Affine Variational Autoencoders: An Efficient Approach for Improving Generalization and Robustness to Distribution Shift

Rene Bidart [1 2] **Alexander Wong** [1 2]

Abstract

In this study, we propose the *Affine Variational Autoencoder (AVAE)*, a variant of Variational Autoencoder (VAE) designed to improve robustness by overcoming the inability of VAEs to generalize to distributional shifts in the form of affine perturbations. By optimizing an affine transform to maximize ELBO, the proposed AVAE transforms an input to the training distribution without the need to increase model complexity to model the full distribution of affine transforms. In addition, we introduce a training procedure to create an efficient model by learning a subset of the training distribution, and using the AVAE to improve generalization and robustness to distributional shift at test time. Experiments on affine perturbations demonstrate that the proposed AVAE significantly improves generalization and robustness to distributional shift in the form of affine perturbations without an increase in model complexity.

1. Introduction

While deep neural networks have been shown to be extremely powerful, they are often quite fragile and do not generalize well to distributional shifts without explicit methods to account for such shifts. A particular form of distributional shift that has seen recent attention are affine data perturbations, where out-of-distribution samples can be treated as affine transformed variants of samples within the training distribution. As an example, Fig. 1 shows examples of the inability of a variational autoencoder (VAE) to encode and decode images after various rotational perturbations.

Existing methods for improving robustness to affine perturbation have limitations, including data augmentation which forces an increase in model complexity (Section 4.1), explicit equivariance which results in an increase in memory use and is difficult to scale to more transforms(Cohen & Welling, 2016), or are approximate methods, where the model must estimate a transformation to bring the image back to a canonical orientation (Jaderberg et al., 2015).

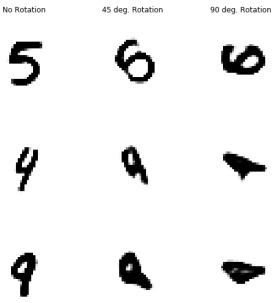

Figure 1. Examples of reconstructed images after various rotational perturbations from a VAE trained on MNIST. The quality of the reconstructed images degrade significantly under increasing rotational perturbation.

In this study, we introduce the *Affine Variational Autoencoder (AVAE)*, an extension of the Variational Autoencoder (VAE) designed to improve generalization and robustness to distribution shift related to affine perturbation. We take an alternative approach by explicitly leveraging VAE's loss to evaluate if a sample is within the training distribution, with the corresponding affine transform performed within the AVAE optimized to reduce this loss. Instead of increasing model complexity as used in data augmentation, we increase computational complexity to transform a sample back within the distribution the model was trained on, with the added benefit of returning the affine transform needed to bring the sample back to the training distribution.

More specifically, we introduce a method for enabling pre-trained VAEs, which follow a certain training distribution, to generalize to the full set of affine transforms, and a way to train a model on the full distribution, without increasing the model capacity. Given some data X, we consider the full set of possible affine transforms of X to be X_{full}, and a subset of these transforms as X_{sub}. These methods are:

1. A module added to a pre-trained VAE trained on X_{sub}, allowing for improved generalization to samples from X_{full} through optimization of an affine transform
2. A training procedure to allow the AVAE to be trained on the full dataset, X_{full}, by optimizing the affine

[1]Systems Design Engineering, University of Waterloo, Waterloo, ON, Canada [2]Waterloo Artificial Intelligence Institute, Waterloo, ON, Canada. Correspondence to: Rene Bidart <rbbidart@uwaterloo.ca>.

Presented at the ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning.

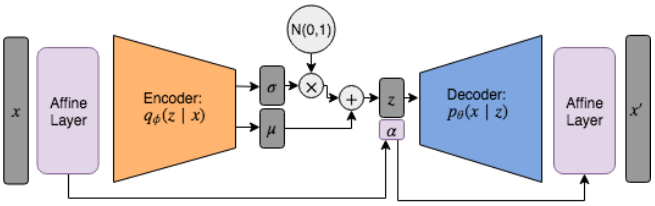

Figure 2. Affine Variational Autoencoder (AVAE). This extends upon the conventional VAE by introducing two affine layers. The first affine layer performs a learned affine transform to the input, parameterized by α. This is encoded and decoded by the encoder and decoder, respectively, and finally the second affine layer performs the inverse transform, producing the final output, x'.

transform for each batch during training, allowing a model to be trained on the full dataset without forcing an increase in model capacity.

2. Background and Related Work

2.1. Variational autoencoders

Variational autoencoders (Kingma & Welling, 2013) are generative models where it is assumed that the data, $X = \{x\}_{i=1}^{n}$ are generated from latent variables. An encoder and decoder network are simultaneously trained, where the encoder models the posterior distribution of latent variables z, given an image x as $q_\phi(z|x)$, and a decoder that gives the distribution of data (in this case, an image) conditioned this latent variable. It is assumed that the latent variables are independent standard normal.

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

The objective of the VAE is to maximize the evidence lower bound (ELBO) shown in equation 1. In many cases, convolutional neural networks (CNNs) are used to approximate both p and q. This entire network is differentiable, so can be trained using stochastic gradient descent with this loss.

2.2. Generalization to Affine Transforms

There have been many attempts to both learn representations that are robust to distributional shifts under a set of transformations as well as increasing interpretability by encoding these transforms explicitly in the latent space for uses such as image classification, compression, and generation.

Spatial Transformer Networks (STN) (Jaderberg et al., 2015) apply an affine transform to the input image, transforming it to some canonical orientation. This can be described in terms of three components:

1. **Localization Network**: This is a neural network taking the input image and outputting the affine transformation parameters, $\alpha \in \mathbb{R}^6$ to be applied.
2. **Sampling Grid**: Given an affine transform, the grid of coordinates in the input associated with each point in the output
3. **Data sampling**: Given the grid, use bilinear sampling to apply it to the input.

More generally, there has been research on learning disentangled representations where semantically relevant variables are explicit in the latent space (Ridgeway, 2016). In general 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 (Kingma et al., 2014). For face generation, disentangling shape and appearance was tackled through the synthesis of appearance on a template followed by a deformation (Shu et al., 2018). 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 (Kulkarni et al., 2015). These methods all require supervised inputs, where they are labeled based on some factor of variation. Other work has created networks that are equivariant to one specific factor of variation, for example, constructing deep convolutional neural networks that are equivariant to rotation and reflection (Cohen & Welling, 2016). While this is an interesting method, adding more factors of variation in this way increases the complexity dramatically, so is difficult to scale.

3. Affine Variational Autoencoders

The proposed notion of affine variational autoencoders (AVAEs) can be described as follows. For a given input x, we can estimate how far it is outside the training distribution, X_{sub} using the ELBO, as shown in Equation 1. In this work, we focus on out-of-distribution samples that can be treated as affine transform variants of in-distribution samples. Therefore, given a sample $x \notin X_{sub}$, we formulate the problem as transforming x to $x_A \in X_{sub}$ by optimizing the affine transform to reduce the VAE's loss. This approach is useful because the model capacity required for a subset of the full distribution, X_{sub} is less than that required for a model expected to generalize to the full distribution X_{full}. This procedure enables us to trade-off model complexity for computational complexity of performing this optimization.

3.1. Model

The AVAE extends upon the conventional VAE architecture with the introduction of two affine transform layers, before and after the conventional VAE, as shown in Fig. 2. An input sample is first fed into first affine layer, which performs an affine transform before passing it into the encoder for latent space representation. The output of the decoder in the AVAE is fed into a second affine layer, which performs an inverse affine transform on the output of the decoder, producing the final output. The parameters of the affine transform applied to the input sample are appended to the latent space and later used in the final, inverse affine layer. The affine layers are implemented similarly to the STN (Jaderberg

et al., 2015) (see section 2.2), in the form of a sampling grid and data sampling, but instead of learning the affine transform parameters through a localization network, we optimize these parameters to reduce VAE loss.

3.2. Optimization and Training

Given an input sample x, the objective of the conventional VAE is to learn an encoder q_ϕ, and decoder p_ρ to maximize the likelihood of the data, by maximizing the ELBO as indicated in section 2.1. In the proposed AVAE, the addition of the affine layers allows us to optimize a transform, $A(x) = x_A$, to reduce the VAE loss of $A^{-1}(p_\rho(q_\rho(x_A)))$, maximizing the probability of this image being from the training distribution. More specifically, in the AVAE, to encode a sample we must learn α that maximizes the ELBO in order to find the optimal affine transformation to apply to the input sample, which we can formulate as:

$$\underset{\alpha}{\text{argmin}} \left\{ L_{VAE}[A^{-1}(p_\rho(q_\phi(A(x))))] \right\} \tag{2}$$

where A refers to the first affine layer, A^{-1} refers to the second affine layer, both taking α as parameters. Note that VAE parameters q_ϕ, p_ρ remain unchanged during this.

This network is differentiable, so it is possible to take the derivative of the loss w.r.t. the affine transform parameters, α, so this can be optimized using stochastic gradient descent. We find in practice there are issues with the optimizer being caught in local minima, so we use multiple random restarts, where we first try the loss at a set number of affine parameters, and only perform gradient descent on the best performing parameters.

Given a set of samples at some canonical orientation, such as MNIST, the AVAE can be trained identically to the conventional VAE, or the affine layers can be added to an existing pre-trained model. However, an issue with training an AVAE in an identical manner as a conventional VAE is that the benefits of generalization and robustness to distributional shift provided by the AVAE are diminished if the training data isn't already in a canonical orientation like in MNIST. Training on X_{full} instead of X_{sub} will require a greater capacity model, as indicted in section 4.1.

3.3. Transformation Optimization During Training

To preserve the benefits of the AVAE when the dataset spans the full set of possible affine transforms, we introduce an alternative training scheme where we force the model to learn to encode the images well at only a subset of the possible transformations, limiting the model capacity required. This enforcement to a subset of possible transformations is accomplished indirectly by iteratively optimizing the affine transform for each batch.

More specifically, we optimize the affine transform for each input sample before performing gradient descent on the encoder and decoder parameters during training. This has the effect of transforming the training data to have similar orientational properties, reducing the number of possible transformations that the model must learn to encode and enabling one to reap the benefits of generalization and robustness to distributional shift provided by the AVAE.

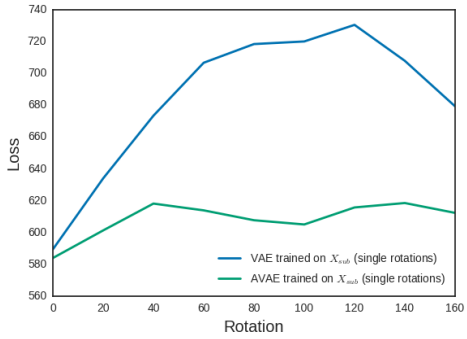

Figure 3. Average loss of VAE and AVAE over the MNIST validation set under varying rotational perturbations. Both models were trained without data augmentation, so conventional VAE does not generalize well to new rotational perturbations as the input data deviates from the training set. The AVAE is robust to this distribution shift, as shown by the relatively flat loss across perturbations.

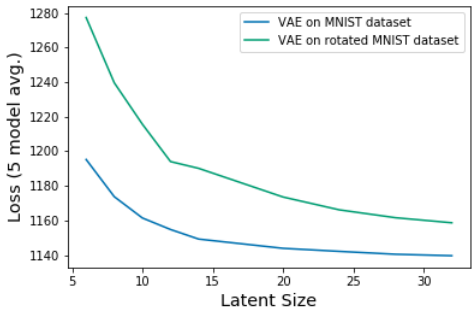

Figure 4. Comparison of the loss associated with VAEs trained on standard MNIST dataset, and the dataset of all rotational perturbations. A larger latent size is required to generalize to the rotated dataset with comparable loss. This is loss averaged over 5 models.

4. Experiments

To evaluate the efficacy of the proposed AVAE, we study its robustness to distributional shifts under various affine perturbation. We focus mostly on a specific type of affine perturbation in the form of rotational perturbation to study the behaviour of AVAEs in greater depth. Results for more general affine perturbations is also presented.

4.1. Limitations of VAE and Data Augmentation

VAEs do not generalize to out-of-distribution samples that are variants of in-distribution samples under rotational perturbations unless they are explicitly trained on this type of data. Fig. 3 shows the loss associated with a VAE encoding and decoding the MNIST validation set under different rotational perturbations. The performance of a conventional VAE decreases steadily as the rotational perturbations deviate more from the training set, with the loss reaching a maximum around $120°$. It decreases after this because many digits are similar when rotated $180°$.

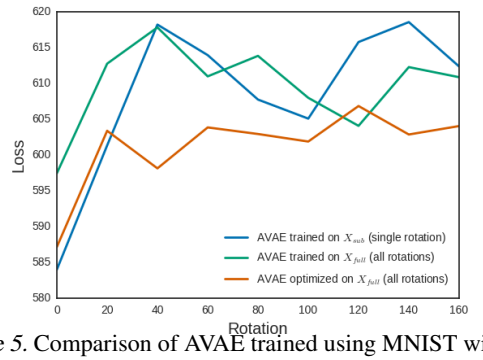

Figure 5. Comparison of AVAE trained using MNIST with no rotation augmentation (X_{sub}), trained on rotation augmentations (X_{full}), and trained on rotation augmentations with the proposed transformation optimization during training. Training with transformation optimization generally outperforms the other approaches.

The common approach to dealing with this limitation is data augmentation, directly training the model on X_{full}, but to train a VAE on a more diverse dataset requires a more complex model. We investigate this by altering the latent size in the model, as shown in Fig. 4. This could be partially avoided through the use of a more complex model instead of larger latent size, but reinforces the assertion that it is expensive to make a VAE generalize well to perturbations via data augmentation.

4.2. AVAE on Rotational Perturbations

We first investigate if a conventional VAE can be extended into a AVAE to generalize and be more robust to rotational perturbations. Fig. 3 shows the loss is reduced across all rotational perturbations by extending a conventional VAE into an AVAE. The increase in performance is greatest at the rotational perturbations that are furthest from the original training distribution, and the models perform similarly when rotational perturbations are close to the standard orientation the data was trained on, $0°$.

Table 1. VAE and AVAE average validation loss over $0° - 180°$ rotational perturbations on MNIST

Model	Training Data	Avg. Loss
AVAE (trans. opt.)	random perturbations	601.2
AVAE	single orientation	608.6
AVAE	random perturbations	609.8
VAE	single orientation	684.4

4.3. Transformation Optimization During Training

Earlier we speculated that optimizing the affine transformation training would allow the model to encode a smaller subset of the full distribution, enabling better performance for a given model complexity, with improved generalization to the rest of the distribution during test time. In appendix 1 we show that empirically this is the case, because as the model trains it learns to encode a progressively smaller portion of the original dataset.

Figure 6. Examples of the reconstructions of images transformed by the a random affine transformation from the VAE (top) and AVAE (bottom).

In Table 1, we can see that over all rotational perturbations this seems to be true, with the AVAE using transformation optimization during training achieving the best results. The AVAE trained on a single orientation is slightly better than random perturbations because it isn't forced to encode the full dataset, but these models overall perform similarly.

Looking at Figure 5, these methods of training face a trade-off, because the rotation-augmented AVAE is better for encoding out-of-distribution samples, but the single orientation AVAE should perform better once the samples are brought into the correct orientation. Because the optimization process at test time isn't perfect, the rotation augmented AVAE still has some benefits for these out-of-distribution samples. The AVAE with transformation optimization during training effectively balances both these considerations, and based on Figure 5 it is the superior model at almost all rotations.

4.4. General Affine Transforms

We also investigated the generalization of AVAE to inputs perturbed by more general affine transforms through the use of the AVAE's optimization process. Inputs we perturbed by random rotations, random shears up to $55°$, as well as random scaling by up to 50%. Based on these transforms, we found the AVAE was able to decrease the validation loss by 14% compared to the conventional VAE. An example of this is shown in Figure 6.

5. Conclusion

In this study, we introduced affine variational autoencoders, extending upon VAEs to improve generalization and robustness to distributional shifts due to affine transformations by optimizing the affine transform based on the VAE loss. In experiments using rotational and general affine perturbations, it was shown that the proposed AVAEs can indeed improve robustness in practice. This shows a practical trade off between optimization and model capacity, where to generalize to affine transforms we can substitute increased model capacity with this optimization procedure. In addition, we introduced a training procedure that forces the model to learn only a subset of the possible affine transformations, allowing the AVAE to be trained on any dataset.

References

Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. URL http://arxiv.org/abs/1511.07289.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. URL http://arxiv.org/abs/1506.02025.

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

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

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

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

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

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

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on mnist. *arXiv preprint arXiv:1805.09190*, 2018.

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

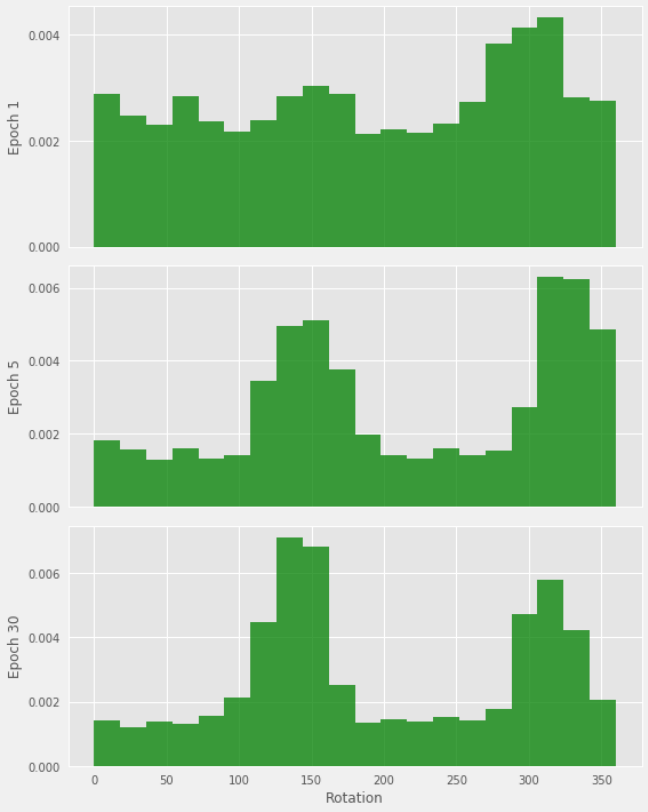

Figure 7. Distribution of rotations of the "1" digit during training of the AVAE at epochs 1, 5 and 30. We can see as training progresses the AVAE learns to encode most digits at the same orientation as the most dense representation of the data.

A. Appendix

A.1. Transformation Optimization During Training

Under the normal training process, the distribution of rotations the model is trained on should be uniformly distributed over $[0°, 360°]$. Through the optimization process during training, we find that the AVAE learns a more efficient representation where the digits of a given class are oriented at the same rotation. For the digit "1", the model learns to encode at two orientations $180°$ apart because the "1" is almost identical when under this rotation, as shown in Figure 7.

The model learns to encode the "6" and "9" digits as $180°$ rotations of one another, as this is the most compressed representation of the data. This is shown in Figure 8.

A.2. Implementation

All experiments were implemented using Pytorch (Paszke et al., 2017), and optimization was done using the Adam Optimizer (Kingma & Ba, 2014) with the learning rate set to 0.001, weight decay set to 0.0005 and batch size of 256. The MNIST dataset was normalized by mean and standard deviation and 0 padded to 40x40 pixels.

We use a VAE architecture based on a previously successful

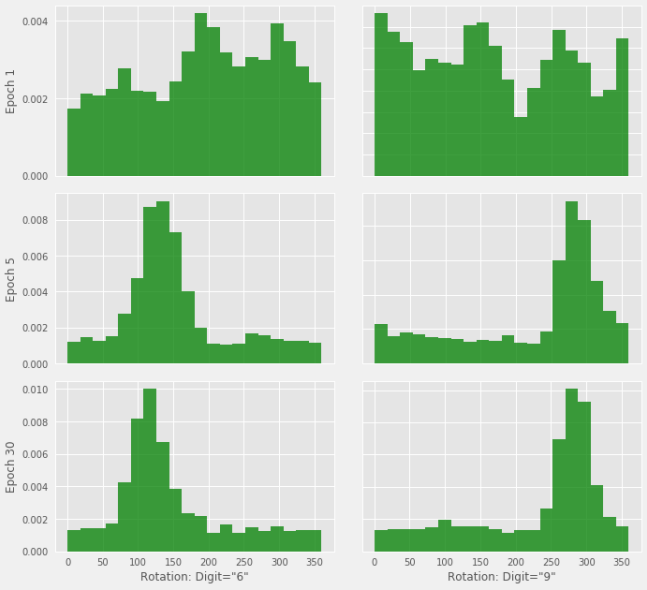

Figure 8. Distribution of rotations of the ”6” and ”9” digits during training of the AVAE at epochs 1, 5 and 30. We can see as training progresses the AVAE learns to encode most digits at the same orientation, but additionally these numbers are encoded as 180° rotations of one another.

implementation(Schott et al., 2018), which uses and encoder composed of four convolutional layers of sizes [32, 32, 64, 16], and a decoder composed of transposed convolutions of sizes [32, 16, 16, 1]. For the first three layers of both the encoder and decoder, Exponential Linear Unit activation functions (ELU)(Clevert et al., 2015) are used as well as batch normalization(Ioffe & Szegedy, 2015).