EXPLICITLY DISENTANGLING IMAGE CONTENT FROM TRANSLATION AND ROTATION WITH SPATIAL-VAE

Yanislav Donchev Donchev, Joe Simon & Pier Paolo Ippolito
{ydd1g16,js23g16,ppi1u16}@ecs.soton.ac.uk

1 INTRODUCTION

Common issues when training models using real images are the inconsistencies in the pose of the captured image. For example, a famous landmark could be captured by tourists from different angles, elevations and shifts in position, resulting in different looking images of the same object relative to a fixed position. When learning a low-dimensional latent representation of these images, these inconsistencies in the pose might be mixed up with the actual content of the image in the latent space.

Spatial-VAE (SVAE) is a variational autoencoder framework that seeks to learn the rotation and translation pose variables (or any other linear transformation) separately from the continuous latent space variables in an unsupervised manner. Unlike a vanilla[1] VAE (VVAE), where the decoder outputs the whole image at once, the decoder of an SVAE outputs a single-pixel given the pixel's spatial coordinates. Apart from the normal latent variables of a VVAE, the encoder of an SVAE learns a set of transformation parameters with which the input pixel coordinates are transformed, thus "explicitly disentangling the content from linear transformations" of the input image.

In this report, we reproduce the implementation of the NeurIPS 2019 paper - SpatialVAE, which attempts to learn the rotation and translation pose transformations and critique its reproducibility by comparing results and judging its difficulty. Although the authors provide code[2] for their paper, we reimplemented their work[3] based on the paper alone using PyTorch and PyTorch Lightning.

2 IMPLEMENTATION DETAILS

The SVAE architecture (see Figure 1) is very similar to a VVAE with a few notable changes. First, the encoder models a few extra distributions that encode the transformation parameters, in addition to the traditional unconstrained latent variables. The sampling (or reparameterisation) is similar to a VVAE and uses the typical prior $\mathcal{N}(0, I)$ for the unconstrained latent variables $\mathbf{z}$. Since it is likely to have prior knowledge of the transformations, a separate prior is defined for these. Due to the rotation of the image θ being bounded, the authors define an adjustment to the standard KL divergence (equation (3) in Bepler et al. (2019)). The biggest difference from a VVAE is in the decoder: instead of accepting only $\mathbf{z}$, the decoder also takes a two-dimensional spatial pixel coordinate $\mathbf{x}$, which has been linearly transformed based on the sampled latent space variables for rotation (θ) and translation ($\Delta\mathbf{x}$). The input to the decoder is the concatenated vector $(\mathbf{z}, \mathbf{x})$, where $\mathbf{x} = (T(\theta, \Delta\mathbf{x}) \cdot \mathbf{x}')_{1:2}$, and the transformation matrix and non-transformed spatial coordinates are defined in (1) and (2) respectively. This operation is done for a normalised meshgrid (28×28 in the case of MNIST) to reconstruct the full image.

$$T(\theta, \Delta\mathbf{x}) = \begin{bmatrix} cos(\theta) & -sin(\theta) & \Delta x_0 \\ sin(\theta) & cos(\theta) & \Delta x_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{x}' = \begin{bmatrix} x_0 & x_1 & 1 \end{bmatrix}^T \tag{2}$$

where $x_0, x_1 \in [-1, 1]$.

[1] The standard VAE architecture is referred to as *vanilla*.

[2] Original implementation: https://github.com/tbepler/spatial-VAE.

[3] Our implementation: https://github.com/COMP6248-Reproducability-Challenge/SVAE.

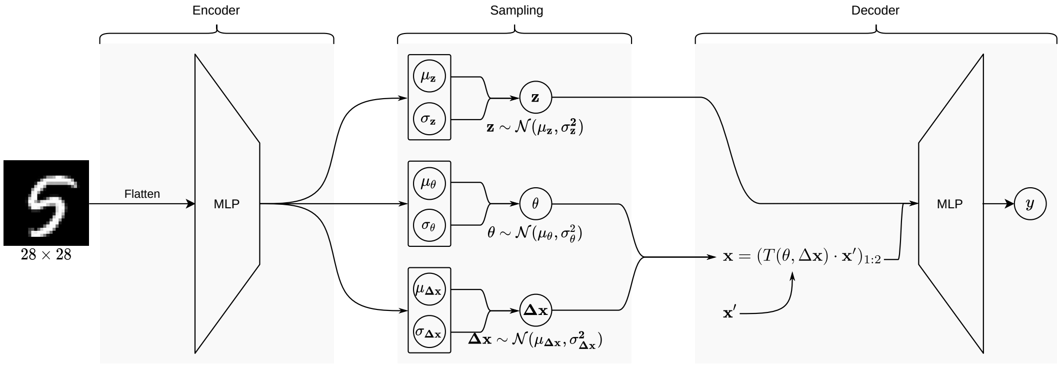
Encoder
Sampling
Decoder
Flatten
MLP
28×28
$\mu_{\mathbf{z}}$
$\sigma_{\mathbf{z}}$
$\mathbf{z}$
$\mathbf{z} \sim \mathcal{N}(\mu_{\mathbf{z}}, \sigma_{\mathbf{z}}^2)$
μ_θ
σ_θ
θ
$\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$
$\mu_{\boldsymbol{\Delta}\mathbf{x}}$
$\sigma_{\boldsymbol{\Delta}\mathbf{x}}$
$\boldsymbol{\Delta}\mathbf{x}$
$\boldsymbol{\Delta}\mathbf{x} \sim \mathcal{N}(\mu_{\boldsymbol{\Delta}\mathbf{x}}, \sigma_{\boldsymbol{\Delta}\mathbf{x}}^2)$
$\mathbf{x} = (T(\theta, \Delta\mathbf{x}) \cdot \mathbf{x}')_{1:2}$
$\mathbf{x}'$
MLP
y

Figure 1: Diagram representation of the spatial-VAE framework. The data flow is clearly defined to guide implementation. A notable difference from a VVAE is that the network outputs a single-pixel value y instead of the full image, and the decoder takes in an extra input - the transformed spatial pixel coordinate **x**. The output is squished by a sigmoid.

3 Experimental Methodology

To test their hypothesis – can the network explicitly differentiate linear transformations from content – the authors use the MNIST dataset and two variations of it. While the standard MNIST dataset provides a good baseline, the two additional variants – rotated MNIST and rotated & translated MNIST – help in observing the network's performance on pose transformed images. The rotated MNIST has randomly sampled angles from $\mathcal{N}(0, \pi^2/16)$ applied to the images along with a small random translation from $\mathcal{N}(0, 1.4^2)$. The more challenging rotated & translated MNIST dataset maintained the same degree of rotation but added a greater degree of translation, randomly sampled from $\mathcal{N}(0, 14^2)$.

The original paper also tests the framework on the *Sloan Digital Sky Survey Galaxy Zoo*, and *Single Particle Electron-Microscopy* to address the challenges mentioned in Section 1. This report focuses solely on experimentation on the MNIST dataset and its variants due to the size and GPU hardware requirements of the latter datasets.

Three VAE models were tested for comparison: VVAE acting as a baseline, SVAE and SVAE with θ and $\Delta\mathbf{x}$ set to zero. Four different latent dimensions Z-D were tested for each of the models above: Z-D $\in \{2, 3, 5, 10\}$. The encoders and decoders are standard two-layer MLPs with 500 neural units each and tanh activations.

The SVAE employs the same parameters but it has three additional latent variables – two that encode translation and one for rotation. It outputs a single probability to represent binary pixels. Its rotation prior for the regular MNIST is set to $\mathcal{N}(0, \pi^2/64)$, while the rotation prior is set to $\mathcal{N}(0, \pi^2/16)$ for the transformed MNIST datasets. The translation prior is set to $\mathcal{N}(0, 1.4^2)$ for the transformed datasets and set to a constant of zero for the regular MNIST. The second SVAE model with $\theta = 0$ and $\Delta\mathbf{x} = 0$ is identical to the above model with the exception that the input coordinates $\mathbf{x}'$ are not transformed (i.e. multiplied by I).

All models are trained with the ADAM optimiser, learning rate of 1-e4 and a minibatch size of 100. The loss function aims to maximise the Evidence Lower Bound (ELBO). Every combination was run for 200 epochs.

4 Results & Evaluation

A total of 36 networks were trained with total runtime of 54 hours. The VVAE models took approximately 0.5 hours each to train, while the SVAE models took significantly longer at approximately 2

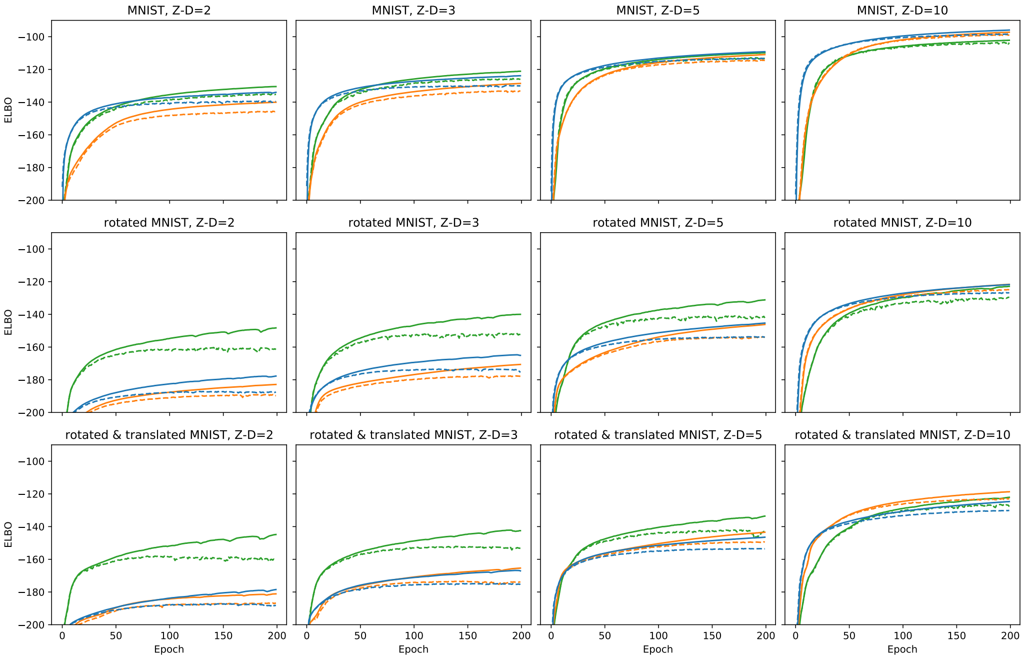

Figure 2: Comparison of ELBO logs of the VAE models on the MNIST flavours while varying the dimensions of the unstructured latent variables. The lines show training (solid) and testing (dashed) ELBO for the VVAE model (blue), SVAE model (green) and SVAE with $\theta = 0$ and $\Delta\mathbf{x} = 0$ (orange).

hours. We trained the networks in parallel on 9 separate GPUs (Nvidia GTX 2070 and Nvidia P100) from Kaggle and Google Colab.

The ELBO logs of these runs are shown in Figure 2. The plots quite closely resemble the images shown in Figure 2 in Bepler et al. (2019). The performance of the SVAE is well highlighted when the latent space dimension is low, and even more so in the more challenging rotated MNIST and rotated translated MNIST datasets. It becomes apparent that the positive effect of SVAE on ELBO diminishes as Z-D grows, and VVAE achieves similar results. Overall, the differences with the original results are marginal and could be the effect of a different random initialisation.

Similarly, the latent space manifolds of the models are reproduced in Figure 3 and seem to be in line with Figure 3 in Bepler et al. (2019), confirming that the described framework is reproducible. The benefit of SVAE becomes apparent as it can capture the content of the heavily transformed MNIST datasets, while VVAE fails to do so.

To make the benefit of SVAE more intuitive, we converted our model that was trained on the rotated & translated MNIST dataset to the Open Neural Network Exchange (ONNX) format and deployed it in an interactive webpage[4]. Handwritten digits can be drawn and linearly transformed using sliders, while the model generates stable samples that correct for these transformations.

While the experiments and framework were well described allowing us to replicate it from scratch, the lack of specific model initializations and repeated runs on different random seed values prevented robust testing and prevented the measurement of uncertainty in the results.

Additionally, their codebase seemed unwieldy and hard to follow due to the lack of good coding practices and sparse comments, making it difficult for people to understand the codebase along with the paper. Our codebase aimed to rewrite the entire codebase solely based on the original paper and Kingma & Welling (2014), providing a closely matched structure and good practices, allowing the

[4] Interactive demo: https://comp6248-reproducability-challenge.github.io/SVAE/

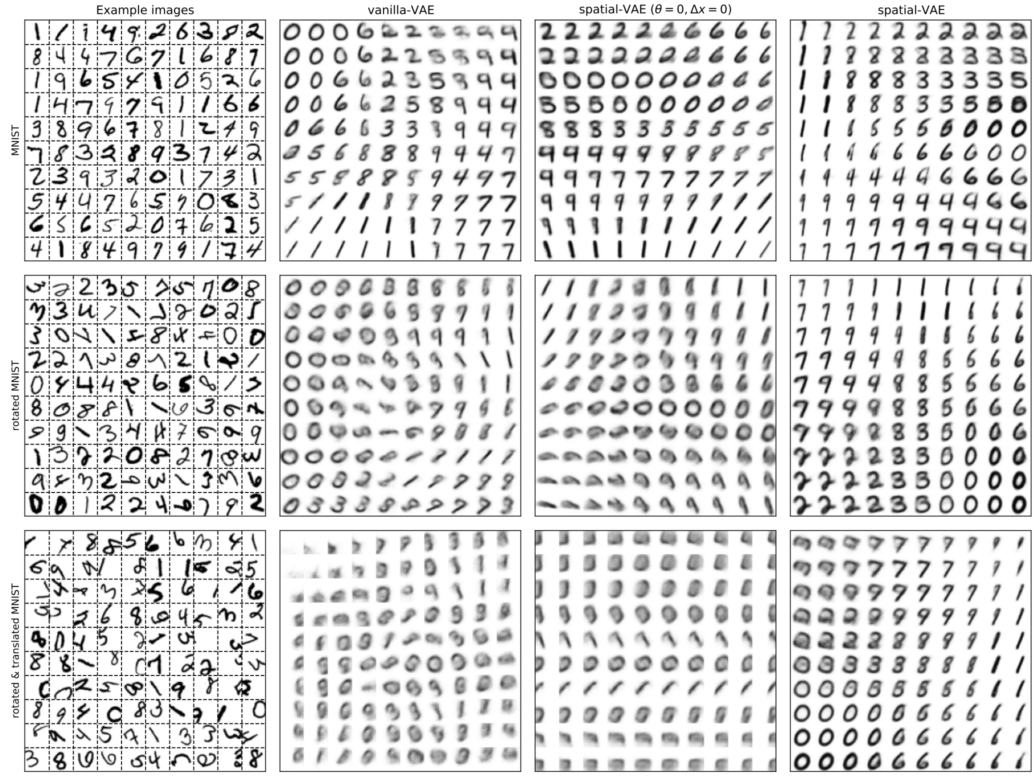
Example images
vanilla-VAE
spatial-VAE ($\theta = 0, \Delta x = 0$)
spatial-VAE
MNIST
rotated MNIST
rotated & translated MNIST

Figure 3: The latent space manifolds of the three models with Z-D=2, trained on the MNIST variants. The plots show the outputs of the decoders for uniform samples of the prior.

readers to better understand the work. Improved commenting on the structure and the maths, and Python docstrings were added to further aid users.

5 Conclusion

Overall, the paper was written well allowing its results to be reproducible and the claims of the authors stood true – the SVAE architecture successfully learns linear transformations of the input image, separate from the semantics. However this comes at a cost: the decoder has to be executed once for each pixel of the image, which significantly increases the runtime. Furthermore, if the claimed disentangling is not required in the application, a VVAE with a bigger latent space achieves a similar ELBO to that of the SVAE at a much lower computational cost. Additionally, an effort was put on to make the work more accessible and readable through the help of a rewritten codebase and a live interactive demo of the work.

References

Tristan Bepler, Ellen Zhong, Kotaro Kelley, Edward Brignole, and Bonnie Berger. Explicitly disentangling image content from translation and rotation with spatial-VAE. In *Advances in Neural Information Processing Systems*, pp. 15409–15419, 2019.

Diederik P Kingma and Max Welling. Stochastic gradient VB and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, 2014.