Disentangling Symbols and Movements: Factor-VAE on NAR Dataset

Subrat Kumar Swain(2021QIZ8247)
Sweta Mahajan(2021AIZ8356)

July 27, 2022

Abstract
We have implemented Factor-VAE on NAR data. The data is taken from here [3]. FactorVAE [1] is a method which tries to disentangle the semantic attributes of the input image by encouraging the distribution of the latents to be factorial which in turn encourages the latent dimensions to be independent of each other. If the conditional distribution of the latents given the data is powerful, then the latents encode the input really well which will ensure the semantic factors are well disentangled.

`Code: https://github.com/swainsubrat/FactorVAE`

1 Key Idea

Learning latent representation of images which are semantically meaningful is very much crucial for tasks such as supervised learning, reinforcement learning, transfer learning and zero-shot learning. Learning these features in a disentangled way is also important for synthetic image formation, image editing and film making. Moreover, class disentanglement helps in adversarial detection and defense [4].

In this work, the authors propose Factor VAE to achieve disentanglement. They show that it improves upon $\beta-$VAE by providing better trade-off between disentanglement and reconstruction quality. $\beta-$VAE is also a generative model built on top of the Variational Autoencoder (VAE) Framework.

In traditional VAE, the loss function is,

$$E_{p_{data}(x)}\Big[[-E_{q(z|x)} log\, p(x|z)] + \beta KL(q(z|x)||p(z))\Big]$$

The first term is the KL divergence between the prior and the approximated conditional $q_\phi(z|x^{(i)})$ and the second term is the reconstruction error. In this traditional

VAE, the latent factors achieve limited disentangling on simple datasets like MNIST and FreyFaces and does not scale to complex datasets.

Since x is high dimensional and integrating over it is computationally not feasible, we resort to Monte Carlo estimates.

$$\frac{1}{N}\sum_{i=1}^{N}[E_{q(z|x^{(i)})}[log\, p(x^{(i)}|z)] - \beta KL(q(z|x^{(i)})||p(z))]$$

A slight modification to the loss function yields Beta-VAE which tries to get a balance between how well the input can be reconstructed and how well the latent representations are factorised. We balance this trade-off using a hyper-parameter β. However, the objective function increases the reconstruction error at the expense of better disentanglement. This is the primary shortcomings of Beta-VAE. In Factor VAE, increase in the disentanglement doesn't decrease the quality of reconstruction. Rather it augments the VAE loss function with another term called Total Correlation, $KL(q(z)||\overline{q}(z))$, which is solely responsible for disentangling even better. So, with the same reconstruction error loss as in VAE, we get better disentanglement. Factor VAE implements a discriminator neural network to evaluate the Total Correlation term using the density ratio trick; the authors train the VAE and the discriminator jointly.

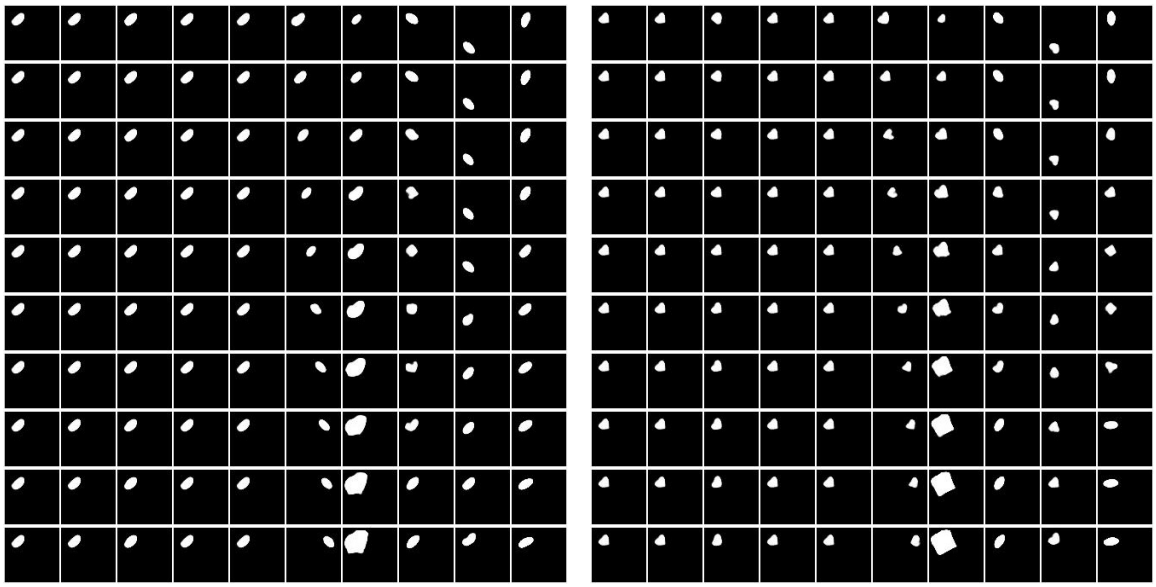

Figure 1: Reconstructions of latent traversals across each latent dimension of dsprites dataset (left)Ellipses, (right)Hearts.

2 Experiment and Results

We start by running the Factor VAE algorithm on the dsprites dataset [2]; 737,280 binary 64 $\times$ 64 images of 2D shapes with ground truth factors[number of values]: shape[3], scale[6], orientation[40], x-position[32], y-position[32]. This is the dataset

used by the authors of the paper to show disentanglement. We are able to successfully reproduce the results Figure [1]. It can be clearly observed that, traversing the latent space and reconstructing the same yields disentangled feature traversal patterns. To apply the same idea on NAR data, we had to make some modification to the architecture and the hyper-parameters. However, the key idea and propagation of the input remain the same. The NAR dataset consists of 60,000 images, shape of each being 64×64. We modified the Convnet used in the encoder as well as the decoder to better capture the features. We kept the size of the hidden dimension same as 10 as increasing it to 20 was not making any significant difference. We hand tuned the other hyper-parameters such as VAE and Discriminator's learning rate, optimizer parameters such as beta, etc. To train the model, we used a V100 GPU with 32GB of RAM. We trained for 4000 epochs which took around 20 hours. The original experiment was done for 700000 epochs. Due to resource constrain, we only performed it for 4000 epochs. At the end of the training, reconstruction loss was oscillating around 70, KL divergence loss was around 13, VAE total correlation loss was between 1 to 1.5 and discriminator loss around 0.5. All the loss value, with iteration is shown in Fig [2].

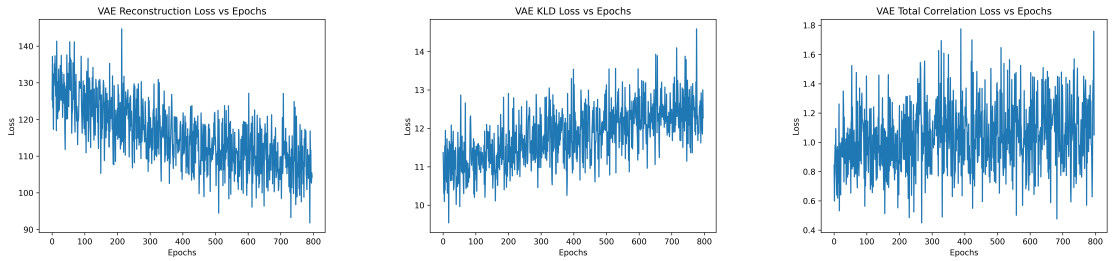

Figure 2: Loss vs Epochs plot for (a) Reconstruction (b) KL Divergence (c) Total Correlation

As shown in Fig [4], we ran the VAE model on the MNIST as well as on the NAR dataset. In MNIST, the background of the input and the reconstructed image is of the same colour. However, for the NAR dataset, the background colour of the reconstructed image is light green whereas the background of the input image is yellow in colour. This can be attributed to the fact that, in the MNIST dataset, the images are placed approximately in the same position on the 28*28 grid. Whereas for NAR dataset, the images are small and are spatially varying from task to task. Also the symbols in the NAR dataset are drawn dark blue in colour surrounded by a strip of light green and this light green colour is captured by the network. Another difference is that, in the MNIST dataset, the digits occupy majority part of the grid whereas in the NAR dataset, the symbols occupy a very small place on the grid. Owing to these attributes, NAR dataset is seemingly difficult to train on than conventional datasets like MNIST.

After completely training the network, we ran inference on the network. In inference, we passed the image through the encoder and got the latent vector. Then we traversed the latent space by perturbing one dimension of the latent vector by 0.1 and increase it by 0.1 for 10 times. We plotted the obtained images in Figure [3]

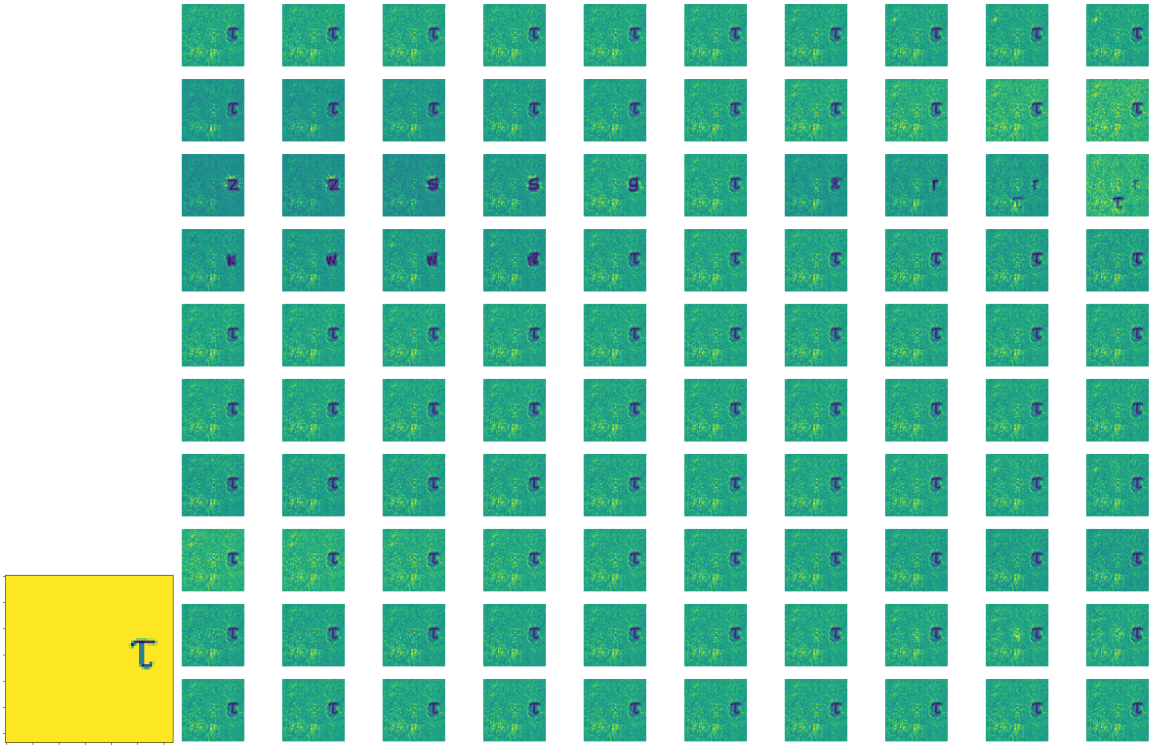

Figure 3: (a) Original image, (b) Reconstructions of latent traversals across each latent dimension of NAR data(Traversed from -0.5 to 0.5 with an interval of 0.1). The sixth one in each row is the original reconstruction, without any perturbations

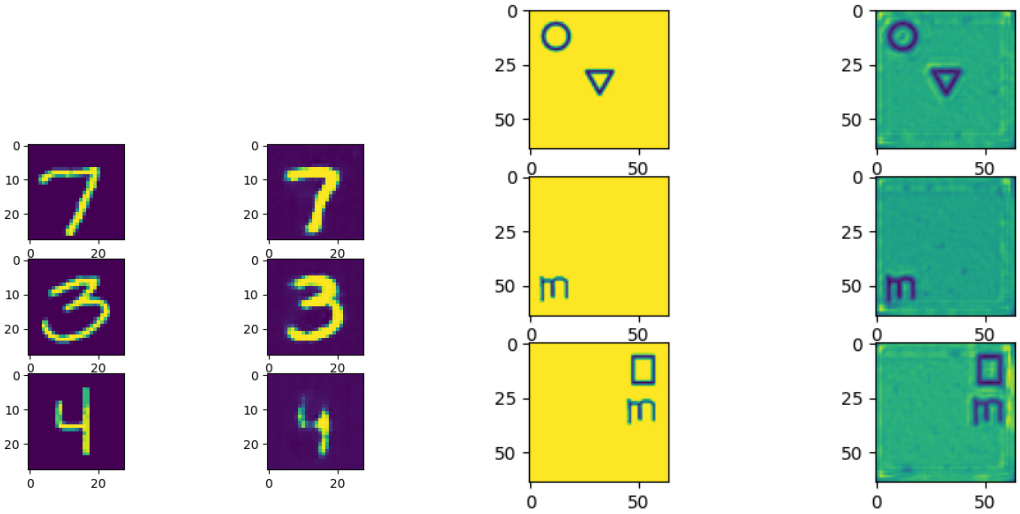

Figure 4: MNIST and NAR reconstruction

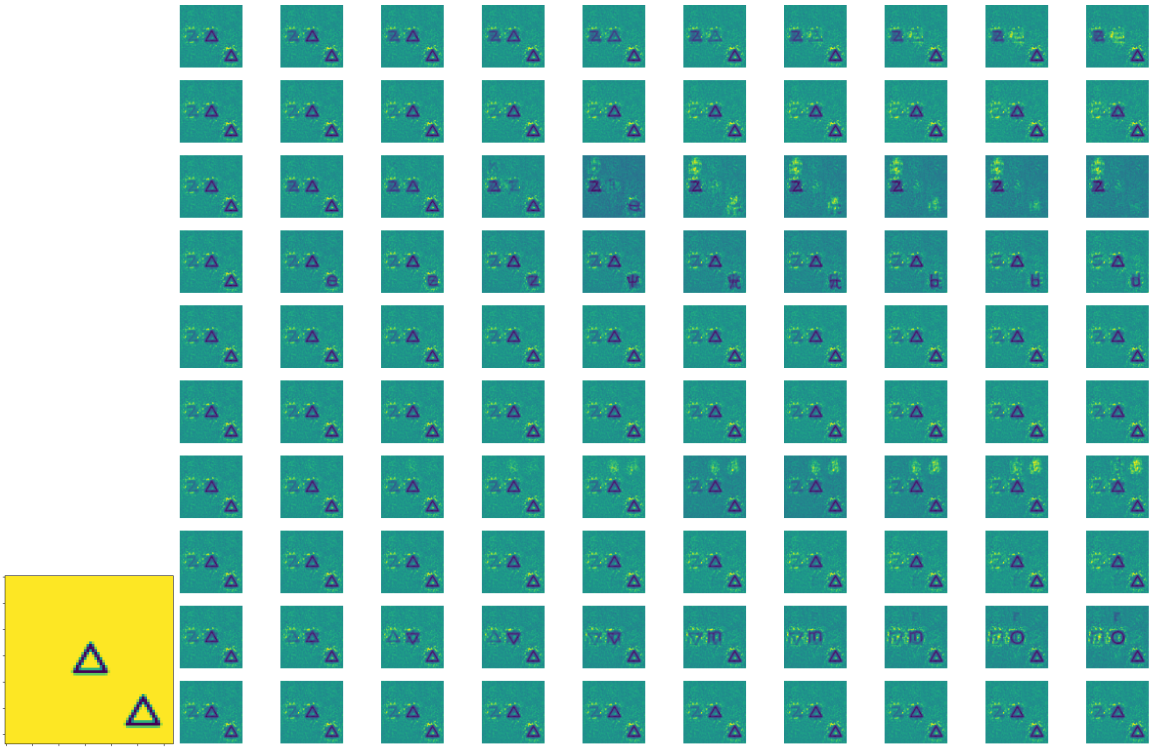

Figure 5: (a) Original image, (b) Reconstructions of latent traversals across each latent dimension of NAR data(Traversed from 0 to 1.0 with an interval of 0.1). The first one in each row is the original reconstruction, without any perturbations

and [5]. In the Figure [3], (a) is the original image. In (b), each row represents a dimension in the latent space and each column represents some perturbations added to that dimension. We started adding perturbation from -0.5 and increase it till 0.5 with a step size of 0.1. It can be clearly seen that, in the first row, there is no change even if perturbing value along that dimension. This signifies that, the disentangled feature captured by the first row doesn't apply to this image. Similarly, it's also invariant in 5^{th} to 10^{th} dimension. Similarly, in Figure [5], (a) is the original image; (b) each row represents a dimension in the latent space and each column represents perturbations of 0.1 started from 0 till 1. Here, traversing through the first dimension makes the triangle in the center invisible and constructs a z in the left most column. Similarly, traversing through each dimension changes something in a particular way.

3 Future Direction

3.1 A New Way to Measure Quality of Disentanglement

In the original paper, authors have mentioned a new way to measure the quality and degree of disentanglement. We also came up with an innovative way to measure disentanglement. This may or may not a viable option to measure the degree of disentanglement. First, we have to obtain a segmentation mask of an image using some state of the art image segmentation model. If we perturb a single factor in the image, say hair, their should be a major change in just one dimension and others

should be almost invariant. If the some features are correlated, there may be a chance of observing changes in some other dimensions too. In an ideal case, if the representations are perfectly disentangled, then empirical variance in all dimensions other than the perturbed one should be zero. This can be used as a metric to measure the quality of disentanglement. Moreover, a similar but opposite argument can also be made. Instead of perturbing just one dimension, one can fix one segmentation mask and perturb all other things. In this case, the empirical variance along the fixed dimension should be zero and others may vary. The method is shown with the help of a flowchart in Figure [6]

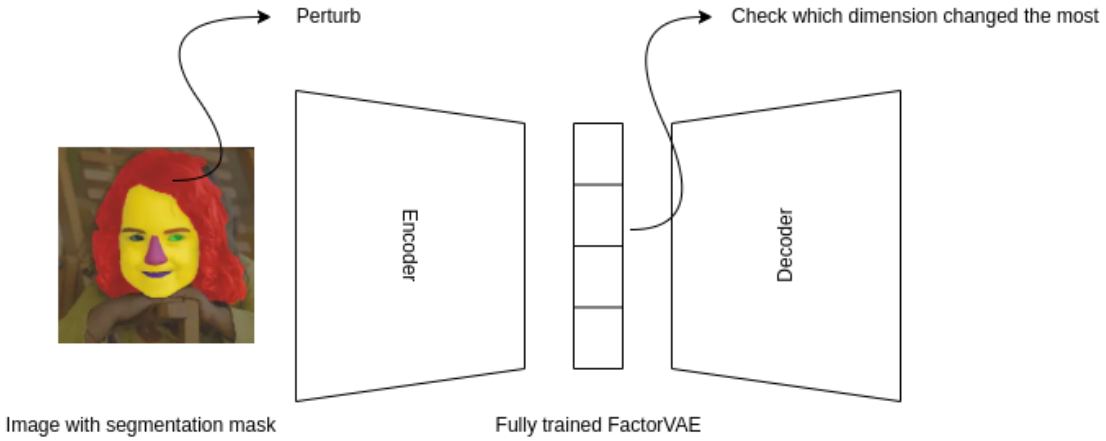

Figure 6: Flowchart for a possible way to measure the quality of disentanglement

3.2 Predicting Solution to NAR Problem using Genetic Algorithm

A possible extension to this work is to predict the solution of the NAR problem. Achieving disentanglement ease this task down by disentangling the representation that helps in searching for solution in the latent space. The proposed method is to find the appropriate perturbation that should be added to the latent dimension that will reconstruct the solution. As the Autoencoder is trained to encode all the images in the training set and create a NAR manifold, the solution is also part of the manifold. Since, searching a n-dimensional space is impractical, using genetic algorithm to search, bring down the number of iterations needed to reach at the optimal solution. Initially, the population will be generated with random vector of size latent dimension. These are the perturbations that will be added to the latent space. Then after adding these perturbations, we'll reconstruct the images. The images that are nearer to one of the options, the fittest, will be chosen to mate and generate off-springs. Iteratively performing this for several iteration will yield the fittest chromosome which can be added to the latent dimension to generate the correct solution.

3.3 Multivariate Normal Approximation to marginal posterior

We can get the approximate marginal posterior by the monte carlo estimate as follows:

$$q(z) = E_{p_{data}(x)}[q(z|x)] = \frac{1}{N}\sum_{i=1}^{N} q(z|x^{(i)})$$

where,

$$q_\theta(z|x) = \prod_{j=1}^{d} \mathcal{N}(z_j|\mu_j, \sigma_j^2)$$

If we assume the prior $p(x)$ to be normal, we can approximate q(z) to be approximately multivariate normal whose mean and covariance matrix can be predicted by a neural network. Since we want the distribution $q(z)$ to be factorial, we can try to minimize the KL-divergence between the multivariate normal and the multivariate standard normal. We can then train the modified Factor VAE with the tweaked loss.

References

[1] Kim, H., and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning* (2018), PMLR, pp. 2649–2658.

[2] Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset, 2017.

[3] Sonwane, A., Shroff, G., Vig, L., Srinivasan, A., and Dash, T. Solving visual analogies using neural algorithmic reasoning. *CoRR abs/2111.10361* (2021).

[4] Yang, K., Zhou, T., Zhang, Y., Tian, X., and Tao, D. Class-disentanglement and applications in adversarial detection and defense. In *Advances in Neural Information Processing Systems* (2021), A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds.