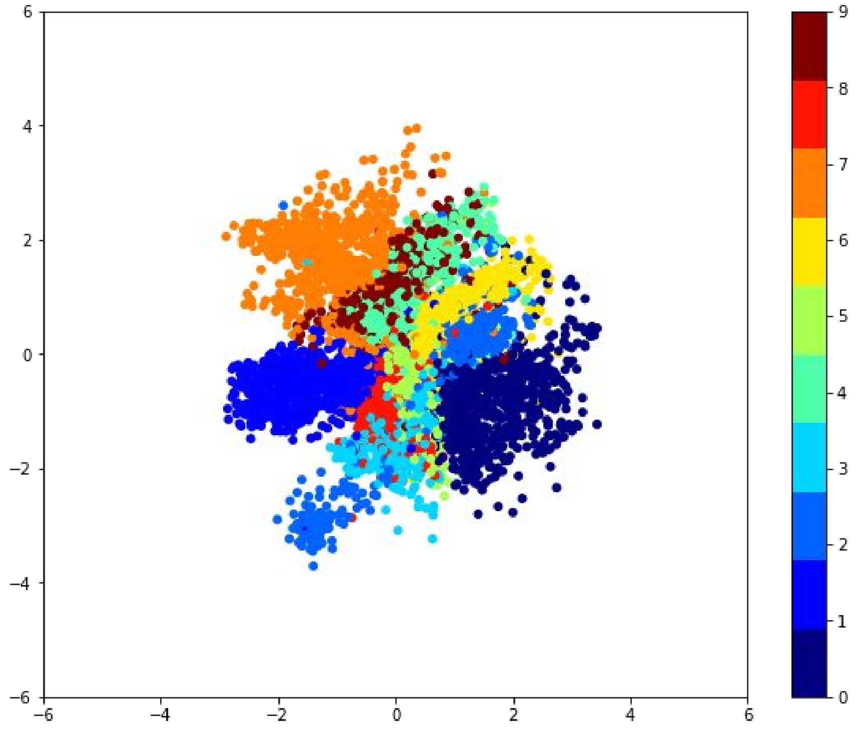


Noisy test data in 2 dimensions

Noisy Latent Space - Variational AE

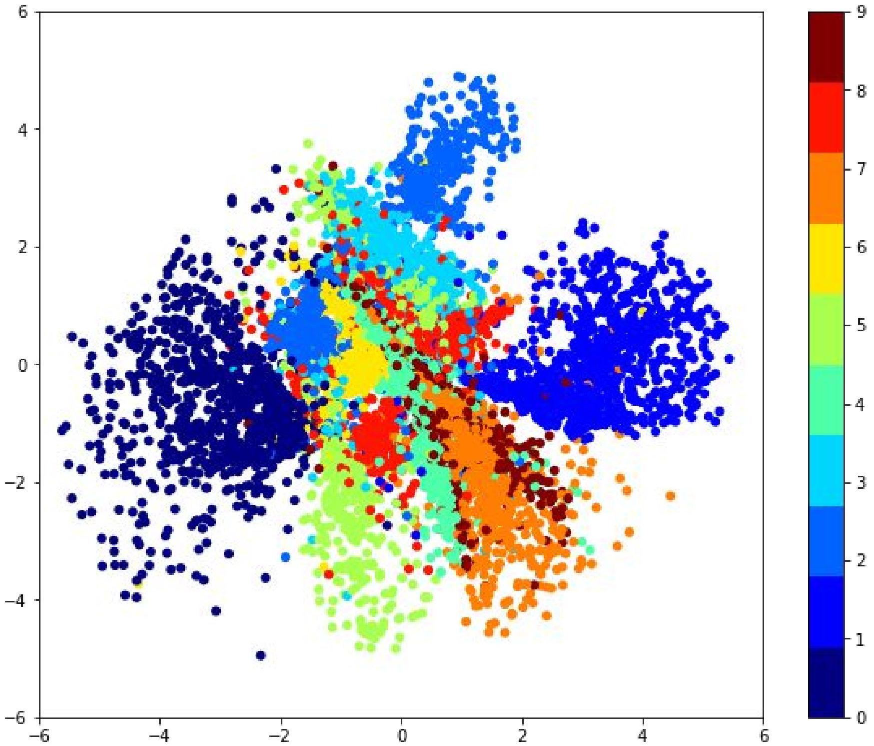


Test data encoded into 2 dimensions

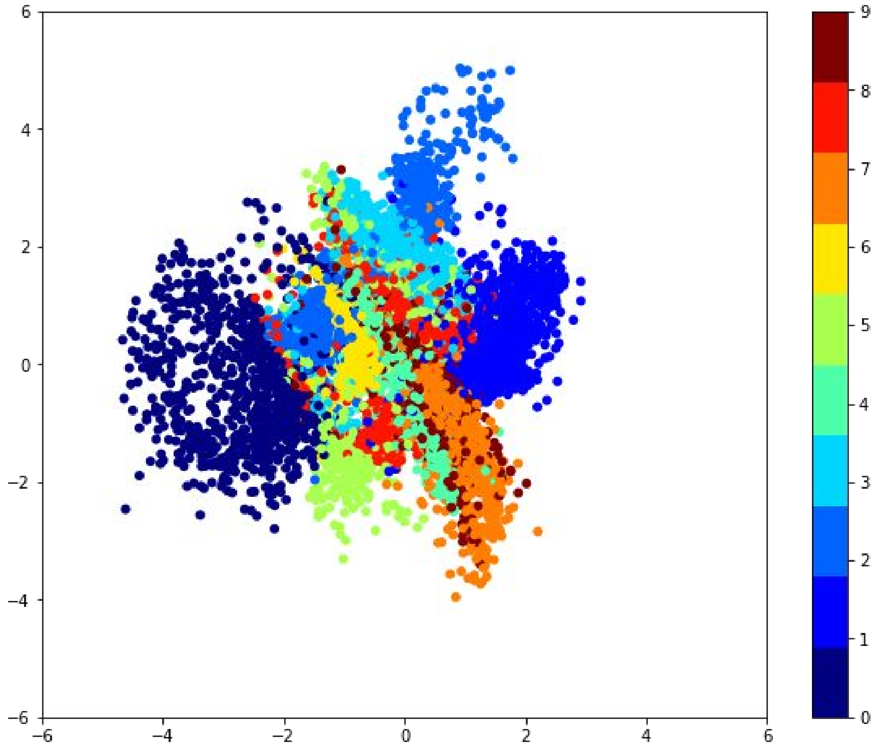


Noisy test data in 2 dimensions

Noisy Latent Space - Adversarial AE



Test data encoded into 2 dimensions



Noisy test data in 2 dimensions

1. Autoencoders

2. Bayesian AEs

3. Evaluation

Visual Reconstruction

Latent Space

Numerical Results

4. Conclusions

Results - Mean Test log-likelihood

Autoencoder / Architecture [hidden layers]	Bayesian NN	Variational AE	Hybrid ("Bayesian VAE")	Adversarial AE
1 hidden layer: **32** neurons	-72.7	-88.3	**-71.4**	-74.8
5 hidden layers: 256x128x**2**x128x256	-131.5	-133.9	-132.4	**-124.9**
5 hidden layers: 256x128x**3**x128x256	-127.3	-122.8	**-120.6**	-136.9
5 hidden layers: 256x128x**32**x128x256	**-59.4**	-79.1	-59.8	-68.2
5 hidden layers: 128x64x**2**x64x128	-140.5	-136.2	**-134.9**	-135.5
5 hidden layers: 128x64x**3**x64x128	-133.7	-123.7	-133.9	**-117.1**
5 hidden layers: 128x64x**32**x64x128	**-62.6**	-87.4	-63.1	-78.0

Results - Denoising Test LL

Autoencoder Architecture [hidden layers]	Bayesian NN	Variational AE	Hybrid ("Bayesian VAE")	Adversarial AE
1 hidden layer: **32** neurons	**-75.5 (-2.8)**	-142.3 (-54)	-105.9 (-34.5)	-106.0 (-31.2)
5 hidden layers: 256x128x**2**x128x256	**-143.6 (-12.1)**	-149.7 (-15.8)	-147.1 (-14.7)	-176.1 (-51.2)
5 hidden layers: 256x128x**3**x128x256	-143.8 (-16.5)	-143.4 (-20.6)	**-133.8 (-13.2)**	-181.6 (-44.7)
5 hidden layers: 256x128x**32**x128x256	**-96.1 (-36.7)**	-116.8 (-37.7)	-102.1 (-42.3)	-140.0 (-71.8)
5 hidden layers: 128x64x**2**x64x128	-153.9 (-13.4)	**-151.0 (-14.8)**	-153.9 (-19.0)	-188.6 (-53.1)
5 hidden layers: 128x64x**3**x64x128	-147.3 (-13.6)	-152.7 (-29)	**-146.7 (-12.8)**	-170.6 (-53.5)
5 hidden layers: 128x64x**32**x64x128	-101.8 (-39.2)	-129.3 (-41.9)	**-97.5 (-34.4)**	-133.0 (-55.0)

4. Conclusions

1. Autoencoders

2. Bayesian AEs

3. Evaluation

4. Conclusions

Conclusions: What we have done...

- Models we tested:
  - Bayesian Neural Network
  - Variational Autoencoder
  - Hybrid Variational Autoencoder
  - Generative Adversarial Network
  - Convolutional versions of the above

- Comparisons with respect to test data:
  - Visual reconstructions
  - Log-Likelihood
  - Latent space representation and reconstruction
  - Effect of gaussian noise on all of the above

Conclusions: Results

- Visual Reconstructions:
  - Seem overall similar
  - Even slight noise confuses models!
- Test Log-Likelihood:
  - All of the models achieve similar results
  - However:
    Models with a constrained latent space seem more sensitive to noisy input!
- Latent space
  - Varies strongly across models
  - General shape does not impact reconstruction performance
  - May be distorted by noise
- Model complexity
  - Training Time: VAEs < GANs ($\approx$ x1.5) < Bayesian NNs ($\approx$ x5.5)
  - Parameters: VAEs < GANs ($\approx$ x1.5) < Bayesian NNs ($\approx$ x4)

Conclusions: Future Work

- We used a Bernoulli likelihood. What about a gaussian one?
- So far we have only compared performance on MNIST
  - What about CIFAR or small datasets?
  - Impact of regularization (DKL and prior)?
- Bayesian variational autoencoder?
  - How do we derive the lower bound of a doubly-stochastic model?
- Better approximation of the true posterior
  - Non-gaussian distributions
  - Different inference techniques
- Test the **generative** capacities of the models

Thank you!
Questions?