Generate Histopathologic Images Using Variational Autoencoders

Lynn Zheng
ruolinzheng@uchicago.edu

December 4, 2020

1 Project Objective

Class imbalance is a serious issue in histopathologic datasets. Even for curated datasets like the Kaggle Cancer Image Dataset, there are 55% normal images and 45% tumor images. This data imbalance issue motivates my two-part project objective. First, to train generative models for the purpose of augmenting histopathologic image datasets. We need class-conditional models to generate normal and tumor images with corresponding class labels. Second, to develop a quantitative method to evaluate the performance of the generator, equivalently, the quality of generated images. This is important since we cannot simply look at generated histopathologic images and conclude that they "look realistic" like we do with the MNIST dataset or the CelebA-HQ (CelebrityAttributes-HighQuality) faces dataset.

Specifically, I will train two types of generative models: Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). Both types of model will be class-conditional and utilize deep convolutional network. I will then generate samples of normal and tumor images using the generator. For the quantitative evaluation procedure, I will train binary classifiers that distinguish between normal and tumor images on the generated data. The binary classifier will also use deep convolutional neural networks. After training the binary classifiers, I will test them on real test data and evaluate their performance. The idea is that, if the generative models are able to model the real normal and tumor image distribution, the generated data will be very similar to real data. Then, a classifier trained on generated data should perform well on real test data. The performance of the classifiers trained on data generated by different generators can thus be used as a surrogate for the performance of the generator. This will allow me to quantitatively compare generative models that have very different architectures, i.e., VAEs which minimizes the sum of a reconstuction loss and a KL-Divergence loss, and, GANs whose generator don't have an explicit objective function.

2 Background Research

I read specifically about papers that apply generative models to the domain of medical images, as medical images are very different from images of day-to-day objects. One paper highlights the application of GANs on histopathologic images for tasks like segementation, data synthesis based on segmentation, and data augmentation as translation settings. [1] Another paper mentions previous works in generating retinal images and chest X-ray images with GANs and achieving pathological image quality. [2]

I also read more broadly about recent advances in generative models like GANs and VAEs, as well as quantitative methods to compare the performance of generative models of drastically different architectures. A 2019 paper on VQ-VAE2 (Vector-Quantized VAE) [3] proposes to evaluate the performance of generative models by training classifiers on the data they generate and subsequently testing the classifiers on real testing data. The performance of the classifier then acts as a proxy for the performance of the generative model.

3 Approach

3.1 Dataset

I used the Kaggle Histopathologic Cancer Detection Dataset. [4] The dataset was curated for the task of identifying metastatic tissue in histopathologic scans of lymph node sections. There are a total of 220,000 images with binary labels (0 for negative/normal, and 1 for positive/tumor). Each image is 96 by 96 pixels, with RGB channels, in TIF formats. According to the dataset description on Kaggle, a positive label indicates that the center 32x32px region of a patch contains at least one pixel of tumor tissue. About 55%, or, 120,000 of the images are normal tissues, and 45% of the images are tumor tissues.

[1] Tschuchnig, M. E., Oostingh, G. J., & Gadermayr, M. (2020). Generative Adversarial Networks in Digital Pathology: A Survey on Trends and Future Potential. arXiv preprint arXiv:2004.14936.

[2] Raza, K., & Singh, N. K. A tour of unsupervised deep learning for medical image analysis. arXiv 2018. arXiv preprint arXiv:1812.07715.

[3] Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In Advances in Neural Information Processing Systems (pp. 14866-14876).

[4] https://www.kaggle.com/c/histopathologic-cancer-detection/data

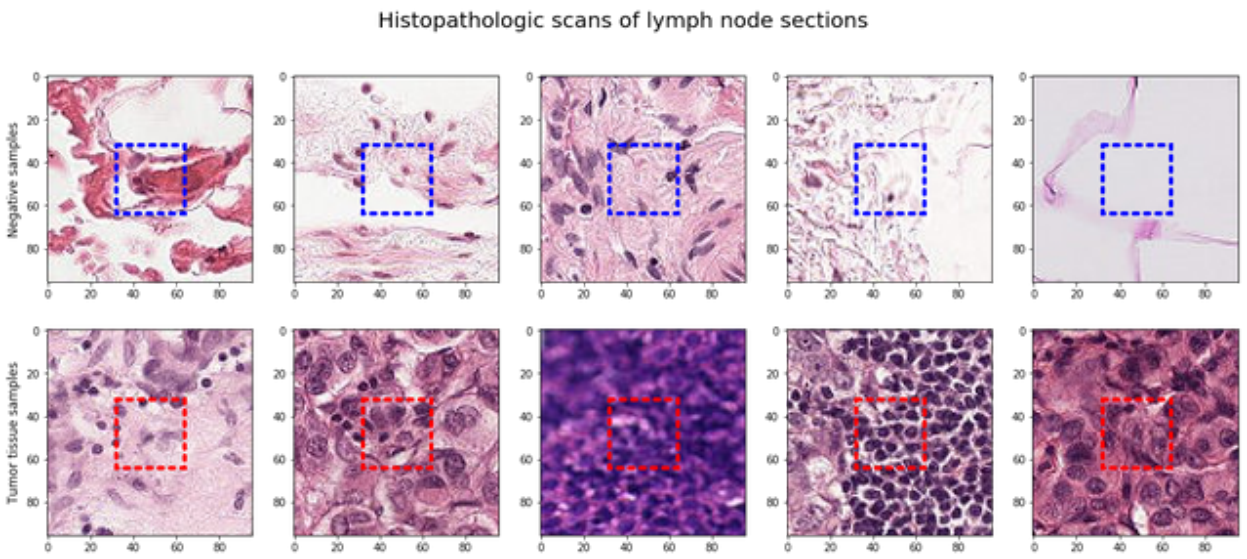

Figure 1: Sample images from the Kaggle dataset. The first row contains normal images and the second row tumor images. The center 32 by 32 patches are highlighted in boxes with dashed lines. Source: https://www.kaggle.com/qitvision/a-complete-ml-pipeline-fast-ai

3.2 Methodology

3.2.1 Data Preparation

Although the images are 96 by 96 pixels, only the pixels in the center 32 by 32 patch determine whether an image is positive or negative. To save data processing time and reduce unnecessary information, I cropped the center 64 by 64 region of each image in the dataset for use. This means that my generator will also generate 64 by 64 images. I split the dataset of 220,000 images into 160,000 for training and 6,000 for testing.

3.2.2 Training Generators and Generating Samples

I trained generators on all the 160,000 images. The model architecture and training hyperparameters will be discussed in the **Technologies** section below. I then generate 160,000 class-conditional samples from the trained generator, preserving the ratio of normal to tumor tissues as in the original 160,000 training set. I saved these 64 by 64 RGB images in TIF formats and their class labels in a CSV file, using the same format as the Kaggle dataset used.

3.2.3 Training Classifiers

I trained binary classifiers using the 160,000 samples generated by the generator. During training, I held out 25% of the dataset as a validation set. I then select the classifier with the

highest validation AUC score. I used AUC instead of accuracy as my model selection metric because the original dataset has a small class imbalance (55% normal and 45% tumor) and I sought to avoid the scenario where a trivial classifier learns to classify any image as normal to achieve an accuracy score seemingly superior to randomly-guessing.

3.2.4 Testing Classifiers

I tested the classifier on the real test set of 60,000 images. The idea is that, if the generator has learned to model the original data distribution and can generate data similar to the real training data, a classifier trained on the generated samples should also perform well on real testing data. For comparison, I trained a baseline classifier on the real training set of 160,000 images.

3.3 Revisions to My Original Proposal

The VQ-VAE paper implemented its model in a Python deep learning framework named Sonnet [5], which I have never heard of. Therefore, instead of using its model, I decided to implement my own VAE structure. Since the vector-quantized proportion of the model architecture is non-trivial to implement, I implemented the plain class-conditional convolutional VAE.

My original plan was to reuse existing open-source implementation of models I need as much as possible, especially for the baseline classifier trained on the real dataset on Kaggle. However, implementations of class-conditional convolutional VAEs and GANs I could find on GitHub and Kaggle are of varying quality: Some are implemented in Keras, some in Tensorflow, and some in PyTorch. Few had documentations. I also decided that it is best if I can draw a parallel between the architecture of the VAE and the GAN: The VAE encoder should have a similar neural architecture to the GAN discriminator, except that the encoder outputs a vector but the discriminator outputs a scalar. The VAE decoder should have a similar neural architecture to the GAN generator.

Out of the above consideration and for the ease of streamlining hyperparameter-tuning, I implemented my own convolutional class-conditional VAE and GAN from scratch in PyTorch, building on vanilla VAEs and GANs I implemented for another class.

3.4 Prototype

I trained the class-conditional VAE and GAN on MNIST and achieved visually-sound results. [6] I recorded the learning rate as well as other hyperparameters and architectural details for use on the histopathologic dataset.

[5] https://github.com/deepmind/sonnet/blob/master/sonnet/examples/vqvae_example.ipynb
[6] Notebook with prefix `PS4` on https://github.com/RuolinZheng08/ttic31230-deep-learning

4 Technologies Used

I used PyTorch as my framework and implemented models from scratch. As a general setting for all the models, I used a mini-batch of size 32 and the Adam optimizer with a learning rate of 0.0005. I trained each model, classifier or generator, for 100 epochs. The training time is about 6 hours per classifier and 2 to 6 hours per generator depending on the model size.

4.1 Classifier Architecture

I used the same classifier architecture for experiments both with real and generated data. The classifier consists of 4 convolutional layers, with ReLU as the activation function.

4.2 VAE Architecture

Both the encoder and the decoder of my VAE consist of 4 convolutional layer connected by batch normalization layers. The encoder uses the leaky ReLU activation function and the decoder uses the ReLU activate function. I tuned two hyperparameters: the number of latent dimensions, ranging from 100, 500, to 2000, and the number of convolutional channels, ranging from 16 to 64.

To implement a class-conditional model, I added an one-hot encoding of the class label as image input channels in addition to the three RGB input channels.

4.3 GAN Architecture

The architecture of my GAN discriminator corresponds to that of my VAE encoder, and the architecture of my GAN generator corresponds to that of my VAE decoder. I experimented with two different loss functions: the binary cross-entropy loss and the Wasserstein loss. [7]

To implement a class-conditional model, I pass in the class label, apply an embedding to the class label, and concatenate the vector with random noise as the input to my generator.

5 Results

I expected that VAEs will generate blurry images as they are designed to learn the average of the data distribution. Therefore, I had hopes for the GANs to generate sharper images.

5.1 VAE

As the number of latent dimension increases, the images generated by VAEs become less blurry. However, they could not reach pixel-level precision so as to be useful as histopathologic images for training classifiers.

[7] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.

Figure 2: Top two rows: negative/normal images; bottom two rows: positive/tumor images

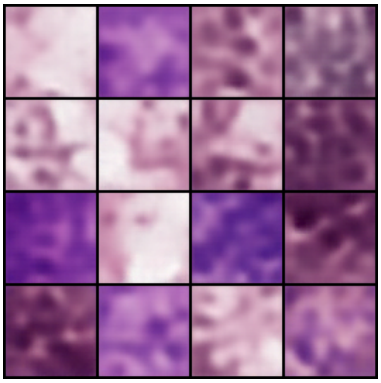

Figure 3: Images generated by a 100-latent-dimension VAE after 100 epochs

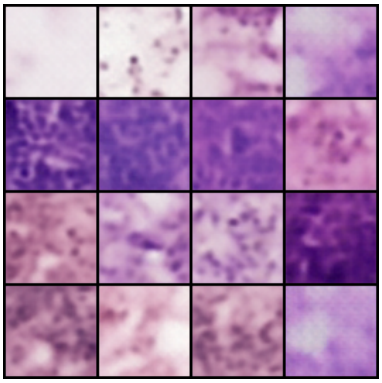

Figure 4: Images generated by a 500-latent-dimension VAE after 100 epochs

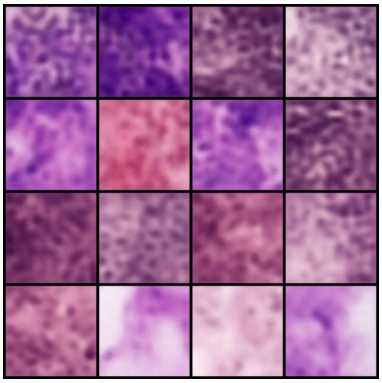

Figure 5: Images generated by a 2000-latent-dimension VAE after 70 epochs

5.2 GAN

Unfortunately, the GANs suffered from mode collapse despite my various attempts at using different loss functions, modifying the training schedule of the generator and the discriminator, and hyperparameter-tuning. The images are therefore completely unusable for training classifiers. The only upside is that at least mode collapse happened on different modes for the two different image classes, as the top two row images with negative class labels are visually very different from the bottom two row images with positive class labels.

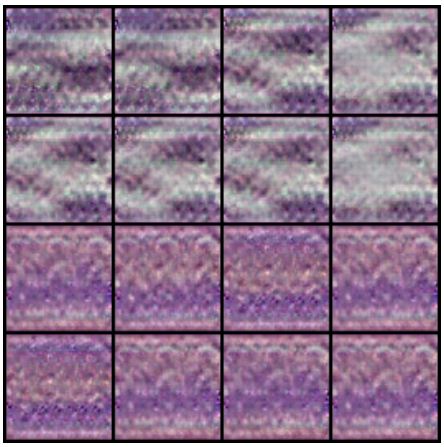

Figure 6: Images generated by a GAN trained using Wasserstein loss after 100 epochs

5.3 Classifiers

Although the images generated by the VAE fail to achieve pixel-level precision and are unlikely to be useful for training histopathologic image classifiers, I followed through my project planning for the purpose of studying the change in test loss, accuracy, and AUC for different model hyperparameter settings.

The baseline classifier trained on real data achieved a 0.92 test AUC without any fine-tuning. The classifiers trained on VAE-generated images, however, had AUC scores around 0.5, which means that they aren't very different from randomly guessing. The best-performing VAE, with 2000 latent dimensions and 16 convolutional channels, achieved a slightly better AUC score than randomly-guessing. One observation is that, as the number of latent dimension increases, test loss generally decreases and test AUC generally increases. An explanation for why the VAE with the most parameters, 2000 latent dimensions and 64 channels, didn't outperform the 2000-latent-dimension one with only 16 channels could be that the former wasn't trained to full convergence despite the 100 epochs.

Model	Latent dimensions	Channels	Test loss	Test accuracy	Test AUC
Baseline	-	-	0.3576	0.8491	0.9225
VAE	100	16	40.67	0.5531	0.4328
	500	16	26.81	0.5496	0.4814
	2000	16	**4.317**	0.5165	**0.5065**
	2000	64	9.875	0.5608	0.4940

Table 1: Classifer test accuracy and AUC

5.4 Confusion Matrices

I generated confusion matrices using a 0.5 threshold cutoff for the class probabilities generated by the classifier. Below is a comparison between the baseline classifier trained on real data and the one trained on VAE-generated data with the highest test AUC. It appears that the latter has erred significantly on false positives.

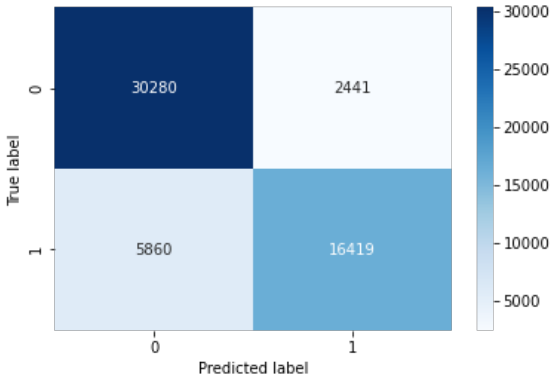

Figure 7: Confusion matrix for the baseline classifier trained on real data

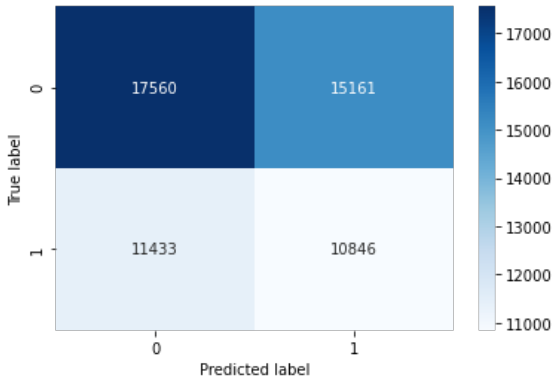

Figure 8: Confusion matrix for the classifier trained on data generated by the VAE with 2000 latent dimensions and 16 convolutional channels

6 Challenges and Lessons Learned

Although the results aren't very impressive, I learned a lot of valuable lessons when working on this project.

I encountered a lot of challenges during the modeling and training phase. For GANs, the mode collapse problem rendered the model unusable. Some possible solutions include to modify the training schedule of the generator and the discriminator. For instance, to pre-train the discriminator to give it a head start, and to update the discriminator more frequently than the generator in a training epoch. This will make sure that the generator can't fool the discriminator too easily. For VAEs, the issue with low-resolution samples isn't unexpected, as VAEs are designed to learn the mean of the data distribution. This makes it impossible to achieve pixel-level precision in generated images. These samples thus differ greatly from images in the Kaggle dataset where an image could have as few as one pixel of tumor tissue and is still labeled positive. Because of this issue, the generated samples are too blurry to be useful as histopathologic images for training classifiers.

I also had some issue with computing resources. The time scope of the project wasn't sufficient for me to fine-tune hyperparameters in the model architecture like the number of convolutional layers or for training like learning rates and types of optimizers. I also found it challenging to train very large models. My GPU ran out of memory when computing gradients for a VAE with 4000 latent dimensions and 256 convolutional channels, which was roughly a 7-GB model.

For next year's students, it'd be great if they have the opportunity to discuss their ideas and read each other's project proposals. I only found out that Zixuan is also doing a project on VAE on the day of our presentation. If we had the opportunity to communicate, I could have pointer him to the Kaggle dataset which contains labeled images that the GDC doesn't provide. I could also have improved the resolution of the VAE-generated images using the

VQ-VAE2 model.

7 Conclusion and Next Steps

The most important takeaway for me is that generative models that perform well on some datasets might fail to translate into a highly specific domain like histopathologic images. I prototyped my generative models on the MNIST dataset and the CelebA-HQ dataset, and was able to generate realistic-looking images. One possible explanation is that, unlike digits or human faces which are highly structured, histopathologic images, especially pixels of tumor tissues, are highly variable. The variability in the original data distribution makes it difficult to model the distribution using generative models.

There are a few possible next steps for my project design. First, I can improve the VAE architecture and use models like the VQ-VAE2 to generate high-resolution images. Second, I can try stabilizing GAN's convergence with trial and error. Finally, I can experiment with training classifiers on a mixture of real and generated data. Althogh the samples generated by a vanilla VAE are too blurry to be useful on their own, they might have captured some high-level representation, or, some "average" of characteristics that distinguish tumor images from normal ones. These features might help reduce noise, biases, and chance of overfitting for the classifier. With this approach, the proportion of real and generated data will be another interesting hyperparameter to tune.

8 References

1. Tschuchnig, M. E., Oostingh, G. J., & Gadermayr, M. (2020). Generative Adversarial Networks in Digital Pathology: A Survey on Trends and Future Potential. arXiv preprint arXiv:2004.14936.

2. Raza, K., & Singh, N. K. A tour of unsupervised deep learning for medical image analysis. arXiv 2018. arXiv preprint arXiv:1812.07715.

3. Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In Advances in Neural Information Processing Systems (pp. 14866-14876).

4. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.

5. Kaggle dataset: https://www.kaggle.com/c/histopathologic-cancer-detection/data

6. Kaggle notebook reference for project understanding: https://www.kaggle.com/qitvision/a-complete-ml-pipeline-fast-ai

9 Code and Presentation Deck

All the code I have written for this project is on my GitHub: https://github.com/RuolinZheng08/cmsc33750-machine-learning-and-cancer/tree/main/project. Please refer to the README file in the project repository for more details. My presentation deck is in a shared Google Slides file: https://docs.google.com/presentation/d/19K1f9nOim8Pkk4uDX1txeSIfWUSSsWA/edit?usp=sharing.