Disentangling Shape and Orientation with Affine Variational Autoencoders

Rene Bidart
Alexander Wong
{rbbidart, a28wong}@uwaterloo.ca

Systems Design Engineering, University of Waterloo
Systems Design Engineering, University of Waterloo

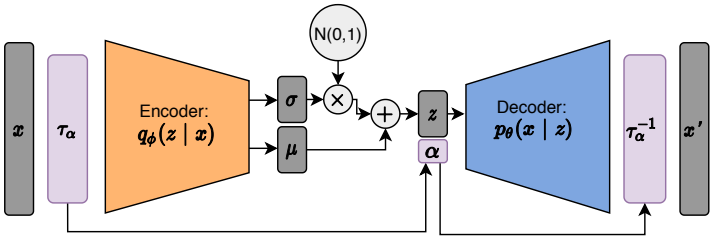

Fig. 1: Affine Variational Autoencoder (AVAE) extends the conventional VAE by introducing two affine layers, the first performing an affine transform to the input, parameterized by α. This is encoded and decoded by the VAE, and finally the second affine layer does the inverse transform, producing the final output, x'.

Abstract

Is it be possible to disentangle an object's orientation from its shape? In this work we create compressed representations of an object by disentangling its orientation and shape with a variational autoencoder augmented with affine transform layers. Even when trained on randomly oriented data, shape and orientation are disentangled during training while the model learns to encode objects at a fixed orientation. We show this process results in a more compressed latent representation for 2d digits on the MNIST dataset, and for 3d objects on the ModelNet dataset.

1 Introduction

There has been recent interest in learning representations of data where factors of variation in the data are separated, called disentangled representations [1], with the aim of increasing interpretability and robustness. Unfortunately the general problem of learning disentangled representations is impossible without assumptions [2], similarly to how there is no classifier to rule them all [3]. Luckily just as we can construct useful classifiers by making assumptions about the structure of the world, we can also make assumptions to learn disentangled representations.

We focus on the restricted problem of learning a disentangled representation of objects' orientation and shape. Previous works have focused on closely related questions using a variety of approaches, including predicting the orientation of an object, both for feed forward networks [4] and variational autoencoders (VAE) [5].

This work differs by learning disentanglement as side effect of creating compressed representations. We extend previous work[6] where the Affine Variational Autoencoder (AVAE) architecture was introduced as a way to create more efficient representations of images. to the 3D case and show in addition to creating a more compressed representation, these representations are also disentangled. For both 2d images with the MNIST [7] dataset and 3d objects with the ModelNet dataset [8] we show the AVAE can disentangle object shape and pose.

1.1 Variational Autoencoders

Variational Autoencoders [9] are generative models where it is assumed that the data, $X = \{x\}_{i=1}^n$ are generated from latent variables, z, with a prior distribution, $p_\theta(z)$, as a centered isotropic multivariate Gaussian. The likelihood, $p_\theta(x|z)$ is assumed to be a multivariate Gaussian, and the posterior, $q_\phi(z|x)$ is assumed to be Gaussian with diagonal covariance. The parameters of the likelihood and posterior are represented with neural networks.

The VAE is trained to maximize likelihood of the data by maximizing the evidence lower bound (ELBO) shown in equation 1. This entire network is differentiable, so can be trained using stochastic gradient descent with this loss.

$$-L_{VAE} = E_{z\sim q_\phi}[log p_\rho(x|z)] - KL(q_\phi(z|x)||p_\rho(z)) \quad (1)$$

1.2 Disentangled Representations

Even though in general it is impossible to learn disentangled representations without making some assumptions about the data [2], there has been extensive work on problems using relatively weak assumptions. These include learning disentangled representations where semantically relevant variables are explicit in the latent space [10]. These are not limited to affine transforms, and include variations such a lighting, color, or physical attributes like shape. One approach is based on semi-supervised learning, where images are generated based on both a latent variable and some relevant factor of variation, which are assumed to be independent [11]. For face generation, disentangling shape and appearance was tackled through the synthesis of appearance on a template followed by a deformation [12]. Other work divides the latent space into explicit and implicit factors of variation, and training process of varying only one factor while fixing the others is used to enforce the disentangled latent space [13]. These methods all require supervised inputs, where they are labeled based on some factor of variation.

1.3 Affine Transforms

Spatial Transformer Networks (STN) [4] aim to transform images to some canonical orientation by applying an affine transform to the input image using a differentiable three stage process. First, a (1) **Localization Network** predicts affine transformation parameters. Next, a (2) **Sampling Grid** of coordinates is used to associate each point in the input to the output. Finally (3) **Bilinear sampling** is used to apply the sampling grid.

Our work uses the Sampling Grid and Bilinear Sampling steps for the affine transform layers, but instead of the localization process the parameters are determined through an optimization process.

Other work [5] has also shown invertible transformations in VAEs can be used to disentangle appearance and perspective but used transformation parameters inferred from the object, instead of the optimization process used in our work.

It is also possible to create networks that are equivariant to one specific factor of variation, for example, constructing deep convolutional neural networks that are equivariant to rotation and reflection [14]. While this is an effective method, it is limited to 90° rotations, and adding more factors of variation increases the complexity dramatically. Other work has focused on rotations specifically using polar coordinates [15], but this is also limited to rotations only.

2 Model

2.1 Affine Variational Autoencoder

Similar to the VAE, we assume the data $\boldsymbol{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$ are generated by an unobserved latent variable z. In this case we assume the prior distribution of $\boldsymbol{z} = [z_1, z_2, ..., z_k]$ can be written as:

$$p(\boldsymbol{z}) = p(z_1, z_2, ..., z_p, z_{p+1}, ..., z_l) = p(z_1, z_2, ..., z_p)p(z_{p+1})...p(z_l) \quad (2)$$

We then relabel $p(\boldsymbol{z})$ as $p(z_1, z_2, ..., z_p)p(\alpha_1)...p(\alpha_k)$. We consider $\boldsymbol{z}$ to be latent variables representing shape, $\boldsymbol{\alpha}$ to be those representing orientation, and assume these are independent.

With a fixed orientation, $\boldsymbol{y}$, this would further simplify the problem because we could model the conditional distribution $p(z_1, z_2, ..., z_p)p(\alpha_1 = y_1)...p(\alpha_k = y_k) = p(z_1, z_2, ..., z_p)$. But this is limited to a single orientation, and we would like a model for all possible orientations.

Assuming we have a standard VAE to model $p(x, \alpha = y)$ for the distribution with fixed orientation, how could we make it generalize to more orientations?

We would like a transform, τ_α that can transform an object $\boldsymbol{x}$ to the correct orientation for our VAE, $\boldsymbol{y}$, along with a corresponding

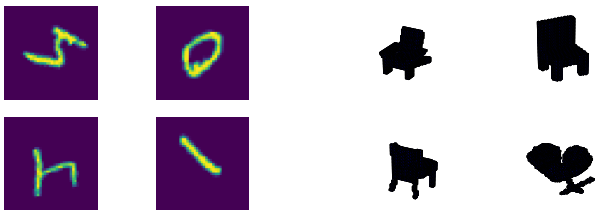

Fig. 2: 2d MNIST and 3d ModelNet object datasets, both shown with rotation augmentation

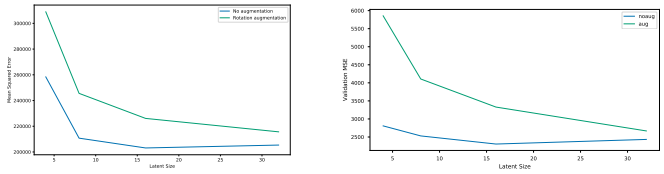

Fig. 3: **(a)**: Reconstruction error of the VAE with and without rotation augmentation. MSE is greater for the rotation augmented data with any latent size, and the difference is most significant with limited latent capacity.
(b): Average MSE of a VAE over the ModelNet validation set (sofa class) for different latent sizes, compared to the same dataset with rotation augmentation. For both classes, performance is lower on rotation augmented data, indicating rotation augmented data has a more complex distribution.

inverse transform τ_α^{-1} to transform the object back to its original orientation. This is an affine transform, so both of these are straightforward to implement, and can be added to the standard VAE, resulting the AVAE shown in figure 1. But how do we know the transform?

Because the VAE is trained by maximizing a lower bound on the log-likelihood of the data, for an object $\boldsymbol{x}$, we will use the loss to approximate $p(x)$. Assuming the VAE was trained on a distribution with fixed orientation $\boldsymbol{y}$, samples at any other orientation should have low $p(x)$. To find the correct orientation, we should optimize the transform parameters, α to minimize the VAE's loss. Given a VAE with encoder q_ϕ, and decoder p_ρ, and the invertible transformation τ_α, we optimize α to maximize the likelihood of x under the model:

$$\alpha^\star = \underset{\alpha}{\operatorname{argmin}} \left\{ L_{VAE}[\tau_\alpha^{-1}(p_\rho(q_\phi(\tau_\alpha(x))))] \right\} \tag{3}$$

Affine transforms can be made differentiable [4], so this can be optimized using gradient descent. In practice, this optimization is difficult and likely to be caught in a local optima, so random restarts are required.

2.2 Disentangling Orientation and Shape

Given a dataset $\boldsymbol{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$ distributed randomly over affine transforms, we use the observation that the conditional distribution of $\boldsymbol{z}$ given a fixed orientation, $p(z_1, z_2, ..., z_p | \alpha_1 = y_1, ..., \alpha_k = y_k)$, is less complex than than the full distribution to enable the disentanglement of orientation and shape.

We train the AVAE on this dataset, but before each step of SGD, the affine parameters are optimized using the above procedure. Given a limited capacity model, the most efficient solution is to encode the objects at a fixed orientation because this simplifies the distribution to be modelled. The goal is that as training progresses, the representation will be progressively more disentangled as the model learns to encode at a fixed orientation and the randomly oriented training data's orientation is optimized towards this orientation.

2.3 Design choices

For the MNIST dataset, we take the likelihood, $p_\theta(x|z)$, to be an isotropic Gaussian distribution with the variance fixed as 1. For the ModelNet dataset, because we take the voxels as representing the volume of an object, we assume each voxel should take values in $\{0, 1\}$. For this reason we use a Bernoulli distribution for the likelihood. For optimization of the affine transform, we use 32 random restarts. We select the 8 best parameters and perform 20 steps of gradient descent on them using the Adam optimizer [16], finally selecting the one with lowest loss.

3 Experimental Results

4 Experimental Results

4.1 Datasets

4.1.1 MNIST

The MNIST dataset is a set of grayscale numbers between 1 and 9, sized 28×28. We do standard preprocessing by normalizing pixel values and also pad the image with zeroes to a size of 40×40, to ensure there is no distortion under rotation or translation, shown in figure 2(a).

4.1.2 ModelNet

The ModelNet dataset [8] is composed of 3d voxel images of common objects, shown in figure 2(b). For this work we use the 10 class version of the dataset, and focus in particular on the sofa and chair classes. The images are padded to $48 \times 48 \times 48$ to allow for rotation and translations without distortion.

4.2 Complexity of Rotation Augmented Objects

We compare the standard VAE on rotation augmented data to one on data of a single orientation, and show for a given latent dimension loss is higher with the rotated data. This is to verify our hypothesis that the rotation augmented data is a more complex distribution so will require a higher capacity model.

For the MNIST dataset, in figure 3(a), we vary the latent capacity for the VAE while comparing rotation augmented data to the single orientation. We see that reconstruction error in terms of mean squared error is greater for the rotation augmented data for any given latent size. At smaller latent sizes this difference is most pronounced, but as latent dimension increases performance converges. This also fits with our assumption, as we would expect that in theory a very high capacity model could achieve the same performance on rotation and single orientation data, as latent capacity is not a limiting factor.

For the ModelNet dataset, in figure 3(b) we vary the dimension of the latent space used to encode the objects. We consistently see the model for all rotations has higher mean squared error (MSE) compared to the model for a single orientation, confirming the hypothesis that the distribution of rotation augmented data is more complex. This is true for all latent vector sizes, but becomes less large as the latent size increases and model capacity is less of a limitation.

We note that for both classes the difference between the rotation augmented and the single orientation data is larger than what was seen that on the MNIST dataset. The loss is twice as high when using the latent size of 4 on the ModelNet dataset compared to only 15% higher on MNIST. This is explains the larger performance benefit seen using the AVAE on the ModelNet dataset. Because rotations added greater complexity to ModelNet than MNIST, the AVAE had greater room to improve performance.

4.3 More Efficient Representations with AVAE

We aim to show for a given latent size, the AVAE outperforms the standard VAE in terms of reconstruction error, indicating a more compressed latent representation. We intentionally use models with small latent dimension to make the differences between the models more clear.

In the AVAE the latent space is explicitly separated into the affine transform parameters for affine transform layers and the shape parameters from the VAE part of the model. Because of this we have to add additional dimensions to the latent space of the standard VAE model to take make the comparison equivalent. For rotation augmented data we use a latent size of 7 which is composed of a 6 dimensional VAE and the single rotation parameter, $[r]$.

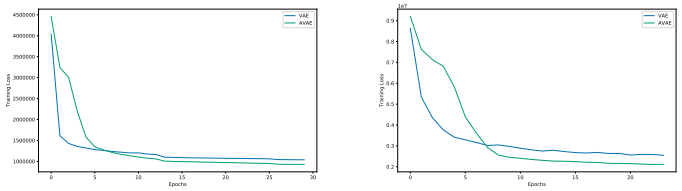

Fig. 4: **(a)**: Training of VAE vs. AVAE on the MNIST dataset with rotation augmentation. AVAE takes longer to converge to a low loss because it takes a few epochs for the model to learn encode digits at a particular orientation
(b): Training of VAE vs. AVAE on the ModelNet dataset with the sofa class, similarly to MNIST the AVAE first has higher loss before it has learned to encode data at a particular orientation

For rotation & translation augmentation we use a size of 9, which is composed of the 6 dimensional latent space, the rotation parameter and translation parameters, $[r, t_x, t_y]$.

In table 1, we see that the AVAE outperforms the standard VAE for both rotation and rotation & translation augmentation by 12% and 10% respectively. Basically, it is more efficient to explicitly use a parameter for the rotation angle which is learned by the AVAE rather than to leave it to the model to learn its own mapping.

Table 1: Improvement of AVAE over VAE on MNIST (MSE)

Augmentation	Validation set (all classes)
Rotation	12%
Rotation & Translation	10%

Table 2: Improvement of AVAE over VAE on ModelNet (MSE)

Augmentation	Sofa class	Chair class
Rotation	30%	48%
Rotation & Translation	38%	18%

We also compare the AVAE to the standard VAE on a rotation augmented version of the ModelNet dataset. Because the AVAE uses 3 additional rotation $[r_x, r_y, r_z]$ parameters, we compare the AVAE with 3 orientation parameters and 16 shape parameters to a standard VAE with 19 dimensional latent space.

We also test this procedure using rotations and translations. Here there is an additional 6 parameters, 3 for rotation and three for translation, $[r_x, r_y, r_z, t_x, t_y, t_z]$, so we compare the 16 dimensional AVAE to a VAE with latent size 22. As shown in Table 2 for both the sofa and chair classes the AVAE shows significant improvement over the standard VAE. This is also true for augmented with both rotation & translation, and here we see a significantly larger improvement compared to the MNIST dataset.

4.3.1 Disentanglement of Rotations During Training

We see evidence disentanglement during training by looking at the loss as the model is trained, as shown in figure 4. We see the standard VAE has a normal loss curve, where the loss quickly goes downward for the first few epochs, and more slowly decreases after. In contrast, the AVAE's loss remains high for the first few epochs, before it eventually decreases quickly to a lower value than the standard VAE.

This is because the AVAE is initially limited by having its' latent space divided between shape and orientation parameters. Initially the rotation optimization process is useless because the model has not learned to encode any orientation well, so the AVAE is forced to encode all orientations with its more limited latent size. As the training progresses the model learns to encode only a single orientation the loss drops quickly. By looking further into the distribution of orientations during training for a single class on the MNIST dataset we can see this disentanglement more clearly.

Using standard rotation augmentation and the normal training process for a VAE, the rotations the model is trained on should be uniformly distributed over $[0°, 360°]$. But the AVAE optimizes rotation before encoding, so this is no longer the case. Here we provide

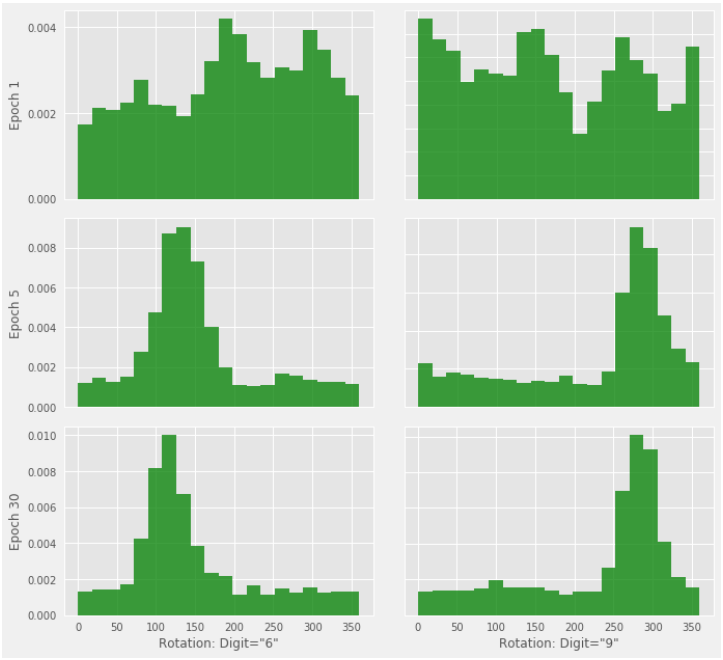

Fig. 5: Evolution of rotations encoded by the AVAE during training. Distribution of rotations of the "6" and "9" digits during training of the AVAE at epochs 1, 5 and 30. As training progresses the AVAE learns to encode most digits at the same orientation, but additionally these numbers are encoded as $180°$ rotations of one another.

evidence that optimizing the affine transform during the training process allows the AVAE to learn a more efficient representation by changing the distribution of rotations it encodes digits at.

We look at how the distribution of angles digits are encoded at evolves over the training process. At the first epoch of training the model encodes each digit at a relatively uniform distribution over rotations, as seen in the top graph of figure 5. The model hasn't learned to encode any rotation better than another, so the optimization of rotation during training is useless, returning another random distribution over rotations. This is what we would expect to see when training a standard VAE.

As training progresses, shown in the lower graphs in figure 5, the model becomes biased towards encoding digits at particular orientations. It learns that it is better to encode only a subset of the true distribution to better utilize the limited latent capacity of the model. This is consistent with our earlier observation that it takes a lower capacity model to encode a single rotation compared to all possible orientations of images.

In figure 5, we are comparing the digits "6" and "9". Because these digits have no rotational symmetries, they are encoded at a single orientation, leading to the unimodal distribution seen for both digits. But these digits are quite similar, nearly being $180°$ rotations of one another. Because we train the AVAE to encode all classes of MNIST, we see it learned a more efficient representation between digits too, encoding the "6" and "9" digits at $180°$ rotations of one another to simplify the data distribution.

5 Conclusion

In this work we proposed a method to enable learning more compressed representations of of 2d images and 3d objects by disentangling their orientation and shape. We used an optimization process to find the best orientation to encode objects at by minimizing the VAE's loss, which can be seen as approximately maximizing the likelihood of the object under our VAE. We showed this learns more compressed representations compared to a standard VAE, and this happens by disentangling orientation and shape. We evaluated this on the 2d MNIST and 3d ModelNet datasets and showed that it is useful on both types of data, but has larger benefits on the 3d data.

Acknowledgments

We thank you for reading these guidelines.

References

[1] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012. [Online]. Available: http://arxiv.org/abs/1206.5538

[2] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," *CoRR*, vol. abs/1811.12359, 2018. [Online]. Available: http://arxiv.org/abs/1811.12359

[3] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996. [Online]. Available: https://doi.org/10.1162/neco.1996.8.7.1341

[4] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015. [Online]. Available: http://arxiv.org/abs/1506.02025

[5] N. S. Detlefsen and S. Hauberg, "Explicit disentanglement of appearance and perspective in generative models," *arXiv preprint arXiv:1906.11881*, 2019.

[6] R. Bidart and A. Wong, "Affine variational autoencoders," in *Image Analysis and Recognition*, F. Karray, A. Campilho, and A. Yu, Eds. Cham: Springer International Publishing, 2019, pp. 461–472.

[7] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[8] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[9] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ArXiv e-prints*, Dec. 2013.

[10] K. Ridgeway, "A survey of inductive biases for factorial representation-learning," *arXiv preprint arXiv:1612.05299*, 2016.

[11] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, 2014, pp. 3581–3589.

[12] Z. Shu, M. Sahasrabudhe, R. Alp Guler, D. Samaras, N. Paragios, and I. Kokkinos, "Deforming autoencoders: Unsupervised disentangling of shape and appearance," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 650–665.

[13] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in neural information processing systems*, 2015, pp. 2539–2547.

[14] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, 2016, pp. 2990–2999.

[15] K. S. Tai, P. Bailis, and G. Valiant, "Equivariant transformer networks," *CoRR*, vol. abs/1901.11399, 2019. [Online]. Available: http://arxiv.org/abs/1901.11399

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.