Latent Constraints for Data Generation

Chandramouli Rajagopalan
Electrical and Computer Engineering
A59020801

Divya Seshadri
Electrical and Computer Engineering
A59019510

Abstract

Dataset augmentation is the process of applying simple and complex transformations to help overcome the increasingly large requirements of Deep Learning Models, it is a standard procedure in almost every supervised learning task. We try to approximate the set of transformations to which a particular label is invariant and improve the generalization using learned data augmentation. To attain quicker results, we constrain our dataset to CIFAR-10[1] and MNIST[2] and compare a set of data augmentation techniques. The potential benefit of such data augmentation is that it can include viewpoint changes, changes in scene lighting, and many nonlinear transformations. We propose a method to learn the data augmentations on the observations to obtain improved generalization. Using the pre-trained VAE and posthoc learning latent constraints, value functions that identify regions in latent space that generate outputs with desired attributes, we can conditionally sample from these regions with gradient-based optimization or amortized actor functions. Combining attribute constraints with a universal "real-ism" constraint, which enforces similarity to the data distribution, we generate realistic conditional images from an unconditional variational autoencoder. Finally, we validate the generalization of the trained VAE by training a classifier on the generated data.
Git Repo: `https://github.com/gitttt-1234/ECE176_Project.git`

1 Introduction

Generative modeling of complex data such as images, video, and audio is an unsolved problem in machine learning and artificial intelligence. These techniques help generate data upon which large models are trained to achieve state-of-the-art performance on the respective tasks. Dataset augmentation and generation have been a primary issue while training models as manual labeling is laborious, time-consuming, and inefficient. Unconditional sampling is an interesting open-ended technical problem, it is arguably of limited practical interest in its own right: if one needs a non-specific image, one can simply pull something at random from the unfathomably vast media databases on the web. Similar solution doesn't work with conditional sampling as more and more attributes/constraints are specified for the data. It becomes exponentially less likely that a satisfactory example can be pulled from a database. One might also want to modify some attributes of an object while preserving its core identity. These are crucial tasks in creative applications, where the typical user desires fine-grained controls. One can enforce user-specified constraints at training time, either by training on a focused subset of data or with conditioning variables. These approaches can be ineffective at times due to the manual need to label data everytime a model needs to be trained on a different task.

Deep latent-variable models, such as Generative Adversarial Networks [3] and Variational Autoencoders (VAEs; [4]), learn to unconditionally generate realistic and varied outputs by sampling from a semantically structured latent space. We hope to leverage that structure in creating new conditional controls for sampling and transformations [5]. Here, we show that new constraints can be enforced post-hoc on pre-trained unsupervised generative models Fig.1. This approach removes the need to retrain the model for each new set of constraints, allowing users to more easily define custom behavior. We separate the problem into (1) creating an the unsupervised model that learns how

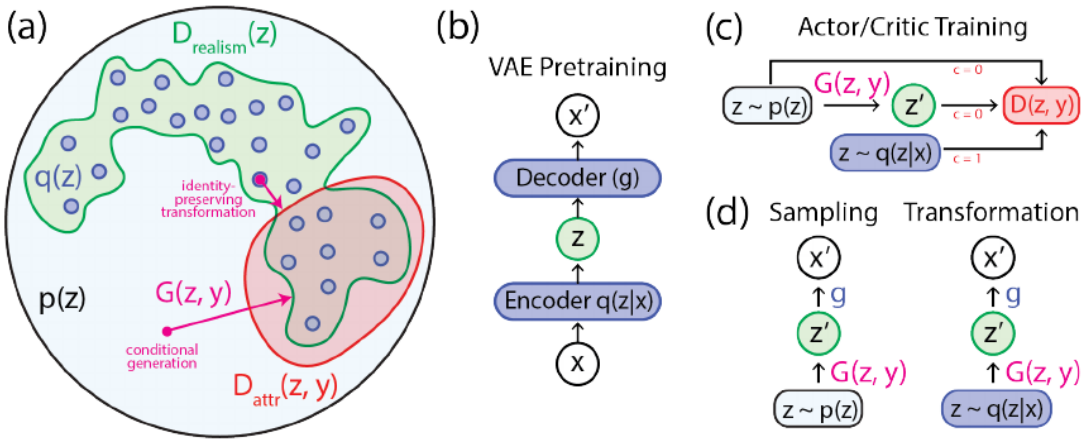


Figure 1: Diagram of latent constraints for a VAE.

to reconstruct data from latent embeddings, and (2) leveraging the latent structure exposed in that embedding space as a source of prior knowledge, upon which we can impose behavioral constraints. We plan to train an unsupervised Variational Autoencoder [5] on MNIST and CIFAR10 datasets. Once the unsupervised model learns to reconstruct data from the latent embeddings, we show that it is possible to generate conditionally from an unconditional model (VAE), learning a critic function $D(z)$ in the embedding space and generate high-value samples with gradient-based optimizations or actor function $G(z)$. Using samples from the VAE to generate exemplars, we can learn an actor-critic pair that satisfies user-specified rule-based constraints in the absence of any labeled data. We finally then use the images generated using the constraints to train a classifier and evaluate the performance of the classifier against a base classifier. Generating images using the constraints on the latent embeddings is beneficial since it does not require any additional labeled data to train on new tasks. As the deep latent network is unsupervised it can be conditioned to perform on various user-specific attributes. These constraints help achieve realistic and conditional samples by varying the parameters of the constraints.

2 Related Work

There are several methods to generate new data from an existing dataset. Generative Adversarial networks[3] learns to unconditionally generate new data with same statistics as that of the input dataset. However, GANs suffer from the modecollapse problem, where the generator assigns mass to a small subset of the support of the population distribution. Conditional GANs[9] generates samples conditioned on attribute information when available, but they must be trained with knowledge of the attribute labels for the whole training set and suffer from problems like blurriness. [6] provides a method to generate samples conditionally from a pre-trained unconditional model using a actor-critic model.

3 Method

We plan on executing the experiments in ***three*** phases.

For the ***first phase***, VAE is trained on the concerned dataset to reconstruct the images from the latent embeddings based on the ELBO loss given below Eqn.1. During this phase, the encoder and decoder part of the VAE is trained to approximate the latent space and for regeneration. We freeze the weights of the encoder and decoder for further training procedures.

$$-\mathcal{L}_{\text{VAE}} = \log p_\theta(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x}) \tag{1}$$

During the ***second phase***, we take the latent embeddings of the VAE to train the critic and actor function based on the loss functions 2 and 3 to generate latent embeddings that closely match the

constraints specifications. The actor is trained to generate samples based on randomly generated latents and the class labels whereas the critic tries to identify the generated latents against the sampled latents. The actor and critic models are trained in such a way that they try to compete against one another in a zero-sum game. The actor tries to perfect the generation of realistic image samples from the random latents and the class labels while the critic tries to classify the generated samples against the real samples. The actor and critic function use the encoder and decoder trained from the previous phase to encode and decode the image samples.

$$\mathcal{L}_{\mathrm{A}}(z) = \mathbb{E}_{z\sim q(z|x)}\left[\mathcal{L}_{c=1}(z)\right] + \mathbb{E}_{z\sim p(z)}\left[\mathcal{L}_{c=0}(z)\right] + \mathbb{E}_{G(p(z))}\left[\mathcal{L}_{c=0}(z)\right] \tag{2}$$

$$\mathcal{L}_{\mathrm{C}}(z) = \mathbb{E}_{z\sim p(z)}\left[\mathcal{L}_{c=1}G(z)\right] + \lambda_{dist}\mathcal{L}_{dist}\left(G(z), z\right)(z)\right] \tag{3}$$

For the ***final phase***, we use the images generated using the constrained latent to train a classifier on the dataset based on loss function 4 and validate its performance against a similar classifier trained without any augmentations performed on the images. We replace few images from each batch by the images generated by actor for each class label and train the classifier. We also try to test the classifier against out-of-distribution dataset in order to validate the performance when there is a change in distribution between training and test dataset. The classifier's architecture is kept similar to the actor's architecture and is trained on the samples.

$$\mathcal{L}_{\text{class}} = -\frac{1}{m} \cdot \sum_{i=1}^{m} y_i \cdot \log(\hat{y_i}) \tag{4}$$

These experiments demonstrate a method to train models without much need for manually labeled data in order to generate diverse image data.

4 Architecture

We have attached the model architectures at the end in Section.8 **Appendix** for continuity purposes.

Architecture of VAE

Fig.14 illustrates the architecture of the variational auto-encoder used for CIFAR-10 dataset. The encoder has 6 blocks with hidden dimensions (32, 64, 128, 256, 512, 1024), and each block having a 3*3 Conv, BatchNorm and a leakyReLU layer. All Conv layers have stride = 2. The output of the encoder network has mu and sigma from the latent vector with dimension 1024 is generated using the reparameterization technique. The decoder has one linear and 5 transposeConv layers, followed by a Conv layer and Tanh function to generate the reconstructed image.

For MNIST dataset, the encoder is a series of 3 linear layers each followed by BatchNorm1d and leakyReLU. There are two layers with latent dimension 1024, for mu and sigma. The decoder has 3 linear layers and a final linear layer has the image dimension 784.

Architecture of Actor-Critic Model

Fig.15 and Fig.Fig.16 shows the architecture for the actor and critic function. The same model is used for both the datasets. The pre-trained weights from the VAE model discussed above is used to train the actor-critic model. The input to these models are the latent vectors from the vae model along with randomly generated fake latent vectors.

In the actor network, there are four parts - cond layer, dz, gate and a series of 4 Linear layers. The one-hot encoded label is given as input to the linear cond layer (dimension = 2048) and the output is concatenated with input latent vector. The output of cond layer (z) is given as input to the series of 4 linear layers. gate is the output of a linear layer with output dim as the latent dimension when z is passed to this layer, followed by sigmoid. dz is the output of a linear layer with output dim as the latent dimension when z is passed to this layer. The transformed z is weighted sum of input latent and the output from the previous layer Eqn.5

$$\hat{z} = (1 - gate) \cdot (z_{input}) + gate \cdot dz \tag{5}$$

The one-hot encoded label is given as input to the linear cond layer (dimension = 2048) and the output is concatenated with input latent vector. This vector is then fed to the critic network having a

series of 4 linear layers with 2048 outputs, each followed by a batchnorm and LeakyReLU. The last layer in the series has a single output which is passed through a sigmoid activation.

Architecture of Classifier

Fig.17 illustrates the architecture of the classifier used for CIFAR-10 dataset. The network is same as the encoder used in vae for CIFAR10, except for the dimension of output of the final layer which is equal to the num of classes in this case.

For MNIST dataset, the classifier has two linear layers with outputs 128 and num of classes respectively, each followed by batchnorm and LeakyRelu.

5 Experiments

For training and testing the models, we use two different datasets and test it's performance against the test subset of the dataset as well as the out-of-distribution dataset.

5.1 Datasets

MNIST[2]:

The MNIST dataset (Modified National Institute of Standards and Technology database) consists of a large collection of monochrome images of handwritten digits (0-9), having 60000 images for the train set and 10,000 images for the test set. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

CIFAR10[1]:

The CIFAR-10 dataset (Canadian Institute for Advanced Research, 10 classes) consists of 60,000 32*32 color images and is a subset of the Tiny Images dataset. The dataset has 10 independent classes: airplane, automobile (but not truck or pickup truck), bird, cat, deer, dog, frog, horse, ship, and truck (but not pickup truck). There are 6000 images per class with 5000 training and 1000 testing images per class. All the images contain only one prominent instance of the object to which the class refers.

Both the datasets are normalized with a mean of 0.5 and standard deviation 0.25. The CIFAR10 dataset are rescaled to 64*64.

Figure 2: MNIST dataset

Figure 3: CIFAR10 dataset

5.2 Model training

The vae, actor-critic and the classifier are trained on both the datasets. For VAE and classifier, two separate models are built for mnist and cifar10 to be considerate of the image quality and resolution, maintaining a simpler model (having only linear layers) for MNIST and having multiple conv layers for CIFAR10 dataset.

VAE training: For both the datasets, the vae was trained with Adam optimizer with the batch size 32 and learning rate as 0.005. The loss function given in Eqn.1 is used with kld weight: 0.00025.

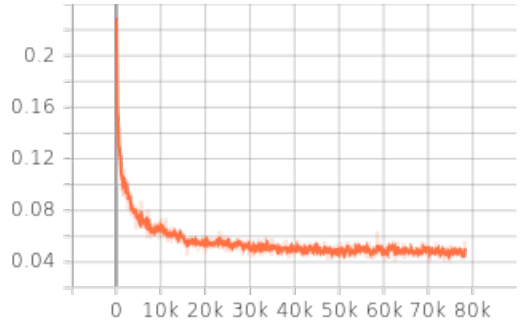
Actor critic training: For both the datasets, the models were trained with Adam optimizer with batch size 128 and learning rate 0.003. kld weight: 0.00025. The models were trained based on the loss functions mentioned in Eqn.2 and Eqn.2.

Classifier training: For both the datasets, batch size was set to 32 and learning rate as 0.005 and the classfier was trained with Adam optimizer for 50 epochs. While training the classifier, in every training batch, randomly selected 1/8th of the images are replaced by the latent samples from actor model, reconstructed with the vae decoder network. New samples were reconstructed from a randomly initialized latent vector given a label vector. This classifier is evaluated against the performance of a base classifier without using any data augmentation techniques. The classifier is trained based on the loss function mentioned in Eqn.4.

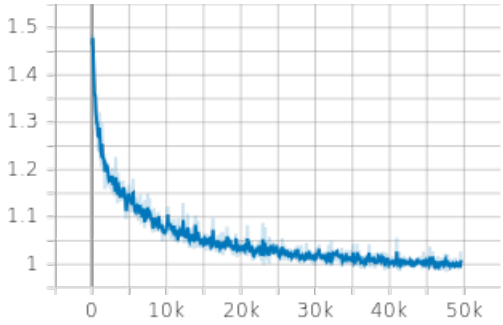
Testing We used three different datasets - DDU's Dirty-MNIST [7] and CIFAR-10-C[8] to test our models to evaluate how well it performs on new data. Once the classifier is trained, the model is tested on the above datasets. CIFAR-10-C is the dataset that has images with multiple noises applied to the base CIFAR10 dataset. The model is tested on all the different types of noise and the average test accuracy is computed.

6 Results and Discussion

VAE was trained on both datasets and the training curve for the same are shown below in Fig.4a and Fig.4b. Fig.6 and Fig.5 represent the generated samples from the decoder of the VAE.



(a) Reconstruction Loss plot for CIFAR10 dataset.



(b) Reconstruction Loss plot for MNIST dataset.

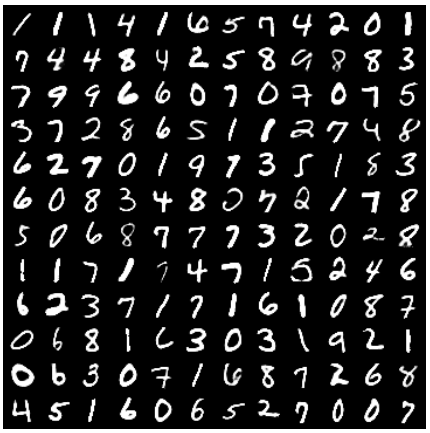


Figure 5: MNIST: Reconstructed image by VAE.

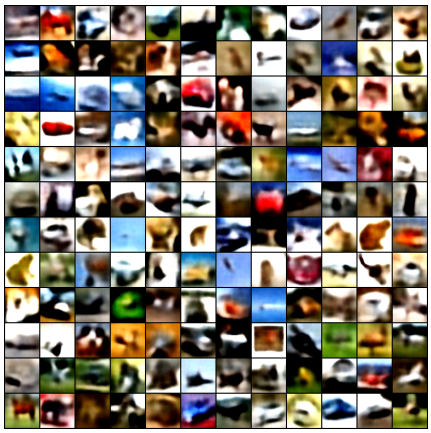


Figure 6: CIFAR: Reconstructed image by VAE

We then train the actor-critic model based on the frozen encoder and decoder models from the VAE. The trained actor-critic model was able to generate samples that closely resemble the data without

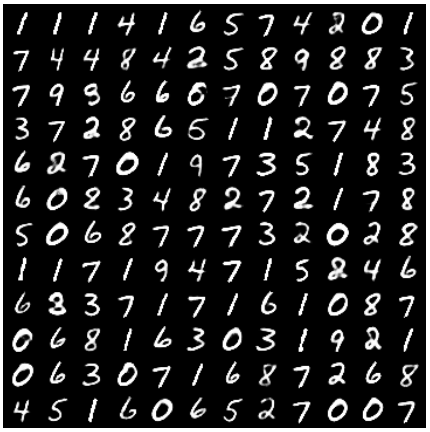


Figure 7: MNIST: Reconstructed image using Actor model

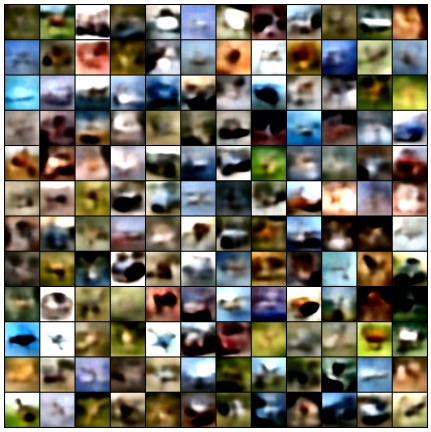


Figure 8: CIFAR10: Reconstructed image using Actor model

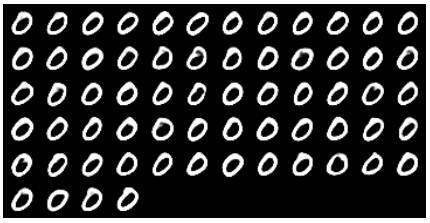


Figure 9: MNIST: Reconstructed image using Actor model for label 0

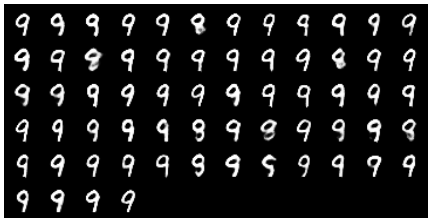


Figure 10: MNIST: Reconstructed image using Actor model for label 9

needing actual data representation. We also present a few samples from a set of classes of actor-critic model on both the datasets. We utilize the latents from the encoder and decoder to generate samples that are not actually present in the dataset. The model trained on MNIST was able to generate clearer and better samples compared to the model trained on CIFAR due to the better representation of latents trained by VAE on MNIST compared to CIFAR dataset. The resolution of CIFAR also caused poor representation of the latent as it has only 32x32 pixels. The resized larger image introduced artifacts which resulted in poor reconstruction by the VAE. Inferior latent representation by the VAE trained on the CIFAR resulted in a below average actor and critic model as they heavily depend on the latent representation by the pretrained VAE.

The below two plots Fig.13a and Fig.13b shows the performance of the classifier with and without the latent constraint method. It's evident from the graphs that both the models result in almost the same accuracy. This proves the objective that even with less data, more labelled images generated from the actor-critic model results in similar/ better performance, reducing the dependency on huge volume of data for training the classifier.

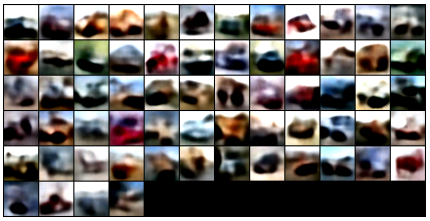


Figure 11: CIFAR10: Reconstructed image using Actor model for the label Truck.

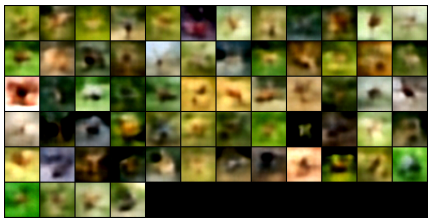
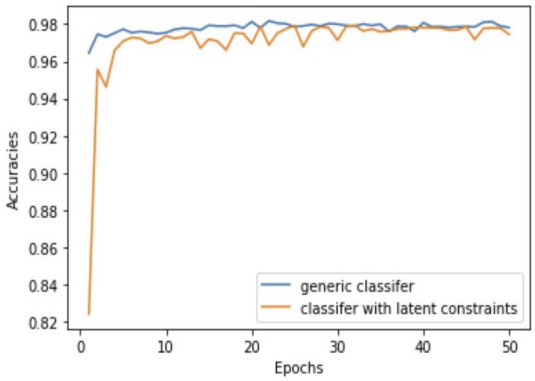
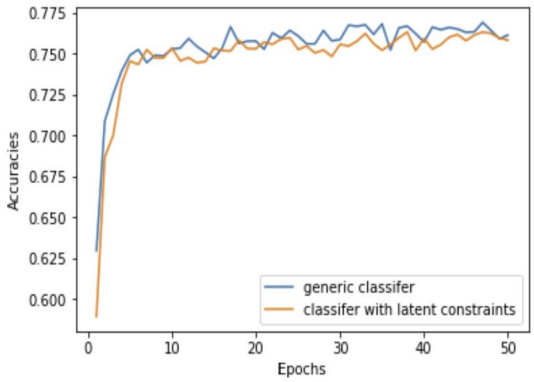


Figure 12: CIFAR10: Reconstructed image using Actor model for the label Deer.



(a) Accuracy plot for MNIST dataset.



(b) Accuracy plot for CIFAR10 dataset.

We finally test the trained classifier on the out-of-distribution data for each of the dataset i.e. Dirty-MNIST[7] for MNIST and CIFAR-C[8] for CIFAR-10. The below Table 1 compares the accuracy of each of the models on the test dataset. The results show that even the classifier trained using generated samples doesn't improve the performance of the general classifier, it significantly enhances the performance when tested on out-of-distribution dataset. However, with CIFAR10 there is not much improvement in the test accuracies on out-of-distribution dataset due to poor latent space by the pretrained VAE.

Accuracy Results on Out-Of-Distribution Data				
Model	MNIST-Test	Dirty-MNIST Test	CIFAR-Test	CIFAR-C Test
Classifier	95%	41%	74%	65%
Classifier Trained Using Actor-Critic VAE	94%	**57**%	75%	**66**%

Table 1: Comparison of Classifier Accuracies on the Test Dataset.

7 Conclusion

We implemented the paper[6] on two different datasets i.e. CIFAR10[1] and MNIST[7] and were able to successfully generate samples from constraining the latent using actor and critic models. We generated samples from random latent vectors by constraining the latent space using class labels. This approach is useful as it doesn't deal with the high dimensional data, instead uses the latent space by the pretrained VAE and therefore is quite stable compared to CGAN[9]. The generated samples help generalize the classifier better leading to better accuracies when tested on out-of-distribution dataset. We find out that the performance of the actor and critic model is largely dependant on the latent representation by the pretrained VAE which affects the performance of classifier.

References

[1] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[2] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[5] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks, 2016.

[6] Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models, 2017.

[7] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021.

[8] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[10] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space, 2017.

[11] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. 2017.

[12] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models, 2017.

[14] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks, 2017.

[15] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks, 2017.

[16] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

8 Appendix

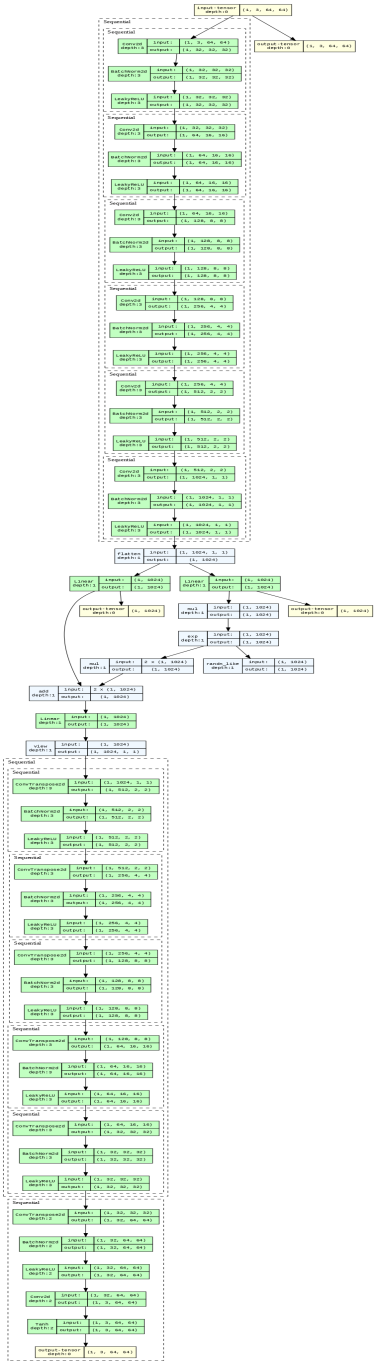


Figure 14: VAE architecture

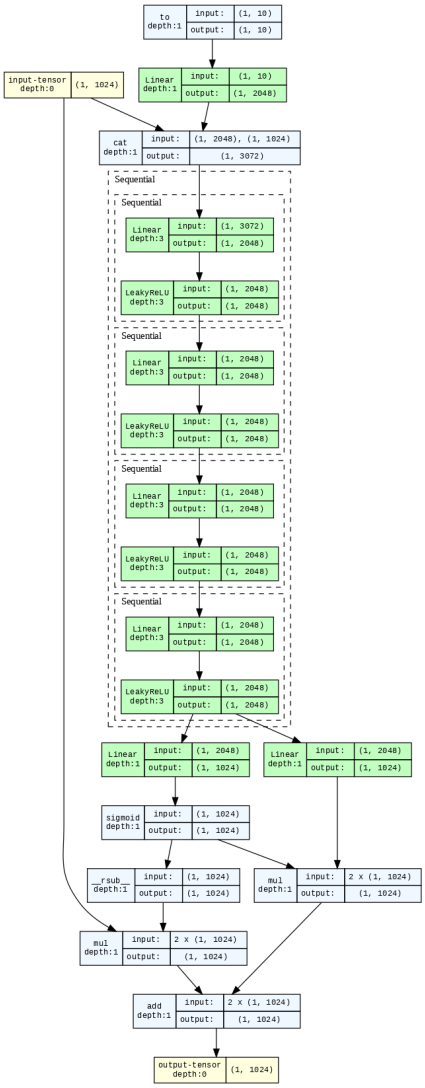


Figure 15: Actor network

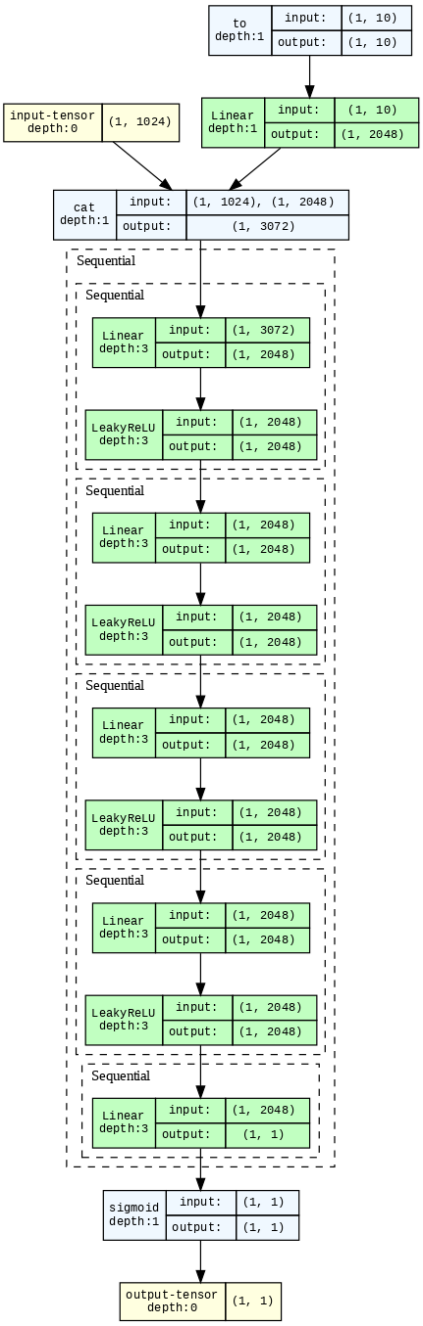


Figure 16: Critic network

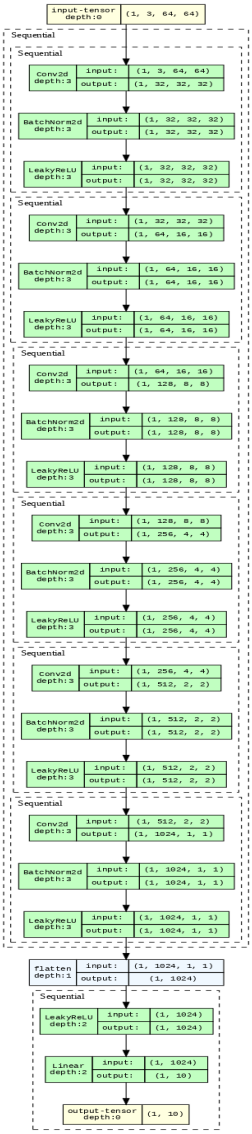


Figure 17: Classifier architecture