Conditional VAE and GAN: Effectiveness and Redundancy

MATH7019: Probabilistic Graphical Models

Project

Western Sydney University

Napat Sahapat
20619406

June 19, 2023

1 Introduction

Generative models allow reconstruction of new images based on a smaller dimension representation. However, without explicitly training the model with the labels included, such models can only generate random images that look like images in the original dataset. To provide more control over which class of images are generated, conditional generative models allows the user to instruct the model to generate a particular class of images. Not only is this functionality useful, it has also been shown to improve the quality of the generated models [Denton et al., 2015].

In this report, we will be comparing the performance of two conditional generative models: Conditional Generative Adversarial Networks (C-GAN) and Conditional Variational Auto-Encoder (C-VAE). The dataset used will be the MNIST digits, and the condition will be the label of the digits themselves. Evaluation and comparison will be made on the basis of how realistic the generated images of each model appear, as well as using quantitative measures: Inception Score and Frechet Inception Distance (FID). Additionally, an analysis on the latent space representation of VAE and CVAE models will be discussed.

2 Model Implementation

2.1 C-GAN

Generative Adversarial Networks (GAN) models employ two competing models, the Generator and the Discriminator, to generate new data that is indistinguishable from original data [Goodfellow et al., 2014]. The Generator performs the role of the image generator, attempting to create a realistic image (taking in a random latent vector, and returning an image). The Discriminator, meanwhile, tries to differentiate between artificially-generated and original images (taking in an image and returning a value between $[0, 1]$. Both models are trained simultaneously.

During training, the generator tries to bring its artificial images as close to 1 as possible, while the discriminator attempts to score the generator's images as 0 and real images as 1. Thus, the cost function for the generator ($G(\cdot)$) can be written as:

$$J_{G(\cdot)} = \text{BCE}(D(G(\tilde{x})), 1) \tag{1}$$

And the discriminator ($D(\cdot)$) cost is given by:

$$J_{D(\cdot)} = \frac{\text{BCE}(D(G(\tilde{x})), 0) + \text{BCE}(D(x_{real}), 1 - \text{alpha})}{2} \tag{2}$$

where x_{real} are the original images, and $\tilde{x}$ are the latent vector given to the generator. The BCE function is the Binary Cross Entropy Loss function, and is given by:

$$\text{BCE}(a, b) = -\mathbb{E}_{\text{data}} \left(b \cdot \log a + (1 - b) \cdot \log (1 - a) \right) \tag{3}$$

One-sided Label Smoothing is a technique which extends GAN by having the Discriminator evaluates real images strictly [Salimans et al., 2016]. A hyper-parameter alpha is imposed on the real images BCE in Equation 2, so instead of comparing discriminator output of real images to 1, it is compared to $1 - \text{alpha}$. Label smoothing have been claimed to improve GAN results by Szegedy et al., 2016, and we will be comparing three values of alpha: 0.2, 0.1 and 0 (no smoothing).

Conditional GAN Adding condition to regular GAN by means of image label has been suggested to improve generative ability [Goodfellow et al., 2014]. To achieve this, a one-hot encoded vector representing the label of the image is appended to two positions within the GAN. First, it is appended to the latent vector input to the Generator. Second, it is appended to the image input to the discriminator.

Network Architecture To keep training simple, only linear hidden layers are implemented in the neural network structures of the Generator and the Discriminator. The Generator has one hidden layer with a size of 256, using a `ReLU` activation, and a `tanh` output activation. The Discriminator also has a single hidden layer of 256 nodes with `ReLU` activation, with a `sigmoid` function for output activation (to constrain output as probabilities). The latent vector size is 64.

2.2 C-VAE

The Variational Auto-Encoder (VAE) is a type of generative models which combine variational Bayesian probabilities with regular auto-encoder [Kingma and Welling, 2022]. As such, instead of representing data as a 1D latent vector, the encoder in a VAE stores the latent representation as a parametric distribution, which can be sampled later by the decoder. Since commonly Gaussian distributions are used for representation, VAEs learn to represent a latent vector with a mean and standard deviation. Hence the encoder will output a μ and $\log \sigma^2$ for each dimension of the latent variable.

In order to have the stochastic sampling part of the process tractable, a commonly used method is the *reparameterisation* trick, which represents a sample $\mathbf{z} = \mu + \epsilon \cdot \sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$ is the stochastic part that will not be traced in the gradient tree of calculations.

The loss function for training the C-VAE is given by:

$$J = \text{BCE}(x_{real}, x_{generated}) - \frac{1}{2} \cdot \sum \left(1 + \log \sigma^2 - \mu^2 - \sigma^2\right) \tag{4}$$

[Kingma and Welling, 2022]

where the left term describes the BCE loss between real and generated images, and the right term describes the KL Divergence.

Conditional VAE Similarly to C-GAN, applying condition to regular VAE curtails appending a one-hot encoded vector of the associated label to both the encoder and decoder. In the encoder input, this refers to appending the one-hot vector to the flattened image vector itself. For the decoder, this means first sampling the latent representation from the latent μ and σ then appending the one-hot vector.

Network Architecture Similarly to C-GAN, only fully-connected, single layers are used in each of the encoder and decoder of C-VAE. The hidden layer sizes are also both 256 nodes. Decoder output activation is `sigmoid` to restrict image output ranges to between $[0, 1]$. Hidden layer activation are all `ReLU`, except the output of the encoder which output separately μ and $\log \sigma^2$ in parallel linear layers. The latent vector size is 64.

2.3 Evaluation Scores

Aside from examning the quality of the generated images by each model, quantitative measures are also used for comparing different model capabilities. These measures, starting with the Inception scores, are originally proposed for the Inception Network classifier [Salimans et al., 2016], which use the classifier to compare the quality of real versus generated images. However, the concept itself can be applied to any set of images, as long as the corresponding classifier is trained properly for those images.

In order to train our supporting classifier, we implemented two convolution layers followed by a fully-connected layer with 256 nodes. Each convolution layer is followed by a batch normalisation step, a `ReLU` activation and a pooling layer (max pooling for first convolution layer, average pooling for second). The classifier is trained on the training set of the MNIST dataset, after only 10 epochs.

Inception Score Originally presented by Salimans et al., 2016, the Inception score utilise the softmax probabilities output of the classifier to calculate the KL Divergence between conditioinal label distribution $p(y|x)$ and the label distribution $p(y)$.

Frechet Inception Distance (FID) Built on top of the Inception Score, the Frechet Inception Distance (FID) compares the difference between real and generated images by using the next to last layer of the classifier [Heusel et al., 2017].

Since Inception Score measures quality, and FID measures distance from the real images, the former would increase with higher quality while the latter would decrease.

2.4 Dataset and Training Regime

The dataset used for our experiment is the classic MNIST digits from the `torchvision` package. The images are 28×28 in size, and there are 60,000 training images and 10,000 test images. Only the training set was used during the training of C-GAN (Section 2.1), C-VAE (Section 2.2 and the classifier (2.3). However, during the Score comparison in Section 3.2 the latent variable analysis in Section 4, the test set is used to generate the plots and evaluate the scores, since the lower volume allows for quicker processing.

The optimiser used is the Adam optimiser in the `pytorch` package, with a learning rate of 0.001. During training, a batch size of 64 is used and the training is run for 100 epochs. The model as well as generated images are saved at regular interval, in order to reuse the model if training was interrupted and to compare performance efficiency during training, respectively.

3 Comparison Results

3.1 Visual Examinations

From the generated images of C-GAN and C-VAE in Figure 1 and Figure 2, respectively, one can notice quite a pronounced difference in quality between the two models. C-VAE generates a much more realistic-looking and sharper quality image compared to C-GAN. Moreover, even after 20 epochs, C-VAE images are much better looking than C-GAN at 100 epochs.

It is important to note, however, that, while the images generated by C-GAN are based on random latent vectors passed onto the generator (with the label appended), the ones created by C-VAE used the actual latent variable returned by the encoder to create a reconstructed image. This was an oversight in the evaluation process that has not been addressed in this project. Thus, the results from the two models may not be directly comparable, since the generation process is different.

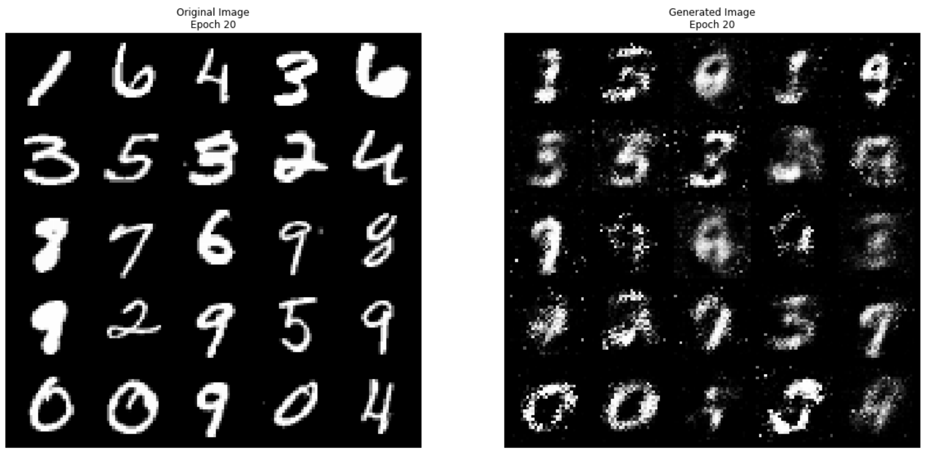
3.2 Quantitative Scores

While the image generation visual inspection procedure may be useful to obtain qualitative insight at a glance, a much more useful evaluation process is to compare quantitative scores for each model. Moreover, variation within the training of each models can also be compared.

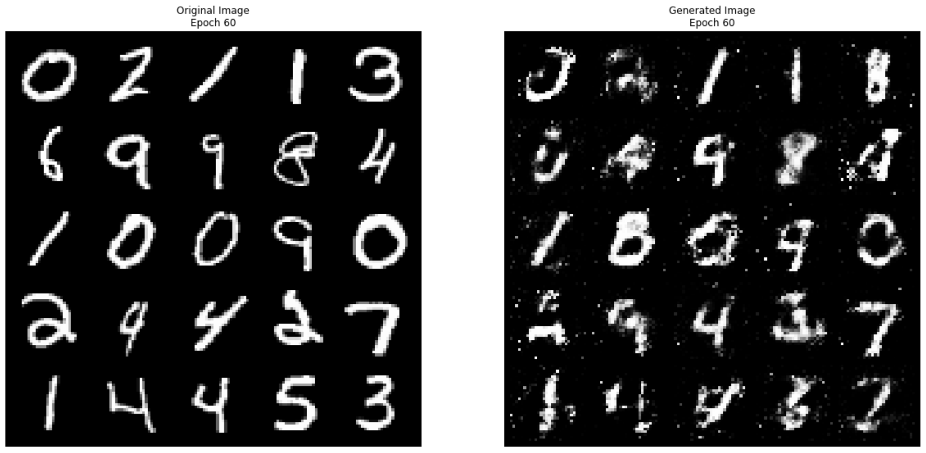
Using the classifier trained in Section 2.3 to evaluate the generated images from each model, the boxplots in Figure 3 and 4 show the Inception Scores and Frechet Inception Distance, respectively. The scores are calculated per batches of 500 images from the MNIST test dataset.

Both the Inception Scores and FID plots show a similar comparative outcome, albeit inversely. C-GAN models generate far lesser-quality images than C-VAE. Interestingly, for C-GAN, using a higher alpha value for one-sided smoothing can be seen to increase Inception Scores for the generated images (i.e. better quality) (Figure 3).

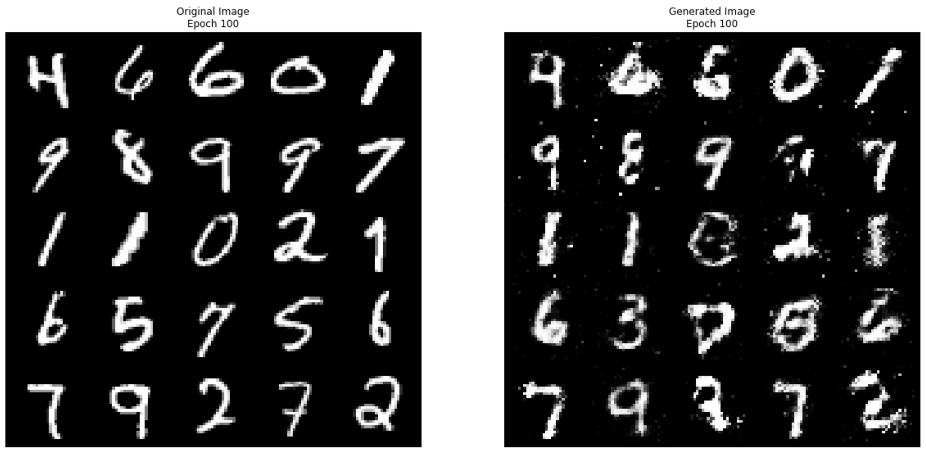
Another interesting observation from the score comparison is that C-VAE and VAE have surprisingly similar quality images based on their scores (Figure 3 and 4). Although it can be argued that C-VAE produce slightly better scores than regular VAE, the difference does not seem to be great, especially considering the distance to real images.



(a) Epoch 20

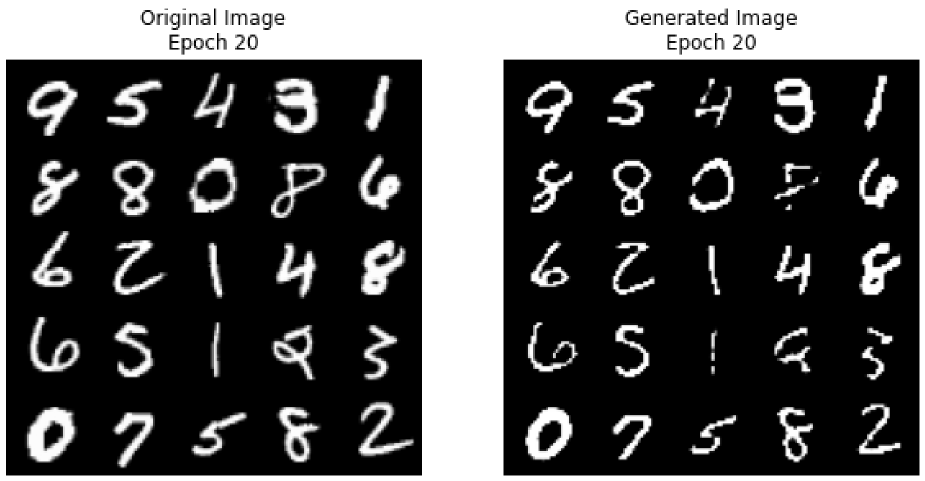


(b) Epoch 60

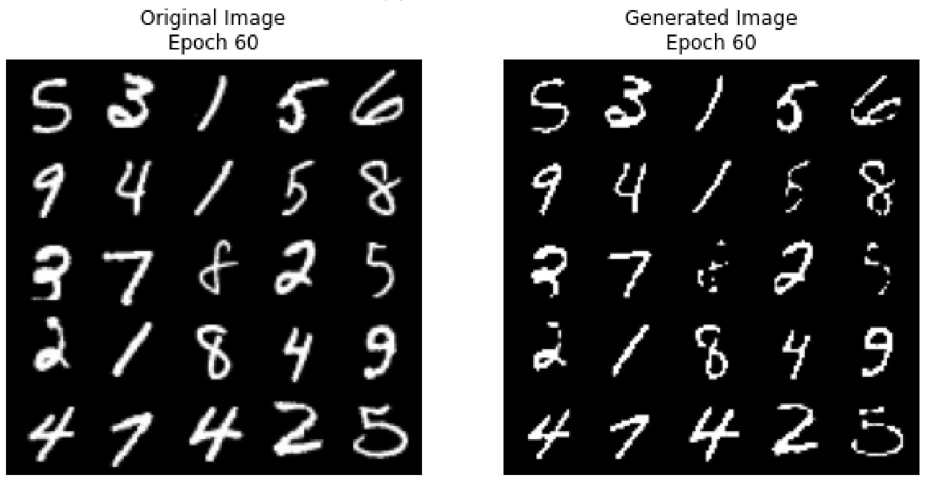


(c) Epoch 100

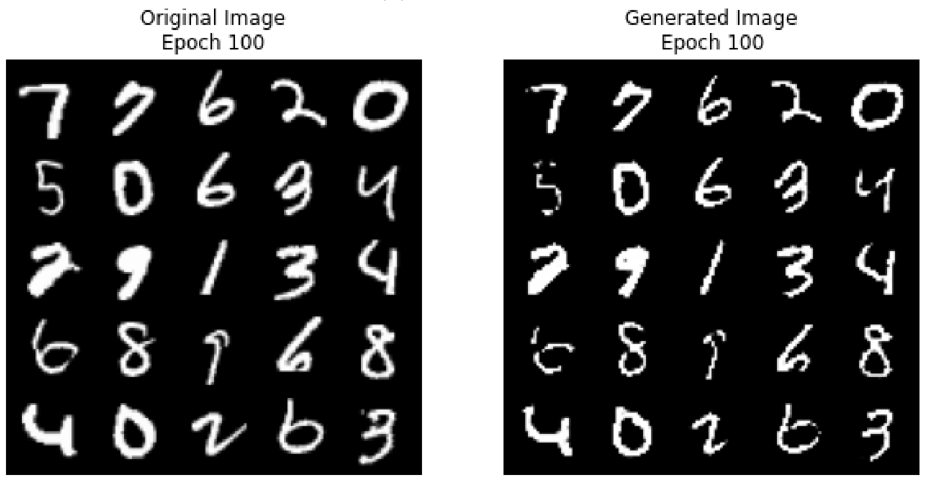
Figure 1: C-GAN Generated Images.
Latent Dimension = 64. Alpha = 0.2



(a) Epoch 20



(b) Epoch 60



(c) Epoch 100

Figure 2: C-VAE Generated Images.
Latent Dimension = 64

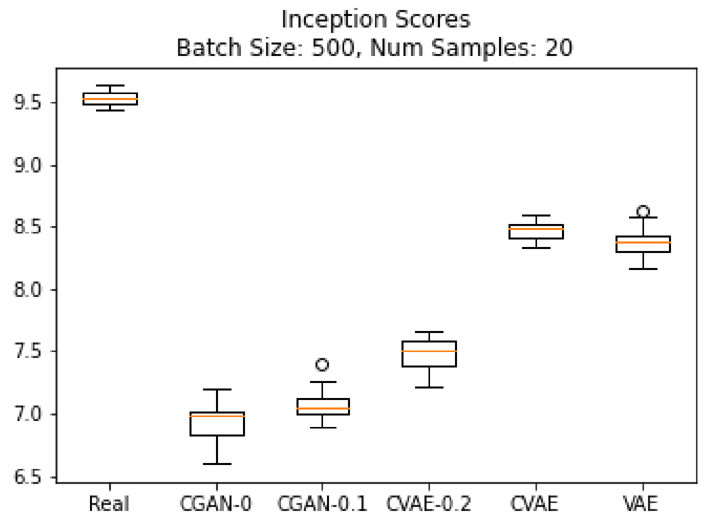


Figure 3: Inception Scores Model Comparison
CGAN-0, -0.1, -0.2 describe the alpha value of one-sided label smoothing adjusted version of GAN.

4 Latent Variable Analysis

In addition to the evaluating how well the generated images resemble real images, another analysis could be made on the latent representation of the images which is generated by the encoder, and is used by the decoder. After all, one of the benefits of auto-encoders is the ability to represent data in smaller dimensions than the original data. Variational Auto-Encoders (VAE), in particular, represent the data in an even more compact way than simple Auto-Encoders: by simply storing the *mean* and *variance* of each dimension, instead of the entire vector.

One question one may have is whether there is any insight to be gained regarding the structure of this latent representation, particularly to the different classes of data (digits, in this case). To answer this question, a latent representation of only 2 dimension is used to train a VAE, so the means of the 2D representation can be visualised on a scatterplot. Note that this latent vector analysis only makes sense for VAE-based generative models only; GAN does not have a middle represetation layer that can provide any insight – the latent vector is passed in to the generator as the initial input, and is not generated by any encoder.

The 2D latent representation is compared between regular VAE and condition VAE (CVAE). The data point is colour-coded by the digit class of the images, to show any pattern (Figure 5).

An interesting pattern can be seen in the latent variable for regular VAE (Figure 5b). Each digit class appears to be grouped surprisingly into distinct regions. Although the border between regions is not clear, and there are many interspersed data points, the overall structure is very striking.

In contrast, the latent representation means created by the CVAE encoders does not show a clear delineation between classes (Figure 5a). A possible explanation is that these *means* which, when combined with a corresponding *variance* to generate a 2D latent variable, are then concatenated with the actual class representation by a one-hot vector (a length 10 vector in this case). As such, the decoder of a CVAE learns to generate a particular class based predominantly on the one-hot vector, while the 2D latent vector plays less of a role in this case. It would be interesting to see if this delienation

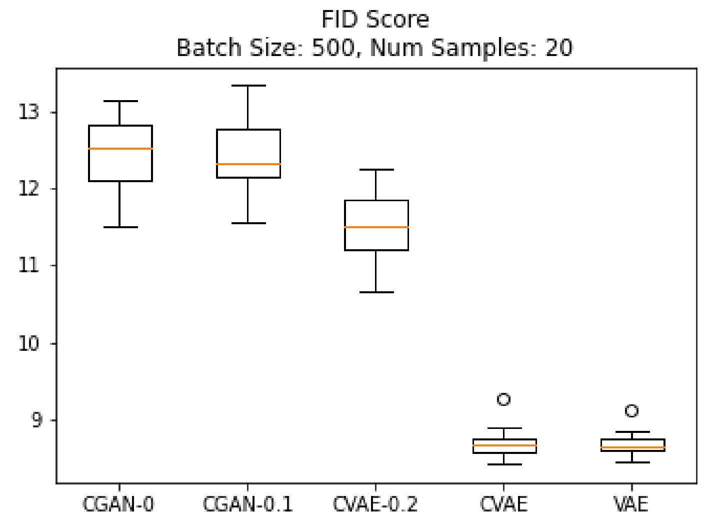


Figure 4: Frechet Inception Distance (FID) Model Comparison
CGAN-0, -0.1, -0.2 describe the alpha value of one-sided label smoothing adjusted version of GAN.

improve with higher latent dimension; however, it would be impossible to visualise dimensions higher than three.

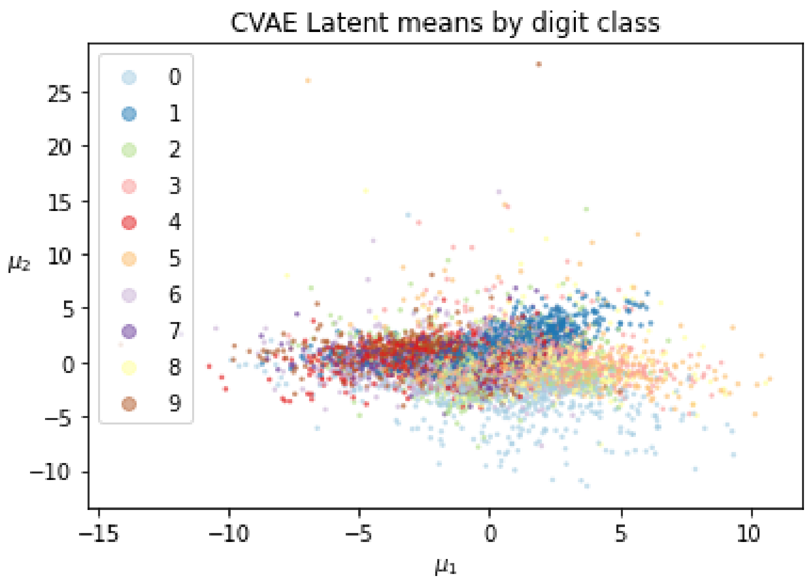
An interesting implication of this result is that, since for CVAE most of the information regarding the class resides in the one-hot vector instead of the latent variable (which depends on a starting image), it may be possible to control the generation of any desired class of digits, using only random latent variable and a one-hot vector of the desired class. In other words, CVAE with lower latent dimension provides for better control of class generation than normal VAE, by supplying the desired class as a condition.

5 Conclusion

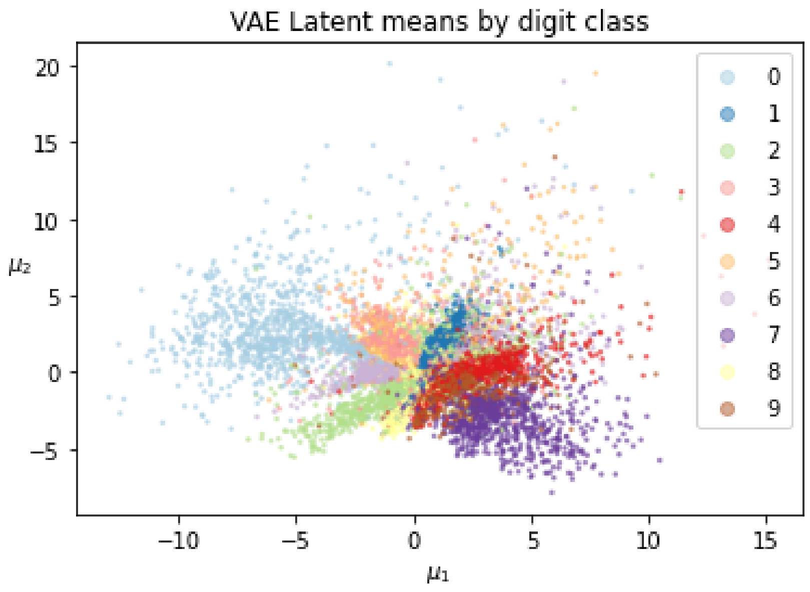
From our analysis of the different variations of the C-VAE and C-GAN generative models, we do not have a significantly strong evidence to suggest that adding labels as conditions to the models improve generative quality. However, C-VAE and VAE appears to outperforms C-GAN by a large margin, given similar training structure and network parameters. The only caveat with the comparison is that the images generated by C-GAN rely on random latent vectors, while those of C-VAE use the encoder-generated latent values directly. For a better comparison the latent vector in C-VAE should also be randomly generated.

Techniques to improve GANs like one-sided label smoothing is also found to improve C-GAN generative capability, with higher alpha values resulting in better quality.

Inception scores and Frechet Inception Distance also provide convenient and objective tools to compare models, rather than relying on subjective judgment. However, to use those quantitative scores, a classifier has to first be built with at least one fully-connected layer before the output.



(a) CVAE



(b) VAE

Figure 5: 2D Latent Representation by Class for CVAE (a) and VAE (b)

Advantages and Disadvantages of Conditional Models

Based on our analysis, using conditional generative models can have the added benefit of allowing the user to supply the condition of which class of images to generate. If combined with natural language processing techniques, such as word vectors, it should be possible to create more fine-tuned, nuanced control over the generation of the desired images.

One apparent disadvantage that we've found in this study, however, is that adding in the label condition may degrade the value of the latent representation vector to encode the image class. Since most of the information about the class now live in the appended label vector, the latent vector lose importance in controlling the class of images to generate. In practice, however, it might be more convenient to append the desired label to the latent vector, and so not being able to use the latent vector to represent each class is not an inconvenience, but simply a very interesting side-effect of the conditional method.

Future Improvements

One main issue of our analysis is that it is not rigorous enough to compare the effects of different attributes in our models. For instance, C-GAN with GAN and C-VAE with VAE, at varying model capacity (e.g. latent vector dimension) may provide useful insight into how adding label conditions affect model performance, as well as robustness. The robustness part may be tested by seeing whether or not applying conditional models with lower dimension decrease quality equally as doing the same for regular models. In other words, if C-VAE with 2 dimension perform closer to C-VAE with 64 dimension than VAE with 2 dimension perform with VAE of 64 dimension, we can be certain that the conditional model is more robust to reducing latent dimension.

As mentioned earlier, using comparable generation methods, i.e. by using random latent variable to generate new images, will be crucial to effective scientific comparison between models.

Finally, analysing the importance of latent variable in relation to appended label condition when controlling image generation can be dine by models with higher latent dimension. Techniques for clustering such as K-Nearest Neighbour, may be performed to cluster class labels for latent variables with higher dimensions.

References

Denton, E. L., Chintala, S., Szlam, A., & Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, *abs/1506.05751*. http://arxiv.org/abs/1506.05751

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, *abs/1706.08500*. http://arxiv.org/abs/1706.08500

Kingma, D. P., & Welling, M. (2022). Auto-encoding variational bayes.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. *CoRR*, *abs/1606.03498*. http://arxiv.org/abs/1606.03498

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.