Sparse VAE

Comparison of VQ-VAE to a VAE with a Task Driven Sparse Embedding

Joshua Athayde[1]

[1] 3rd Year Undergraduate, University of Chicago

December 9, 2022

---

1. Introduction

Sparse coding has been an alternative to Vector Quantization for both featurization and classification. As a result, I thought it would only be natural to extend sparse coding to the context of generating samples from an unknown but trainable distribution.

With the original VAE encoding images in some given continuous distribution, it suffered from blurry images. In contrast, VQ-VAE uses vector quantization for the latent distribution, which seems to resolve this issue. Given that sparse coding also maps our neural network encoding onto a trainable dictionary, I hypothesize that the Sparse VAE will also resolve the fuzziness issue.

In addition, I hypothesize that the Sparse VAE will train to a sufficient level using fewer epochs because each sample trains the entire dictionary rather than just a single embedded vector. To test this hypothesis, I will train both a VQ-VAE and my own implementation of a Sparse VAE on the CIFAR10 dataset and compare results.

2. Related Work

I discovered after starting my project that my idea is somewhat closely related to this repo on GitHub[1]. The main difference in my implementation is that I passed gradients from the decoder straight through to the encoder, which seemed to drastically improve results. Without doing so, my attempts to replicate this tended to generate noise, and coefficients would grow exponentially eventually leading to exceptions.

Additionally, I also used the task driven sparse embedding algorithm listed in this paper[2], but replaced the the gradient of the weight matrix with the gradient of the decoder.

Lastly, I also used the Iterative Shrinking Threshold Algorithm from this GitHub repo[3] (where I copied it because the code to make it a package is currently sitting in an open pull request)

3. Method

For code, see: https://github.com/jathayde314/pytorch-vqvae. For the implementation of VQ-VAE, see: https://github.com/ritheshkumar95/pytorch-vqvae.

The implementation of the Sparse-VAE works as follows. The encoder is a 2d Convolution, BatchNorm layer, ReLU, 2d Convolution, then two ResBlocks. Each ResBlock is a ReLU layer, then 2d Convoution, BatchNorm, ReLU, 2d Convolution, then BatchNorm.

The sparse dictionary finds a sparse encoding using an Iterative Shrinking Threshold Algorithm.

The decoder is the Encoder in reverse with transposed convolutions and a final Tanh activation function.

The loss function is as follows:

$$L = \log p(x|z_q(x)) + \lambda \log p(z_q(x)) + \beta \|z_e(x) - sg[e]\|_2^2$$

where $p(x|z_q(x)) \sim N(0,1)$ and $p(z_q(x)) \sim N(0,1)$. The $\log p(z_q(x))$ comes from the re-parameterization trick, and $E \log p(z_q(x)|x) = 0$ because $z_q(x)$ is deterministic.

Lastly we set:

$$\nabla D = -D\beta^* z_q(x)^T + (x - Dz_q(x))\beta^{*T}$$

Where

$$\beta^* = (D_\Lambda^T D_\Lambda + I)^{-1} \nabla_{z_q(x)_\Lambda} L(x, z_d(x))$$

and Λ is the non-zero coordinates of $z_q(x)$. For more info on this, refer to Mairal.

4. Data

Comparing the two models, we have both reconstruction ($\log p(x|z_q(x))$) and quantization ($\|z_e(x) - sg[e]\|_2^2$) losses.

For note, the blue lines are the Sparse-VAE, and the orange lines are the VQ-VAE.

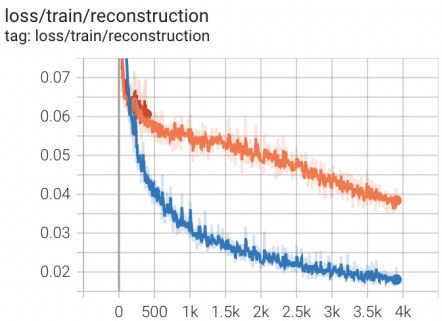

Figure 1. Train Reconstruction Loss

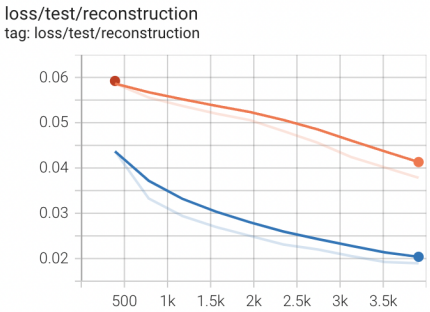

Figure 2. Test Reconstruction Loss

As a result, we can see that the Sparse-VAE produced more similar outputs in both training and testing.

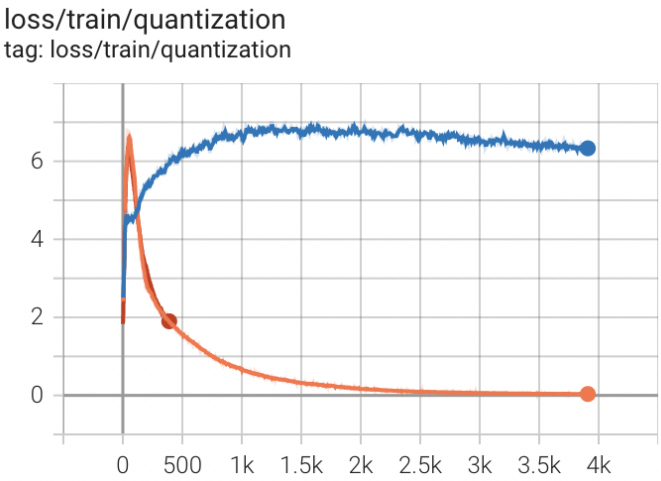

Figure 3. Train Quantization Loss

The high quantization loss for the Sparse VAE implies that the dictionary did affect the final result of the network (because zero loss would have $z_e(x) = z_q(x)$). However, this would imply that we got sparse representaions, but this does not seem to be the case:

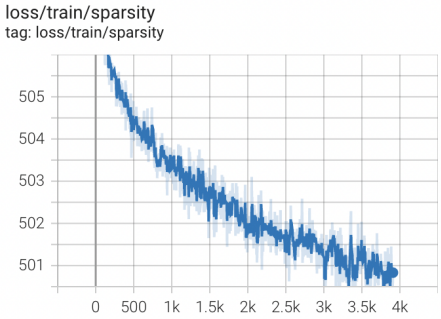

Figure 4. Train Sparsity

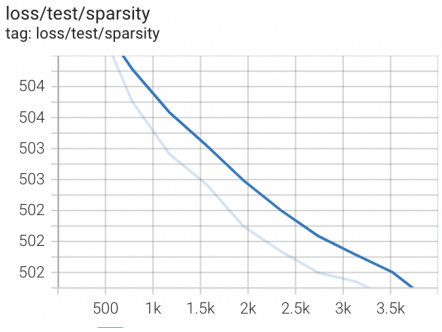

Figure 5. Test Sparsity

For the experiments run, the embedding had 512 vectors, so an average sparsity above 500 is very high.

5. Image Comparison

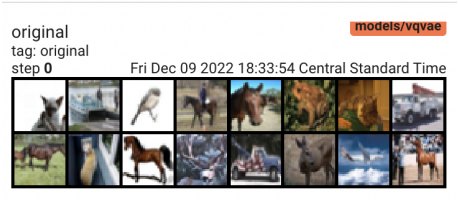

Figure 6. Original VQ VAE

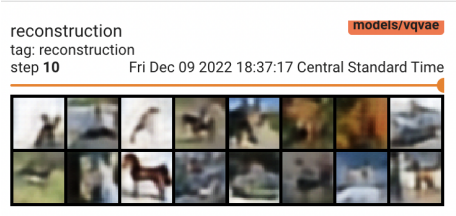

Figure 7. Generated VQ VAE

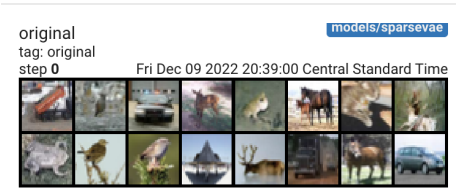

Figure 8. Original Sparse VAE

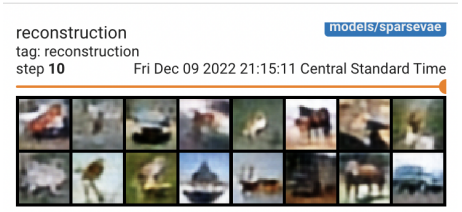

Figure 9. Generated Sparse VAE

6. Conclusion

From the data, we conclude that the Sparse-VAE did train faster than the VQ-VAE. However, given the size of the images in CIFAR10, it was hard to tell if the sparse VAE had the same blurriness issue as a normal VAE would. The Sparse VAE generated images do seem to be slightly less blurry than those of the VQ VAE (although this may be personal bias), which does provide some evidence that the sparse coding does improve results. On the other hand, given that the quantization loss for the Sparse VAE stays high and the lack of sparse representations, it is possible that the Sparse VAE acts no differently than a normal VAE because it may ignore the quantization loss objective.

Despite these improvements, the complexity of the Sparse VAE added ten times as much run-time. Because the bottleneck with training auto-encoders is the compute rather than images (given there are billions on the internet), this additional complexity is likely not worth the tradeoff of better results.

References

[1] Ginger, Yiftach, Or Perel, Roee Litman SparseVQVAE: Sparse Dictionary based Vector Quantized Variational AutoEncoder https://github.com/amzn/sparse-vqvae

[2] Mairal, Julien, et al. Task-Driven Dictionary Learning. Sept. 2010. arxiv.org, https://doi.org/10.1109/TPAMI.2011.156.

[3] Feinman, Reuben PyTorch Lasso, https://github.com/rfeinman/pytorch-lasso