Week 1

- General topic field: GANs
- Maybe: generate image by text queries.
- Be familiar with the progress of GANs in these years. Read literature review papers about GANs.
- Install LaTeX and texmaker, play with it.
- Look at the conference, CVPR (annual), ICCV(odd year), ECCV(even year)

Week 2

Proposal

- Problem statement

Medical image datasets are always highly imbalanced due to the rare pathologic cases. We need to generate high quality images of the minority. In this project, we will apply GANs to synthesis images for data augmentation. With the same baseline model, we will compare the improvement of the augmented dataset.

- Dataset

At the beginning, dataset 1 is used to build our model. It is easy to train and tune. Then, if we have time, dataset 2 (over 100GB) would be used. With the leaderboard, it is better to examine the performance of our model.

1) Small dataset: Red blood cells (from ML II exams).

2) Large dataset: SIIM-ISIC Melanoma Classification. Identifying melanoma in lesion images (from Kaggle ongoing competition).

- Technique and framework

Generative Adversarial Networks (GANs).

- Evaluation and metrics

Evaluation of GAN model:

Inception Score: measure the image quality.

Multi-scale Structural Similarity (MS-SSIM): measure the diversity and avoid mode collapse.

Evaluation of classifiers:

Area under the ROC curve (AUC). In medical diagnosis, we care more about finding out all the positive cases even if some images are misclassified into positive labels. (i.e. sensitivity or recall rate)

- Reference materials

GAN-based Augmentation

Huang, Sheng-Wei, Che-Tsung Lin, Shu-Ping Chen, Yen-Yi Wu, Po-Hao Hsu, and Shang-Hong Lai. "Auggan: Cross domain adaptation with gan-based data augmentation." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 718-731. 2018.

Shin, Hoo-Chang, Neil A. Tenenholtz, Jameson K. Rogers, Christopher G. Schwarz, Matthew L. Senjem, Jeffrey L. Gunter, Katherine P. Andriole, and Mark Michalski. "Medical image synthesis for data augmentation and anonymization using generative adversarial networks." In *International workshop on simulation and synthesis in medical imaging*, pp. 1-11. Springer, Cham, 2018.

Kazeminia, Salome, Christoph Baur, Arjan Kuijper, Bram van Ginneken, Nassir Navab, Shadi Albarqouni, and Anirban Mukhopadhyay. "GANs for medical image analysis." *arXiv preprint arXiv:1809.06222* (2018).

Han, Changhee, Kohei Murao, Tomoyuki Noguchi, Yusuke Kawata, Fumiya Uchiyama, Leonardo Rundo, Hideki Nakayama, and Shin'ichi Satoh. "Learning more with less: Conditional PGGAN-based data augmentation for brain metastases detection using highly-rough annotation on MR images." In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 119-127. 2019.

Other Augmentation (Semi-Supervised Learning)

He, Tong, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. "Bag of tricks for image classification with convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 558-567. 2019.

Berthelot, David, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A. Raffel. "Mixmatch: A holistic approach to semi-supervised learning." In *Advances in Neural Information Processing Systems*, pp. 5050-5060. 2019.

GANs and Literature Review

Gui, Jie, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. "A review on generative adversarial networks: Algorithms, theory, and applications." *arXiv preprint arXiv:2001.06937* (2020).

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672-2680. 2014.

Zhang, Han, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks." *arXiv preprint arXiv:1805.08318* (2018).

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C. Courville. "Improved training of wasserstein gans." In *Advances in neural information processing systems*, pp. 5767-5777. 2017.

Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196* (2017).

Shaham, Tamar Rott, Tali Dekel, and Tomer Michaeli. "Singan: Learning a generative model from a single natural image." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4570-4580. 2019.

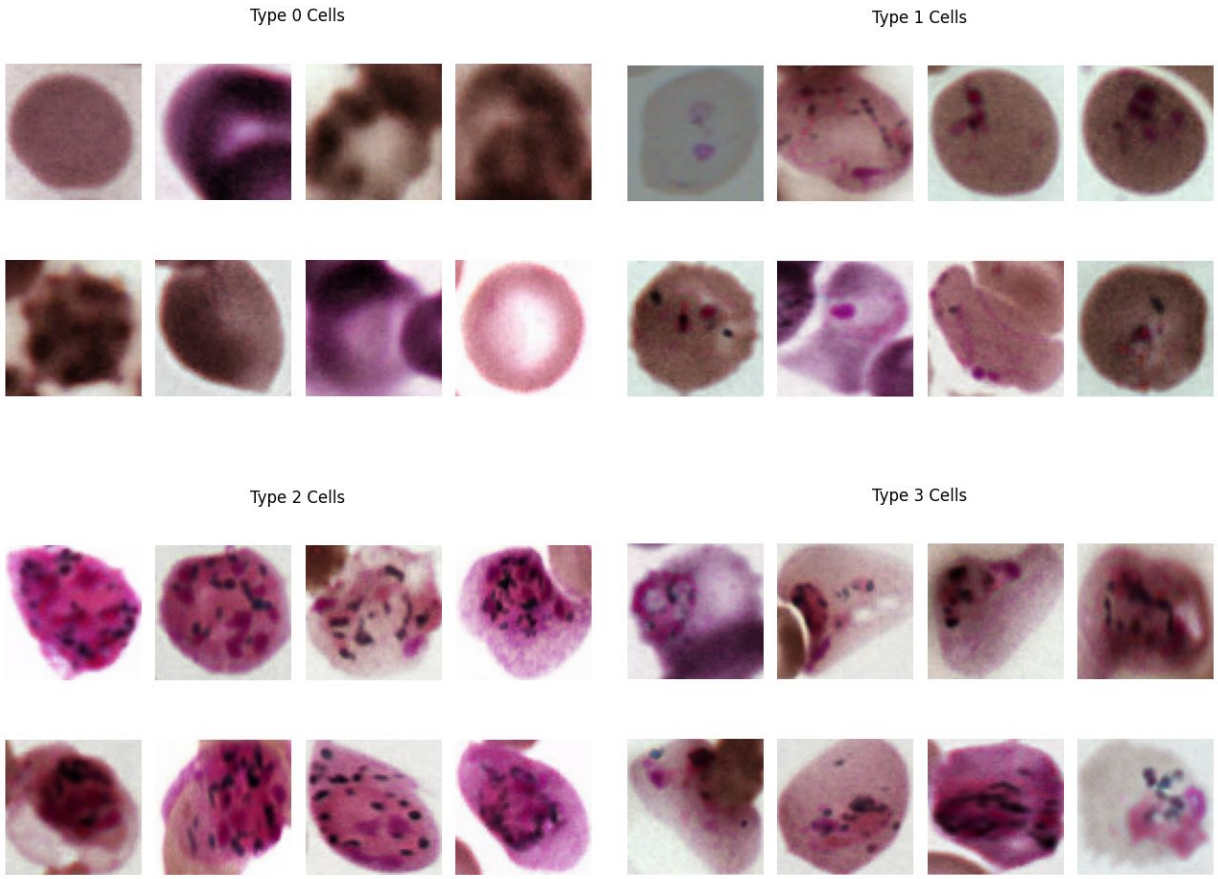
- Tentative work

1. Use GANs to generate a group of images. -> Apply a prior classifier to soft label these images. -> send true data and generated data into the model
2. Train a GAN with the label-0 data (healthy) -> get the D as a pretrained model -> train a GAN with the label-1 data -> generated data -> send all data to train the pretrained D
3.

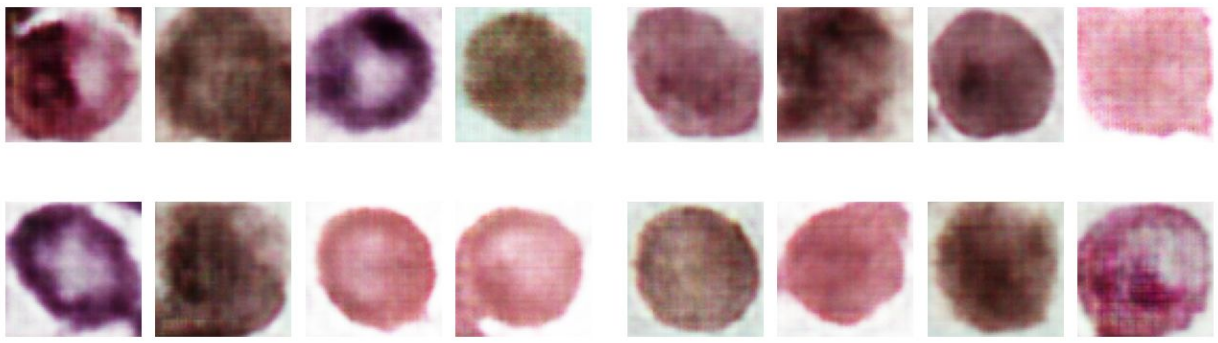
Week 3

- Construct the first GAN model, DCGAN. Generate samples.
- Inception Score is not stable (by literature, not my experiment). Change the evaluation metric to Frechet Inception Distance (FID). Define the FID function to evaluate the generated samples. (Still work on it)

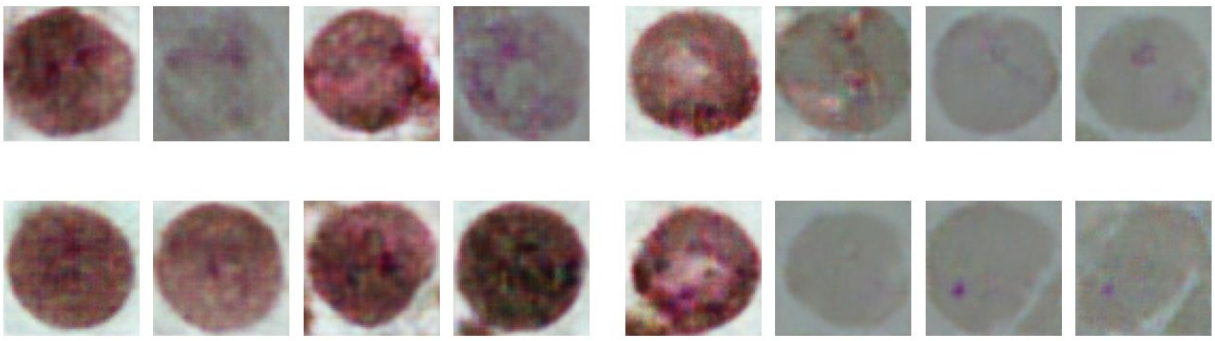
True Dataset



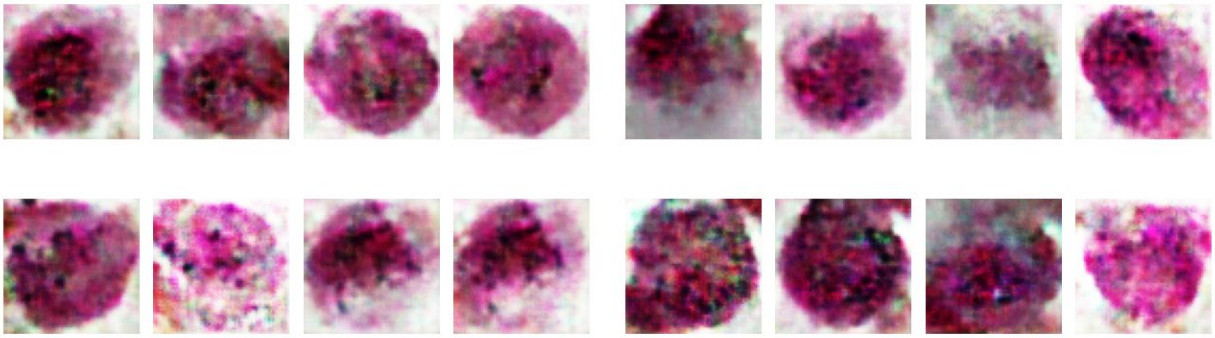
Generated samples (Input: all types of cells)



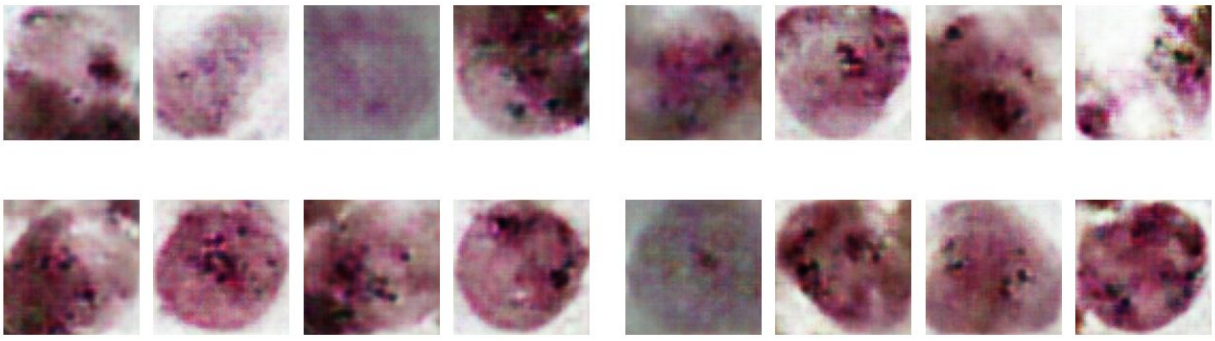
Generated samples (Input: one type of cells)
Type 1



Type 2



Type 3



Week 4

- FID score to test generated samples. (Continued, from last week)

Note: the lower FID means the smaller distance between two distributions.

- Generate all-type samples. Comparing 1000 generated samples with 1000 real samples (in validation set).
  FID = 0.835
- Comparing 1000 real samples (in training set) with 1000 real samples (in validation set).
  FID = 0.004
- Comparing more...

Compare with real samples (Validation)	All-type (1000 samples)	Type1 (73 samples)	Type2 (27 samples)	Type3 (222 samples)
Generated samples	0.835	0.448	3.191	1.358
Real samples (Train)	0.004	0.022	0.098	0.013

- Run a baseline CNN model for cell classification, then try the original/augmented dataset. (Continued, from last week)

Without augmentation. - 30 epochs

```
Epoch 30/30
5508/5508 [==============================] - 1s 208us/step - loss: 0.1125 - accuracy: 0.9609 - val_loss: 0.1768
1722/1722 [==============================] - 0s 141us/step
Final accuracy on validations set: 93.90243887901306 %
Cohen Kappa 0.46529508789821983
F1 score 0.5056512774882392
Confusion Matrix:
 [[1400    0    0    0]
 [  67    1    0    5]
 [   6    0   10   11]
 [ 126    0    3   93]]
```

With 3000 augmented samples. 1000 for each type 1 & 2 & 3. - 30 epochs

```
Epoch 30/30
7908/7908 [==============================] - 1s 137us/step - loss: 0.1610 - accuracy: 0.9311 - val_loss: 0.1441
1722/1722 [==============================] - 0s 138us/step
Final accuracy on validations set: 97.16898798942566 %
Cohen Kappa 0.8182436328608045
F1 score 0.6472630194084553
Confusion Matrix:
 [[1395    0    0    5]
 [  27   20    0   26]
 [   0    0    6   21]
 [  11    0    5  206]]
```

- Improved the CNN structure.

Without augmentation. - 30 epochs

```
Epoch 30/30
5508/5508 [==============================] - 9s 2ms/step - loss: 0.0304 - accuracy: 0.9899 - val_loss: 0.0931
1722/1722 [==============================] - 1s 524us/step
Final accuracy on validations set: 97.45935201644897 %
Cohen Kappa 0.8369885390478119
F1 score 0.7046201208696157
Confusion Matrix:
 [[1397    1    0    2]
 [  25   19    0   29]
 [   0    0   14   13]
 [   6    0   10  206]]
```

With 3000 GAN-based augmented samples. - 30 epochs

```
Epoch 30/30
7908/7908 [==============================] - 12s 2ms/step - loss: 0.0197 - accuracy: 0.9928 - val_loss: 0.0536
1722/1722 [==============================] - 1s 515us/step
Final accuracy on validations set: 98.21428656578064 %
Cohen Kappa 0.8857702887626613
F1 score 0.7920020300451476
Confusion Matrix:
 [[1397    2    0    1]
 [  10   44    0   19]
 [   0    1   15   11]
 [   3    5   10  204]]
```

With traditional augmented samples (by ImageDataGenerator). - 30 epochs

```
Epoch 30/30
173/172 [==============================] - 9s 52ms/step - loss: 0.0637 - accuracy: 0.9804 - val_loss: 0.0775
1722/1722 [==============================] - 1s 474us/step
Final accuracy on validations set: 97.54645824432373 %
Cohen Kappa 0.8725520671433012
F1 score 0.776112663150345
Confusion Matrix:
 [[1391    4    0    5]
 [   8   48    0   17]
 [   0    2   14   11]
 [   3    7   13  199]]
```

With both augmentation. - 30 epochs

```
Epoch 30/30
248/247 [==============================] - 13s 52ms/step - loss: 0.0672 - accuracy: 0.9761 - val_loss: 0.0927
1722/1722 [==============================] - 1s 477us/step
Final accuracy on validations set: 97.69163727760315 %
Cohen Kappa 0.8811971110182075
F1 score 0.8155025028935365
Confusion Matrix:
 [[1380   11    0    9]
 [   1   55    0   17]
 [   0    2   19    6]
 [   2    5   14  201]]
```

Question: These results depend on the hyperparameters (epochs, learning rate, ...). How can I prove the augmented model is better than the non-augmented model? Are these results reliable? Solution: plot of val_loss? -- pretrained models

Tried:

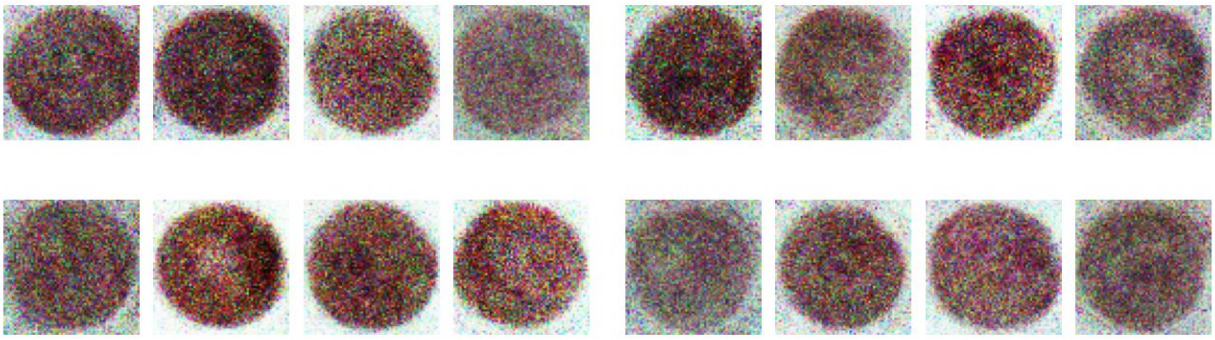
- Guess: CNN-based GAN could be well understood by CNN-based classifiers. SImilarly, MLP-based GAN could be well understood by MLP-based classifiers.
- I tried to drop generated samples (by CNN-based GAN or by MLP-based GAN) into a MLP classifier. By comparison, there were no obvious differences. Not sure. Still the problem mentioned before. I don't know how to compare these two cases in a robust way.

Problems and potential solutions. (Try later)

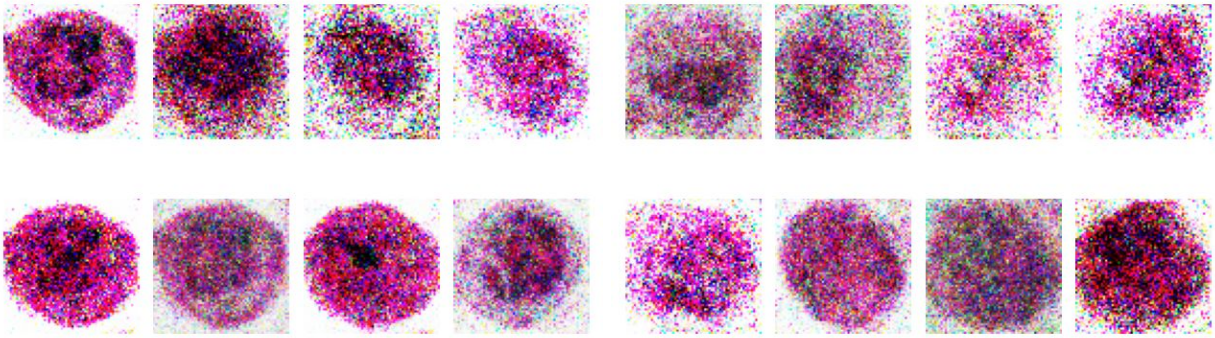
- For each type of cell, I need to build a GAN model to get the generated samples. It takes a lot of computational power and time. We need to consider generating all types of cells by one GAN model.

- Up to now, we've finished a rough pipeline for the GAN-based image recognition. It can successfully run in a simple dataset.
- The input dataset could be other datasets. This dataset is kind of simple. The baseline model without augmentation can easily reach a high performance.
- Apply some advanced GAN models to improve the FID score of generated samples. E.g. Balancing GAN

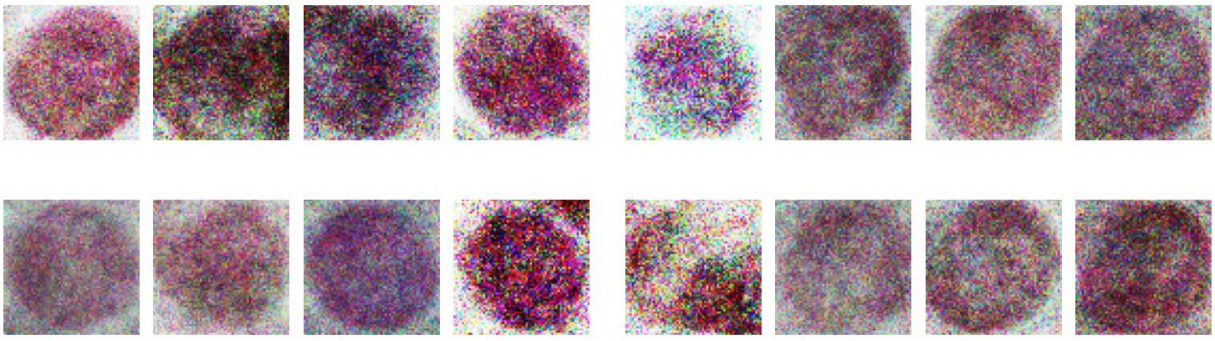
Type1 by mlpGAN



Type2 by mlpGAN

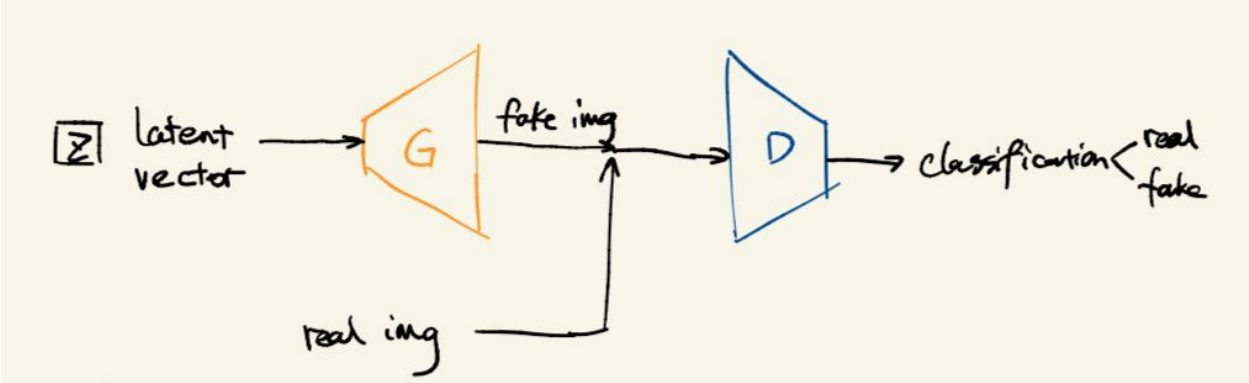


Type3 by mlpGAN

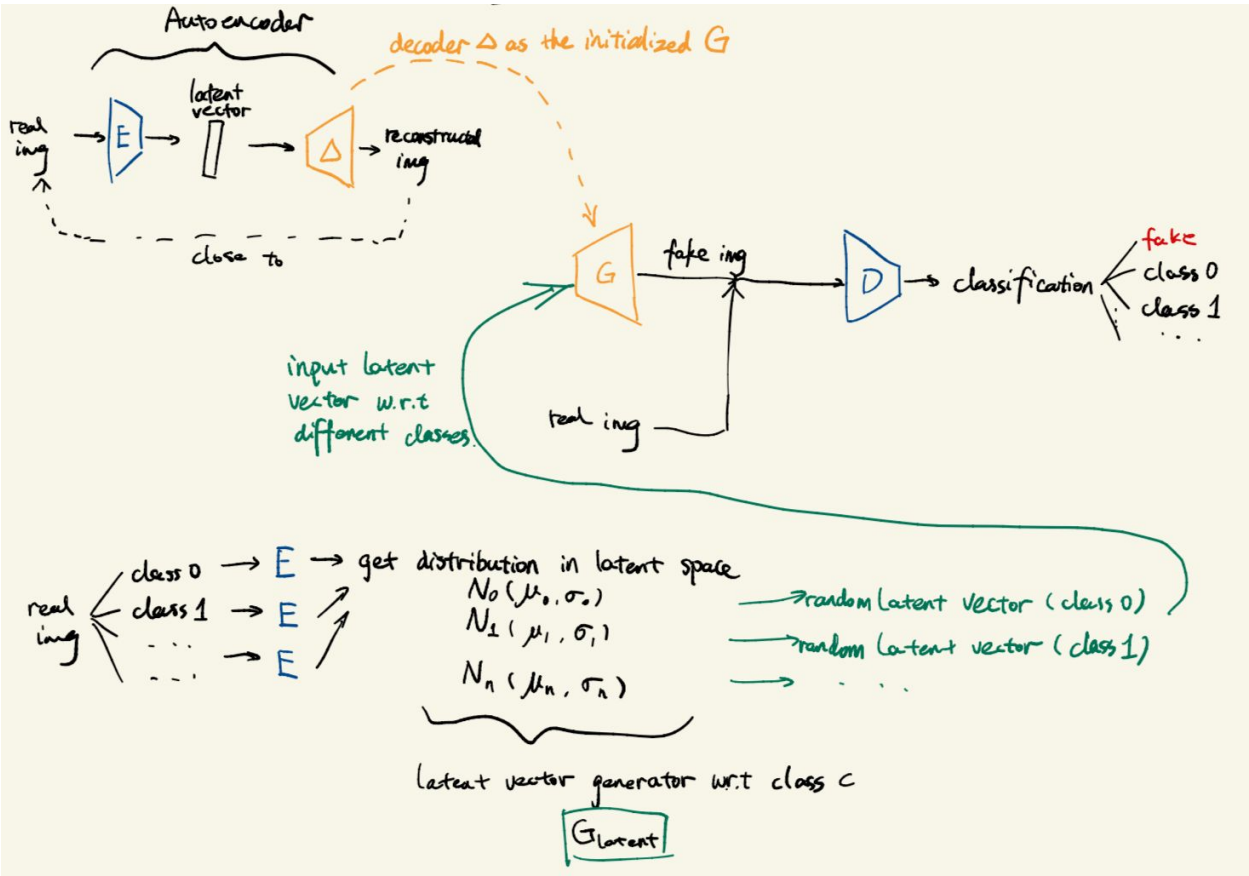


Week 5

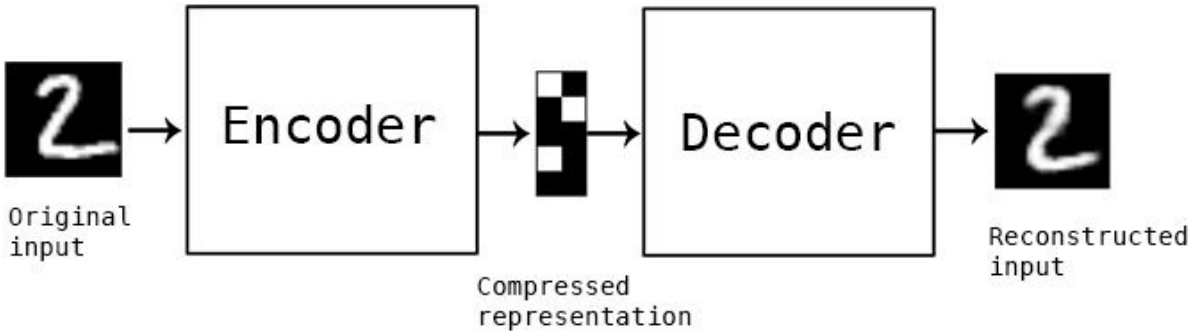
- Construct the Balancing GAN. (still work on it)
    - Previous GAN



    - Balancing GAN



    - Improvement: Replacing KL-JS divergence as Wasserstein distance.
    - Autoencoder:



- Try a pretrained model (ResNet50) for cell classification. The performance is worse than the simple model. It only predicts majority classes. → Try a large and complex dataset.
- Load and preprocess a large dataset from Kaggle. (siim-isic-melanoma-classification)

Week6

Finish the BAGAN construction. It can work in MNIST Fashion dataset, but not in cell-classification or siim-isic-melanoma-classification.

- Try normalization in latent vectors.

Autoencoder:

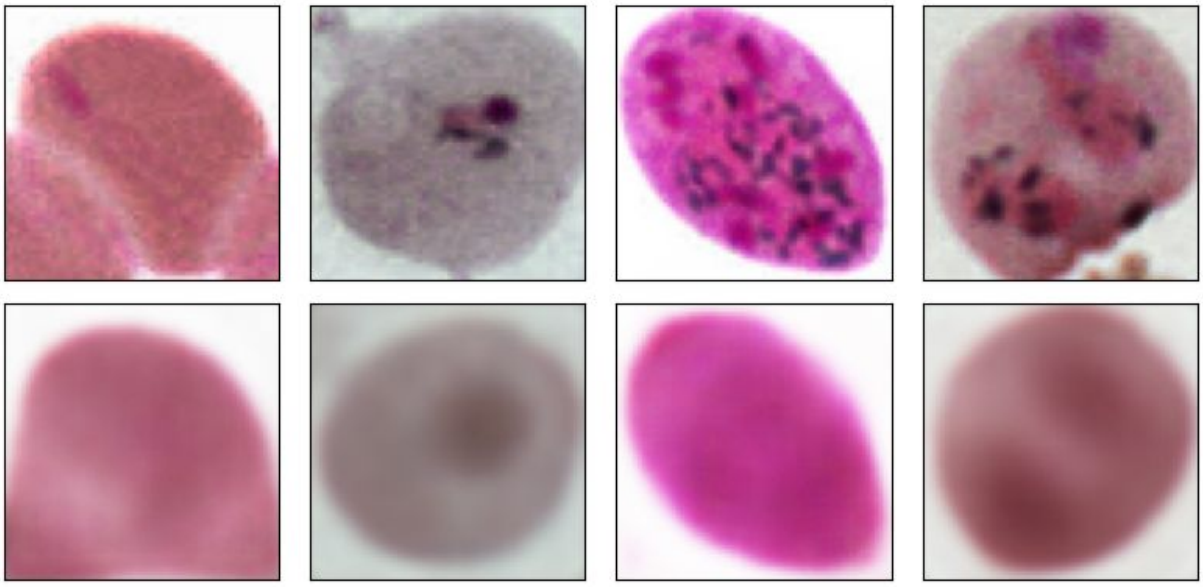
- reconstructed images (1st row: real, 2nd row: reconstructed)

- With randomly generated latent vector w.r.t classes (1st row: real, 2nd row: generated)

BAGAN:

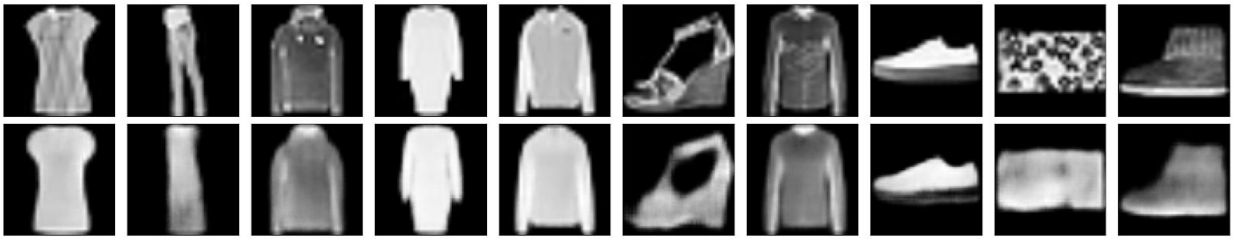
Week7

Improved Autoencoder (loss: mean absolute error)

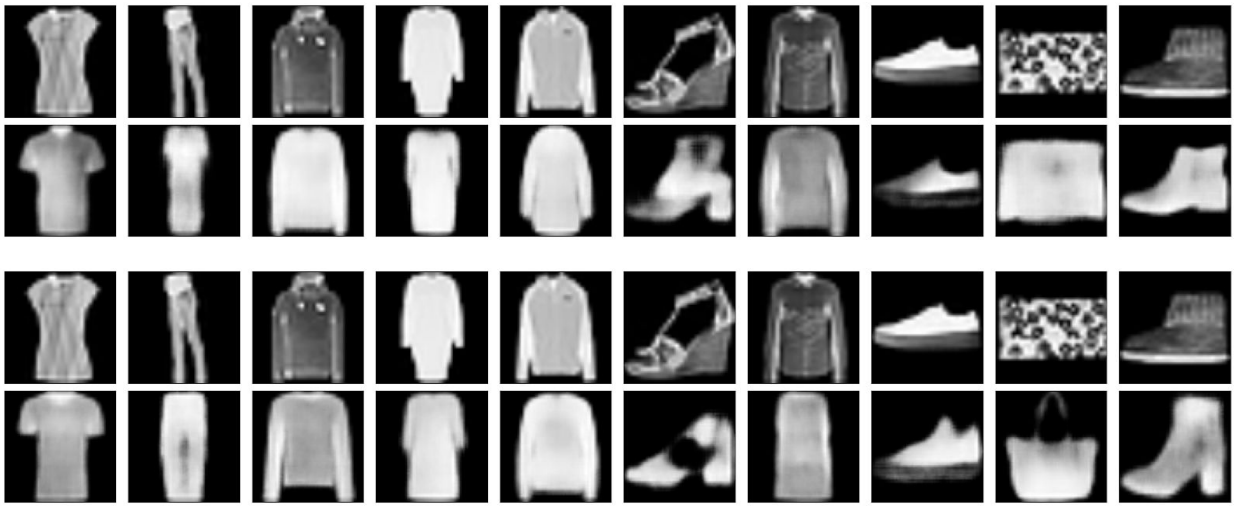


Imbalance dataset (only 100 trousers + 4200 per other cloth)

Reconstructed



Generated

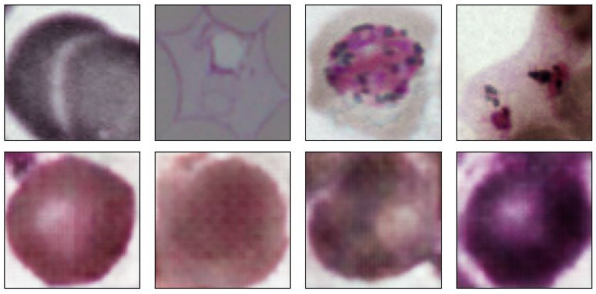


Week8

WGAN (finished last week)

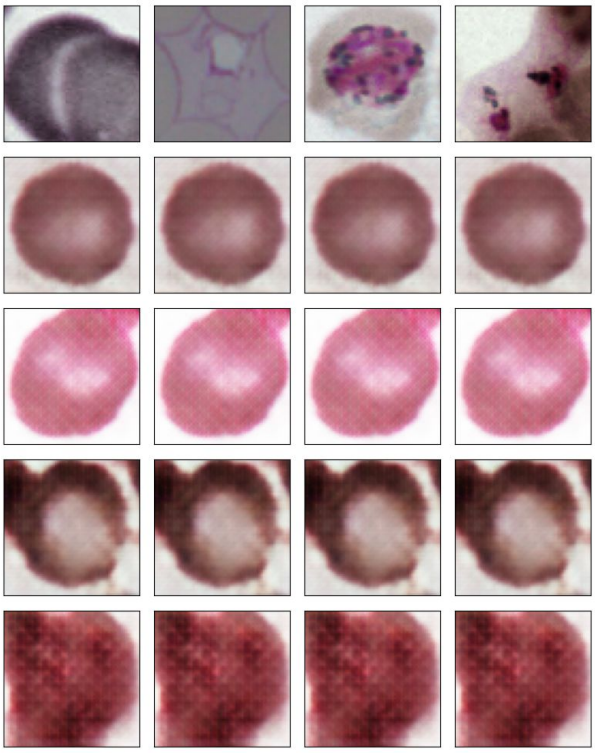
WGAN-GP (finished)

- Very close performance compared to BAGAN.
- FID(alltype cells, WGAN-GP) = 0.039, FID(alltype cells, BAGAN) = 0.038



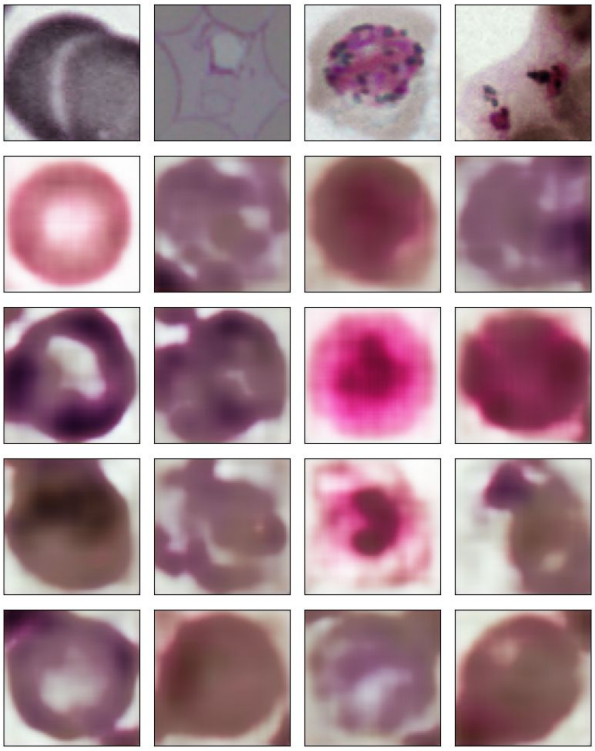
Conditional WGAN-GP (constructed, but some problems in generating different classes)

- Issue: combine Wasserstein metric (real/fake) with cross-entropy metric (diff classes).



Tips: data type ‘float32’ is important in the tensorflow framework!!!

Week9
Answer to previous issue: I cannot simply add the cross-entropy loss and Wasserstein loss together to create a new loss function. Thus, I should not change the output of the Discriminator.
Conditional WGAN-GP (finished)
Add {Embedding, Multiply} layers in both G and D.
(Latent + Label) -> Generator -> Image
(Image + Label) -> Discriminator -> Validity Score



Aggregate Conditional WGAN-GP into BAGAN (ongoing)

In BAGAN, the latent vectors are generated based on normal distribution. (assume each class of latent vectors are distributed normally). If we add an embedding model to generate latent vectors w.r.t different classes.
*BAGAN: supervised latent generator
Image -> Encoder -> Latent vectors (+ Label) -> Embedding -> Labeled latent vectors -> Decoder -> Image

Finish the code of BAGAN with Wasserstein-GP off-line. (GCP cannot redeem codes, not sure the code works or not)
Writing my paper.