Deep Learning Colorectal Adenocarcinomas Nuclei Detection

Author: Zheng Zhao

Date: 2st Mar, 2017

Section I Brief

This experiment aims to analyze the performance of detecting nuclei in pathology images when utilizing state-of-the-art deep learning methods, especially autoencoders, which will strongly support my Ph.D. proposals. Recently, the generative model based autoencoder, "Variational Autoencoder (VAE)" was developed, and soon achieved great success, but our comparative results show that the KL divergence bypassed VAE outperforms all of the other algorithms, which is very interesting. In addition, the training methods for stacked VAE are not very compatible with layerwise methods. There's still a long way to go further into deep VAE designing.

Section II Algorithms:

Autoencoders (**sA**), Stacked Autoencoders (**stA**), Variational Autoencoders (**vA**), Stacked Variational Autoencoders (**vAs**), Variational Autoencoders w/o KL (**vA_noKL**). Stacked Variational Autoencoders w/o Layerwise (**vAs_noLW**). Their structures are shown below:

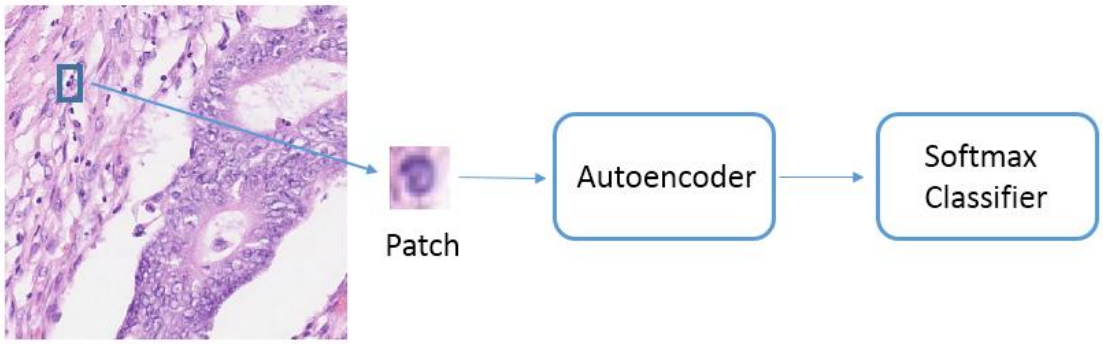


Fig 1. The overall structure of this detection system

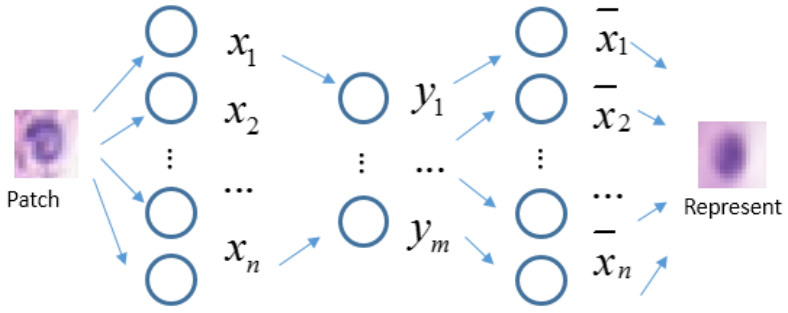


Fig 2. The structure of **sA**, n=867, m=64

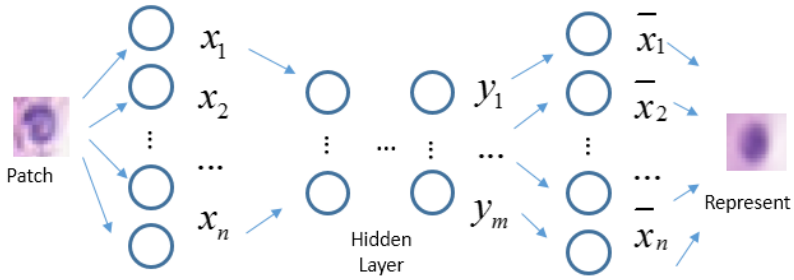


Fig 3. The structure of **stA**, n=867, m=64, hidden layer = 4. L1=512, L2=256, L3=128

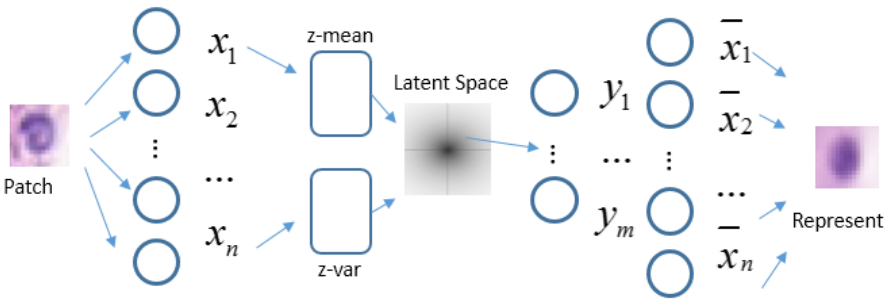


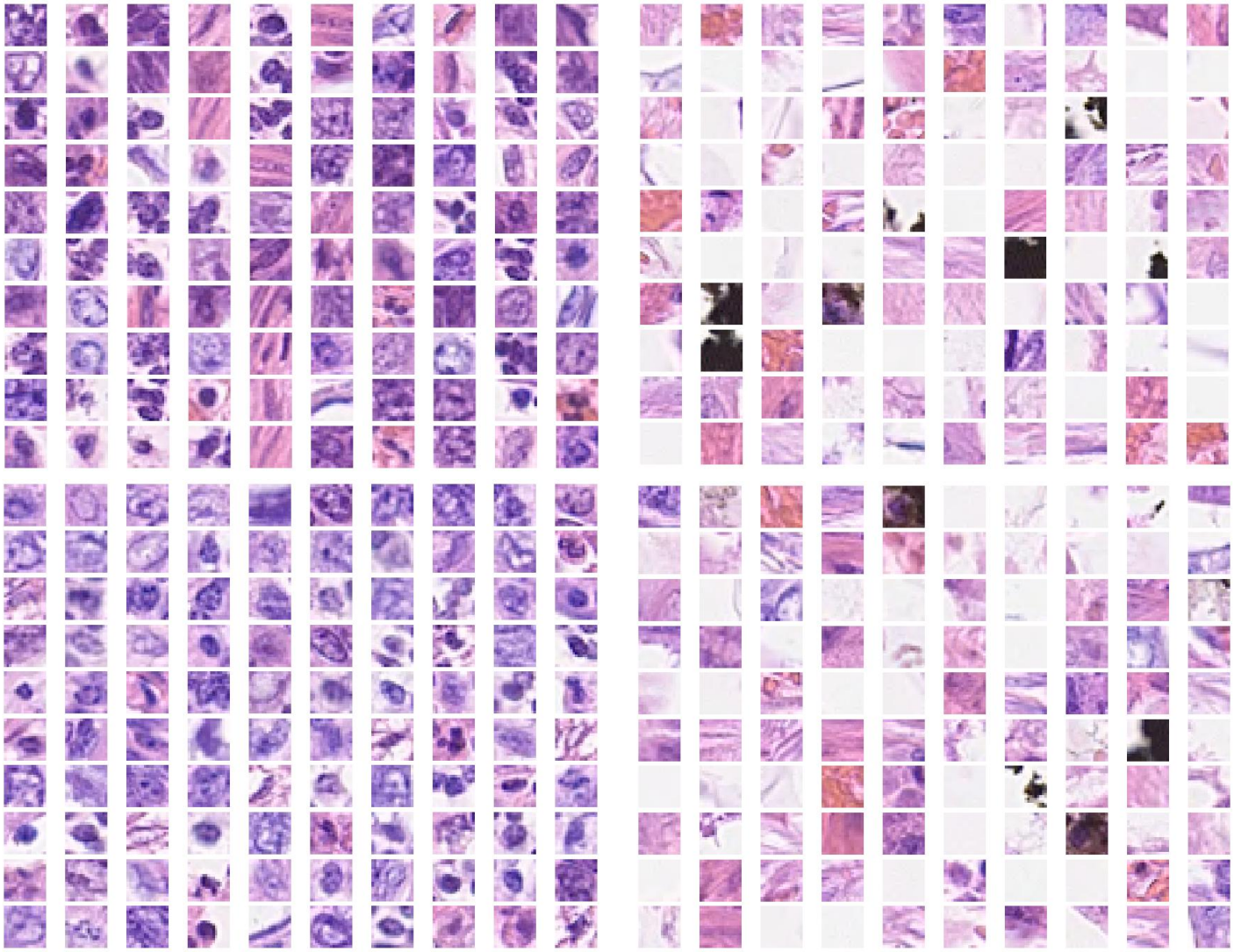
Fig 4. The structure of **vA**, n=867, m=64

For simplicity, the **vAs** and **vA_noKL** are not shown here. The classifier here we adopt is a two layer Softmax classifier. The optimizer to train ANNs is Adadelta, with cross entropy loss function.

Section III Datasets:

We adopt *CRCHistoPhenotypes* [1] datasets for autoencoder and object detection training, this dataset involves 100 H&E stained histology images of colorectal adenocarcinomas. A total of 29,756 nuclei were marked at/around the center for detection purposes. Here, the img01-img50 are taken as training sets, while img51-img100 are taken as test sets. For simplicity, we take 13,400 samples from each of them, divide into positive group and negative group. Positive contains only nuclei samples, and negative group contains background samples. We also manually randomly sample from the grouped datasets to guarantee its accuracy. Sampling program is written in Matlab, (Extraction_Positive_Script.m etc). The 10*10 samples from each groups are shown below: (The top two are samples from training sets, the bottom two are from test sets.)

Positive Sample **Negative Sample**



Section IV Environment:

PC, Intel Core I5 3.1GHz, 4GB, no GPU acceleration, Python, Matlab, Keras (Tensorflow backend), train ANNs with 50 epoch (time consuming considered).

Section V Experiments

1. Reconstruction Performance:

We first tested on autoencoders' reconstruction performance, which is usually regarded as a critical criterion. First two rows are positive samples (input and reconstruction respectively), second two rows are negative samples (input and reconstruction respectively). From theses reconstruction results, we can initially evaluate autoencoders' performance objectively.

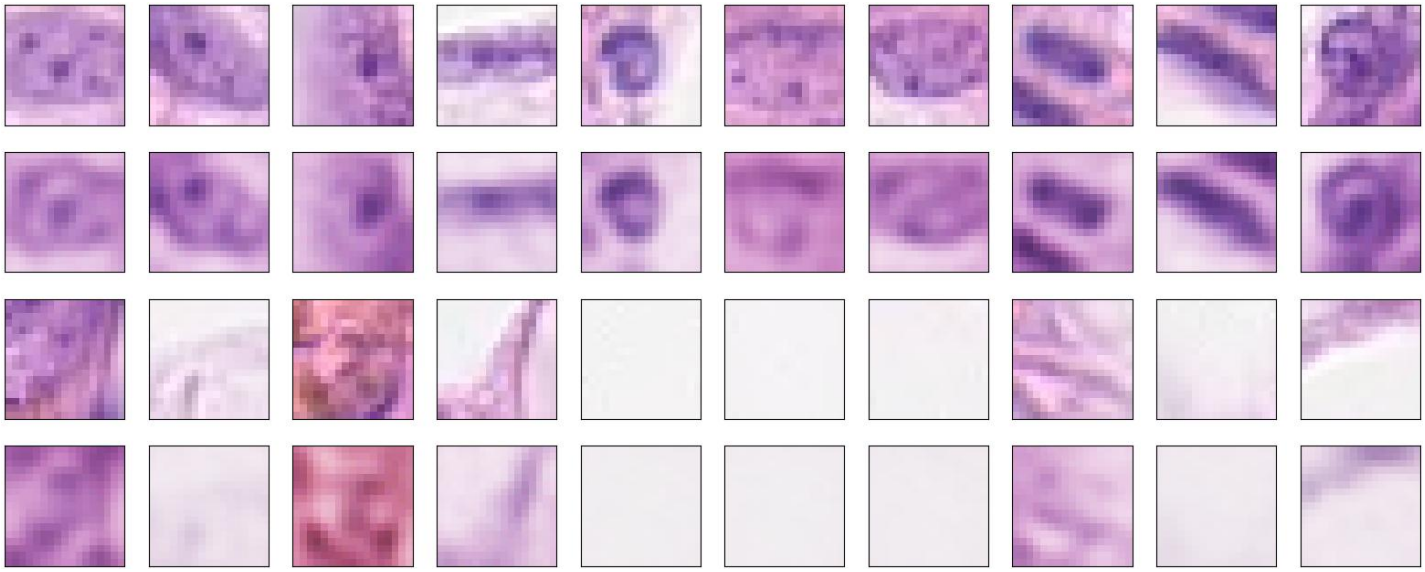


Fig 6. Representaion by **sA**

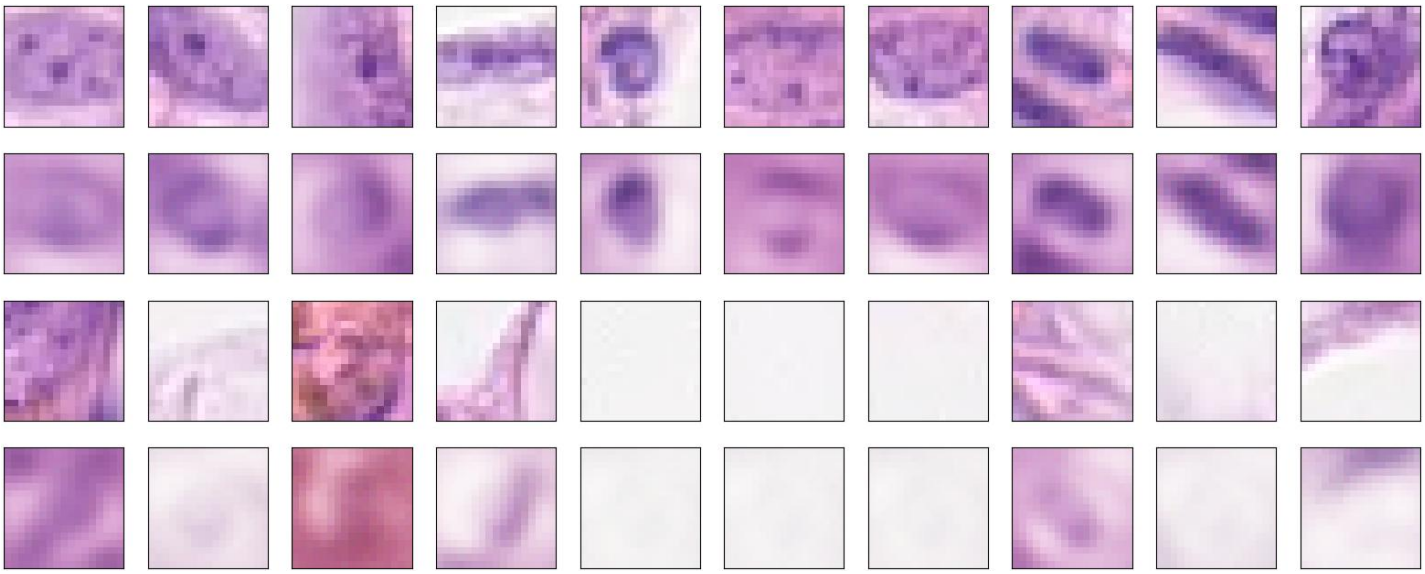


Fig 6. Representaion by **stA**

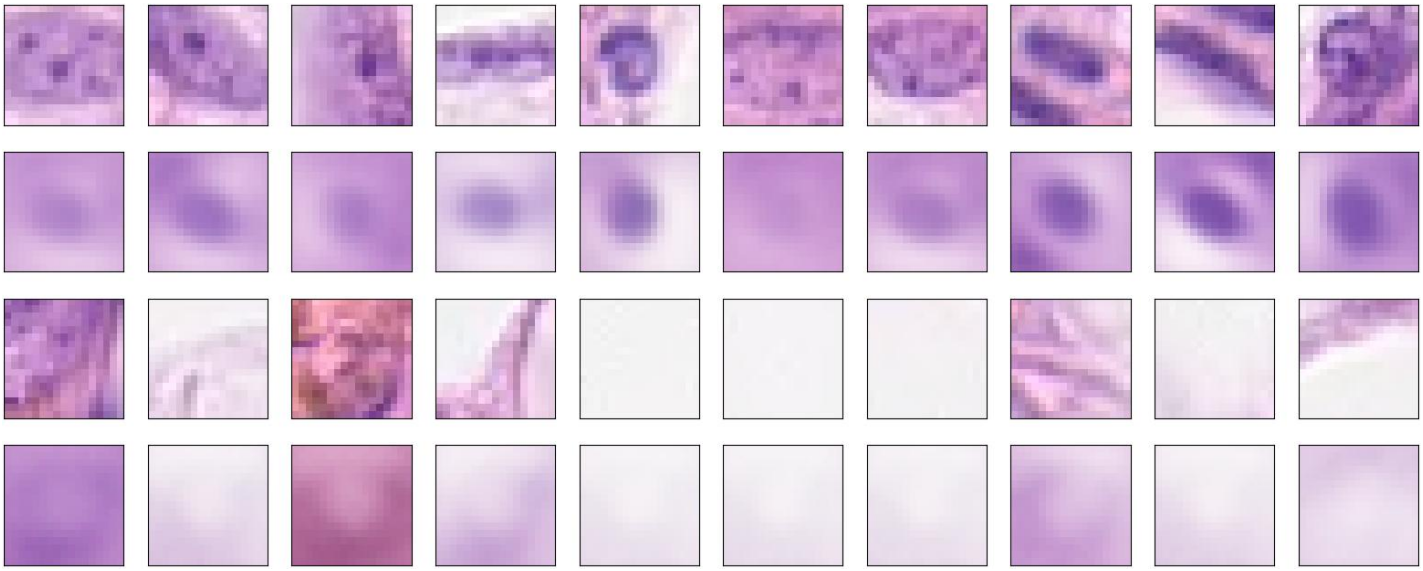


Fig 6. Representaion by **vA**

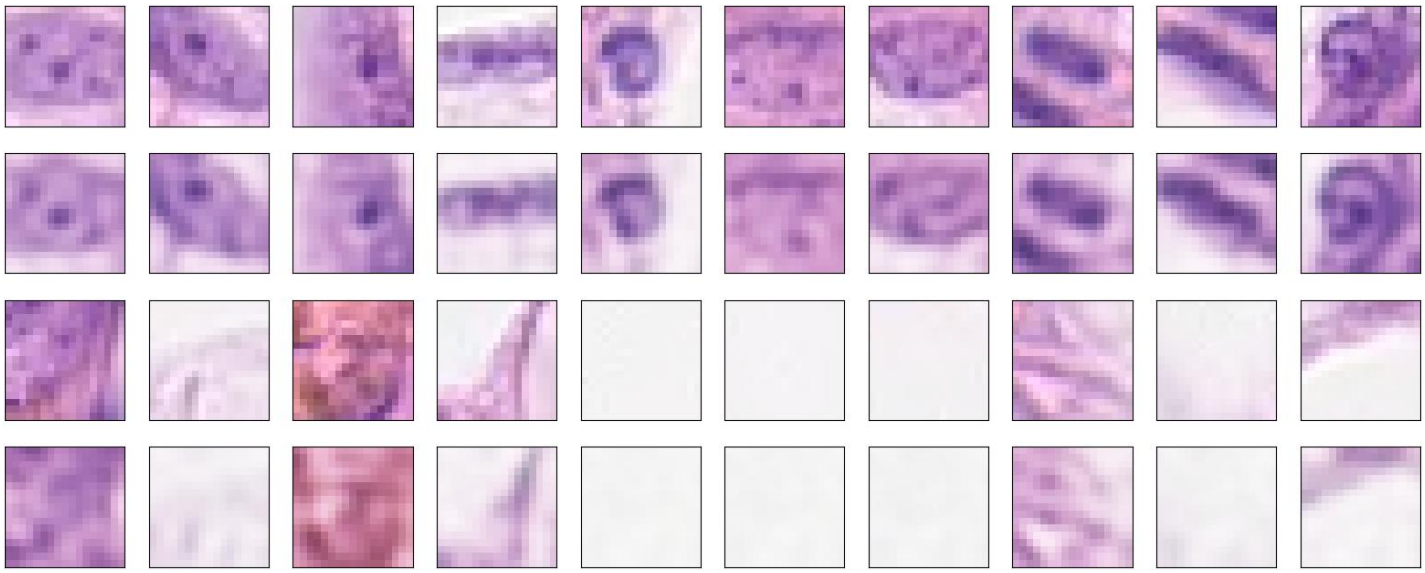


Fig 6. Representaion by **vA_noKL**

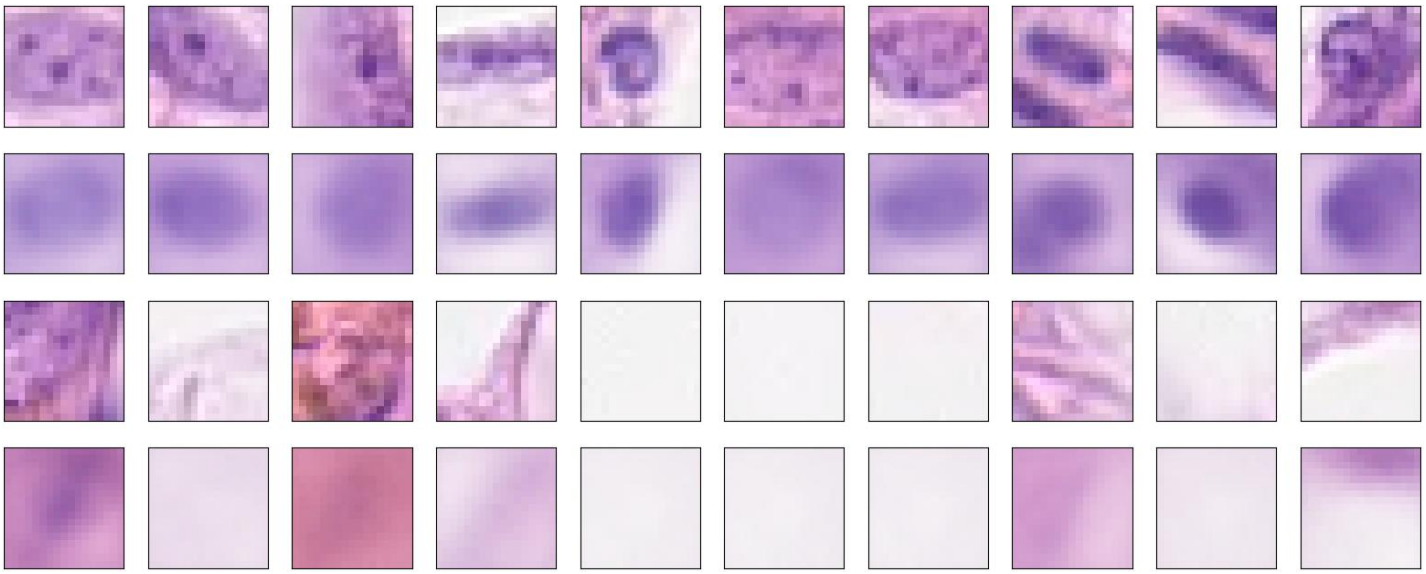


Fig 6. Representaion by **vAs**

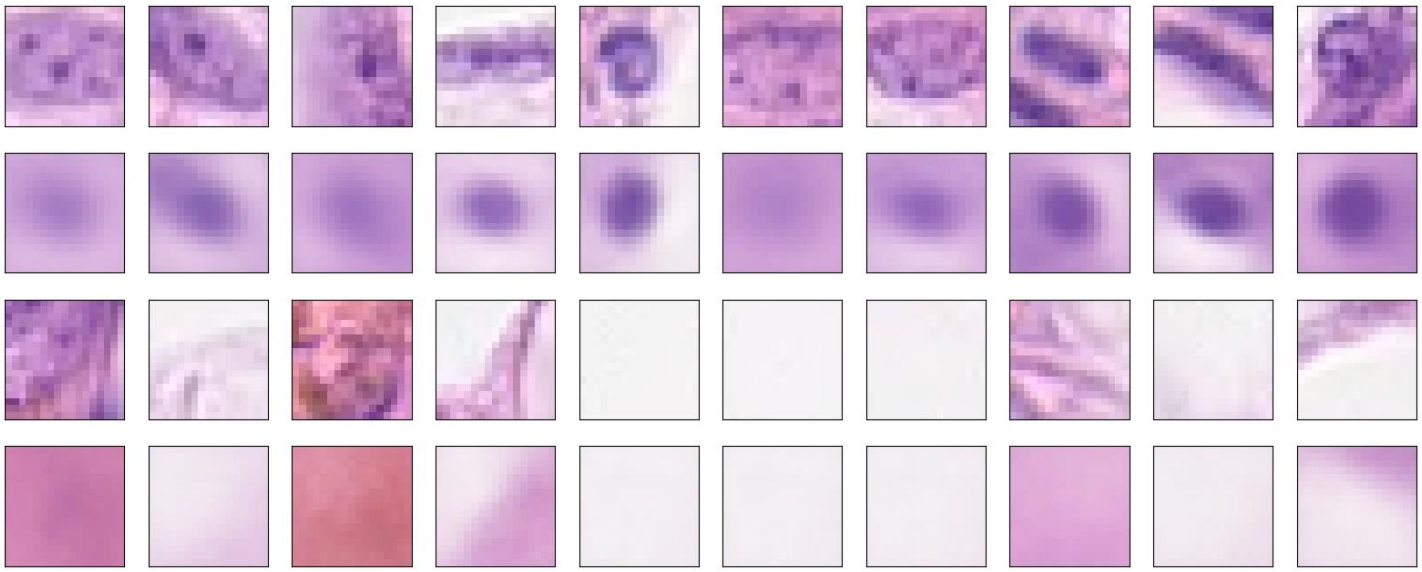


Fig 6. Representaion by **vAs_noLW**

Classifier Training Loss:

Here we use loss value of training classifier to quantify performance of each autoencoders. Because, autoencoders inherently aim to construct an encoded representation of input, and this encoded representation is what the classifier working on.

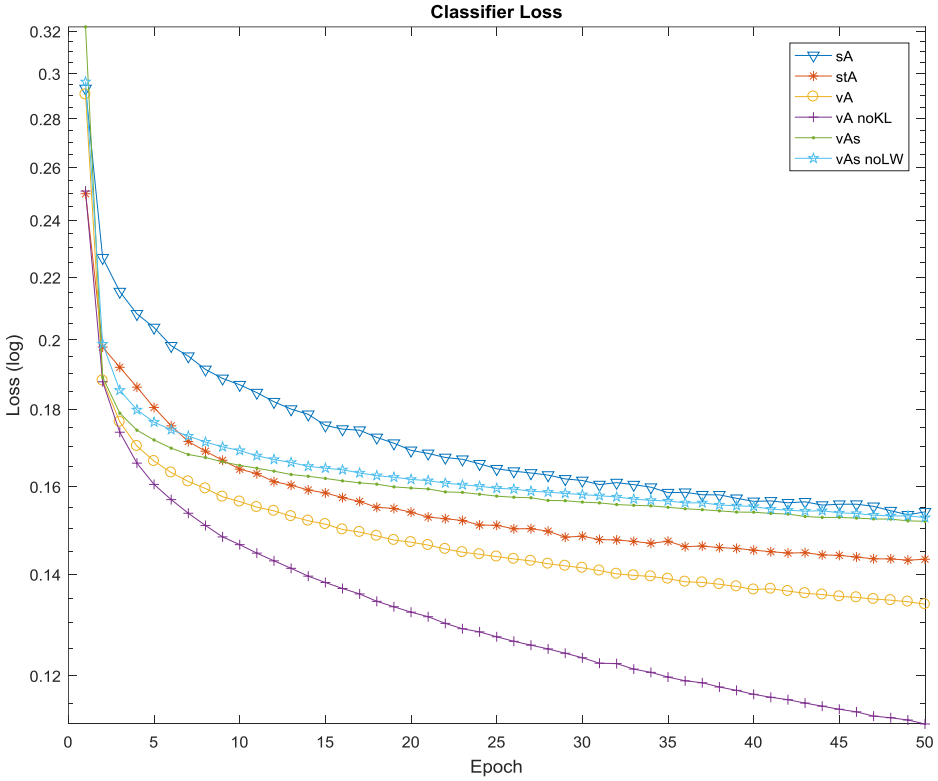


Fig 7. Classifier Loss by Autoencoders

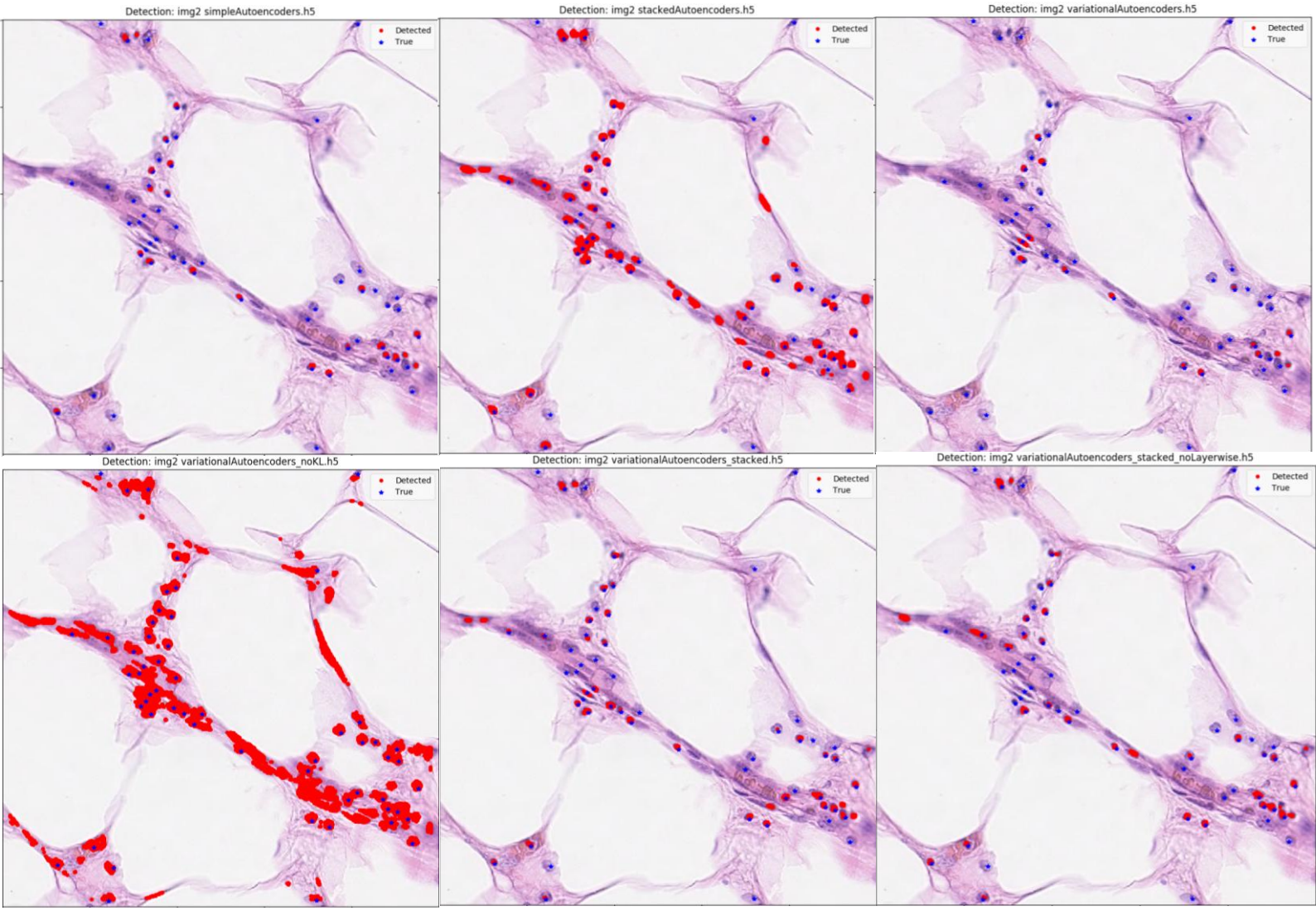
It can be concluded from the reconstruction images and Fig 7 that, **vA** (variational autoencoder) works better than **sA (**simple autoencoder) and **stA** (stacked autoencoder). But **vAs** (stacked

variational autoencoder), or we may call it deep variational autoencoder, did not perform very well as what we previously expected. It is even worse than non-stacked one. We trained the **vAs** with layer-by-layer greedy method, but it only improved very slightly compared with **vAs_noLW**, which means, the **vAs** currently lacks of efficient training method.

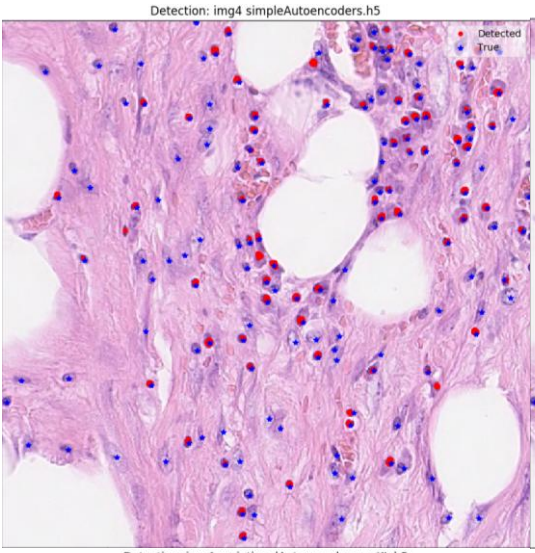
One significant finding, is that if we bypass the KL divergence constrain of **vA**, the performance (vA_**noKL**) will be dramatically improved. The possible reasons about these are briefly discussed in section VI

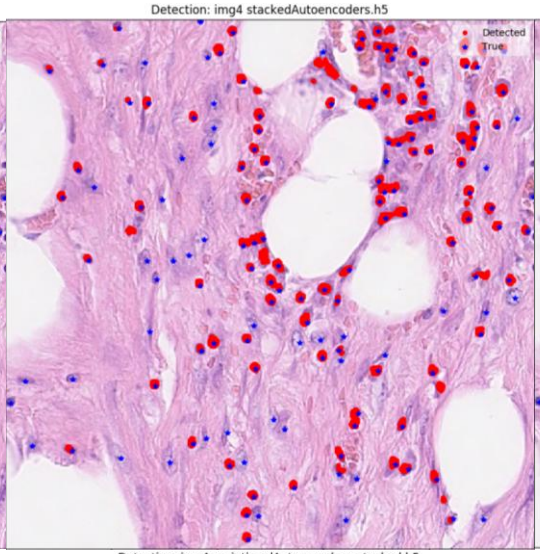
2. Nuclei Detection Performance:

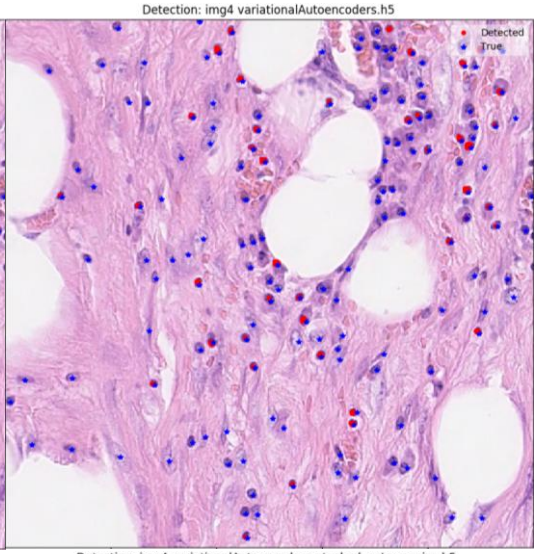
This experiment is divided into 6 distinct groups, which are img2, img4, img48, img55, img81, img89 respectively. Img2-img48 are chosen from the training set, while img55-89 are chosen from the test set. The trained classifier's task is to find the nucleis in the H&E histology image. The blue point with star marker are the true nucleis, manually identified by experts. The red area are potential nuclei area detected by the classifier, with threshold empirically set by 0.6.

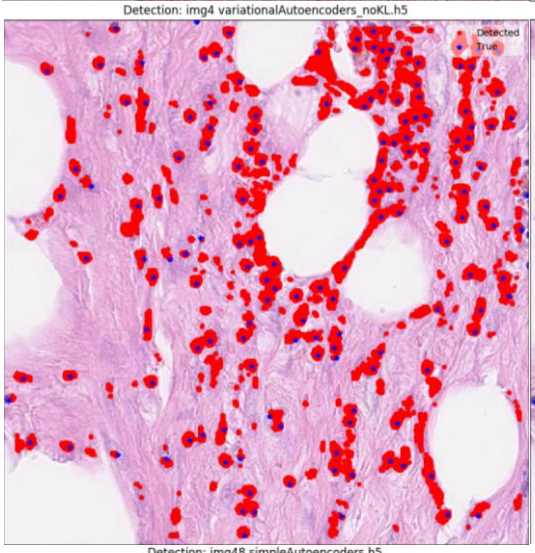


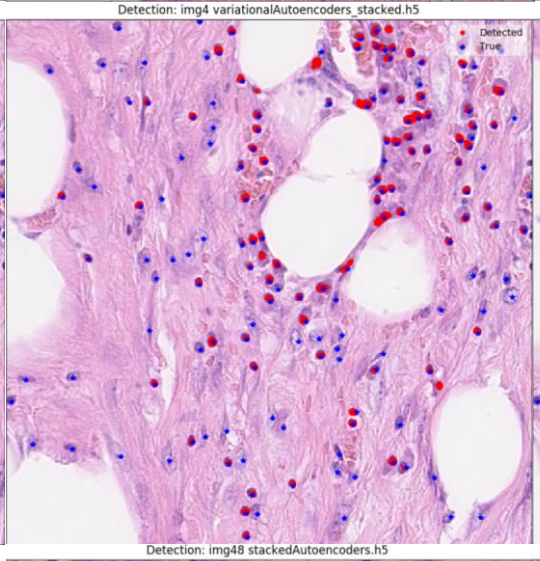
IMG 2

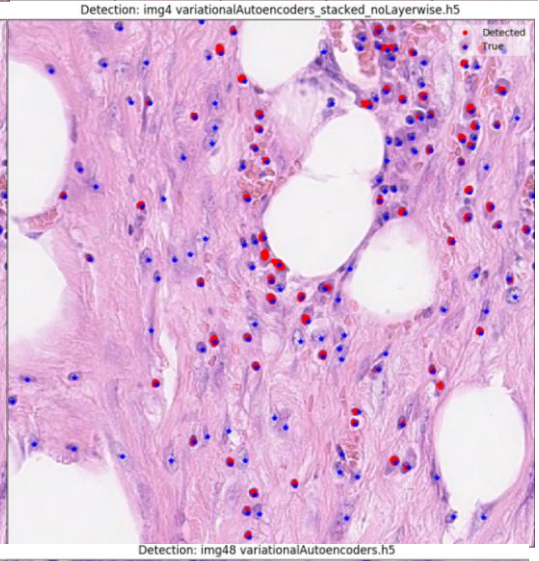

Detection: img4 simpleAutoencoders.h5
Detected
True

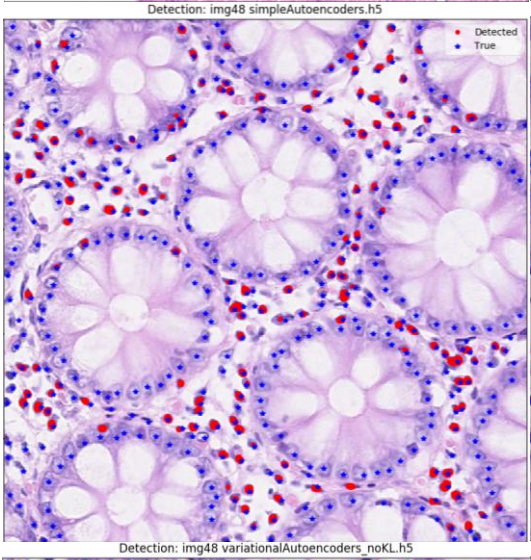

Detection: img4 stackedAutoencoders.h5
Detected
True

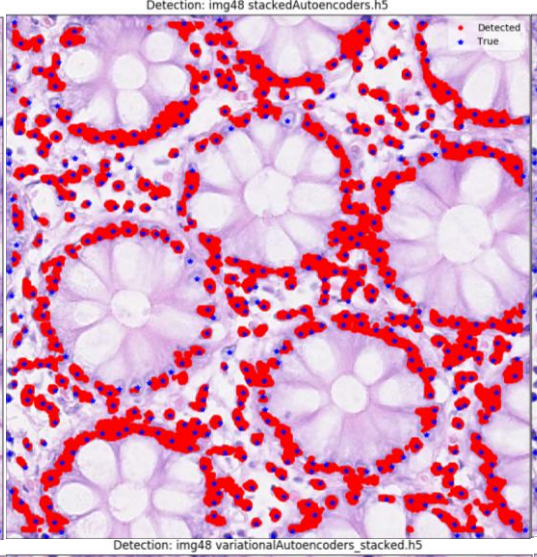

Detection: img4 variationalAutoencoders.h5
Detected
True

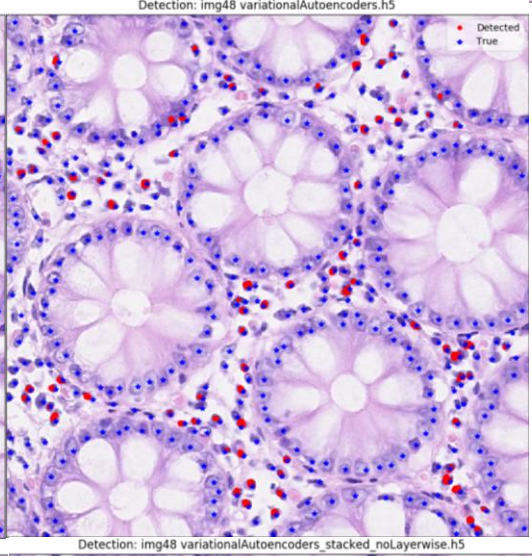

Detection: img4 variationalAutoencoders_noKL.h5
Detected
True

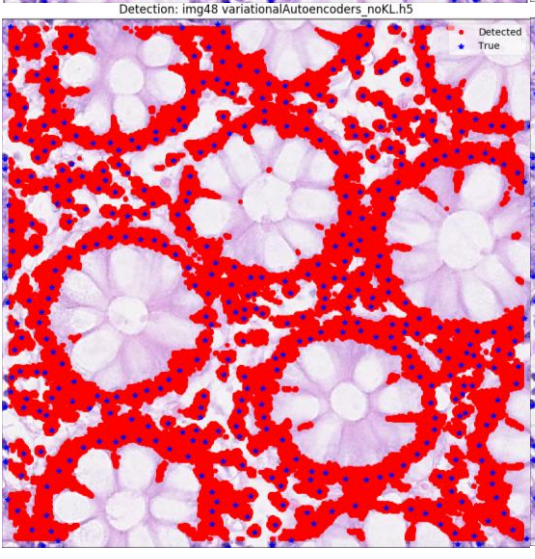

Detection: img4 variationalAutoencoders_stacked.h5
Detected
True

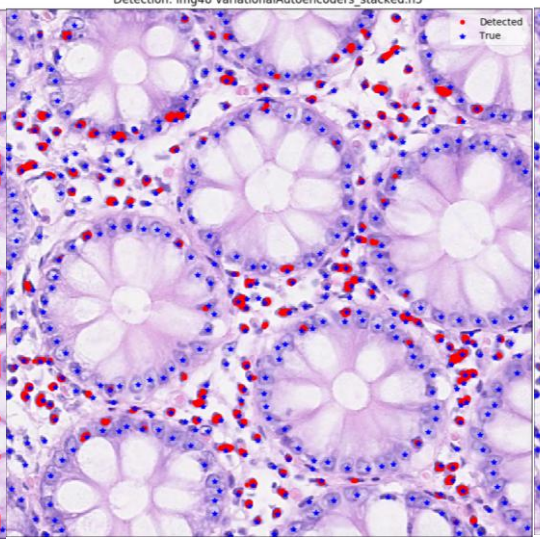

Detection: img4 variationalAutoencoders_stacked_noLayerwise.h5
Detected
True

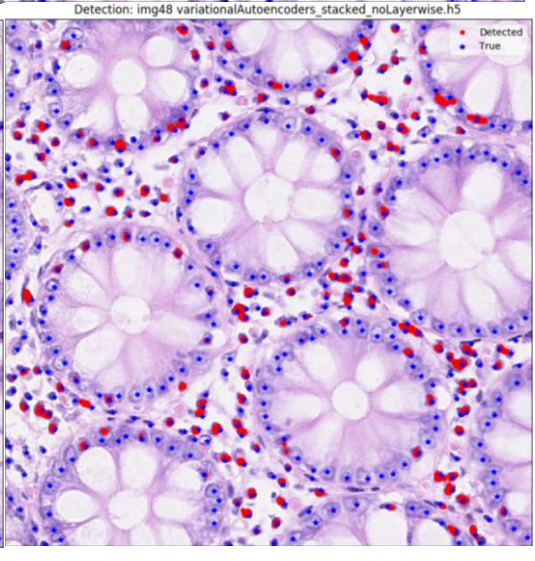

Detection: img48 simpleAutoencoders.h5
Detected
True


Detection: img48 stackedAutoencoders.h5
Detected
True

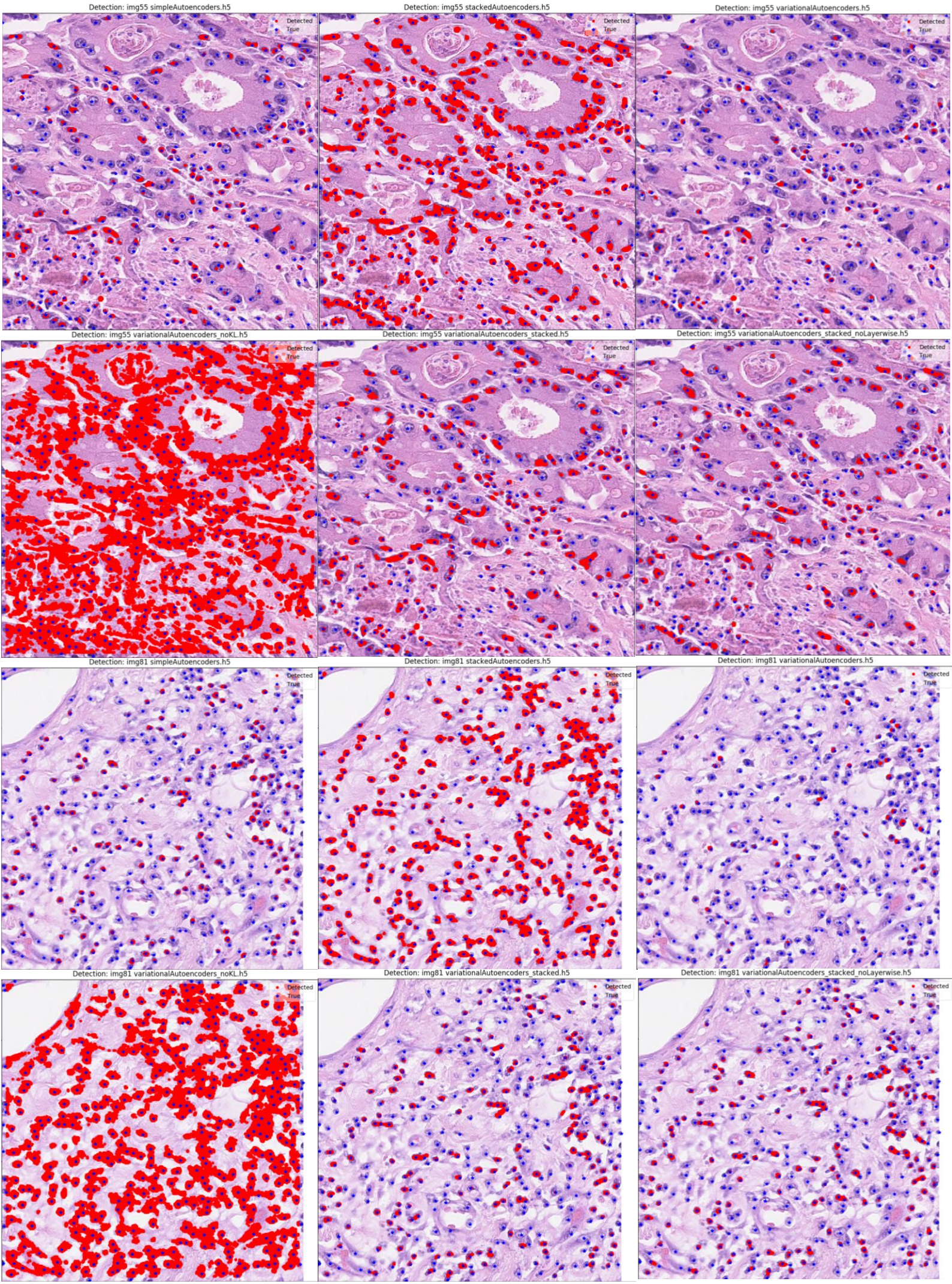

Detection: img48 variationalAutoencoders.h5
Detected
True


Detection: img48 variationalAutoencoders_noKL.h5
Detected
True

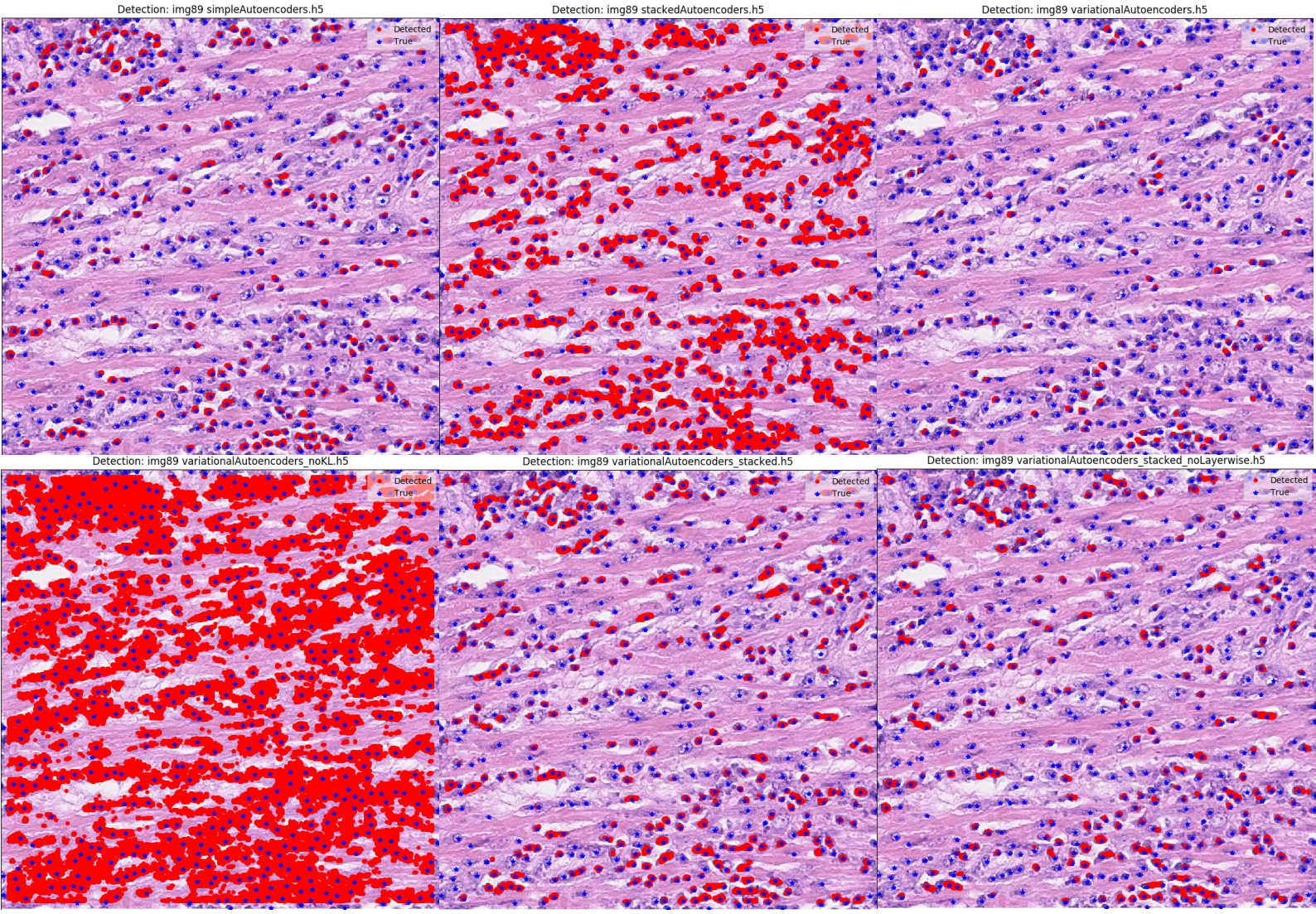

Detection: img48 variationalAutoencoders_stacked.h5
Detected
True


Detection: img48 variationalAutoencoders_stacked_noLayerwise.h5
Detected
True

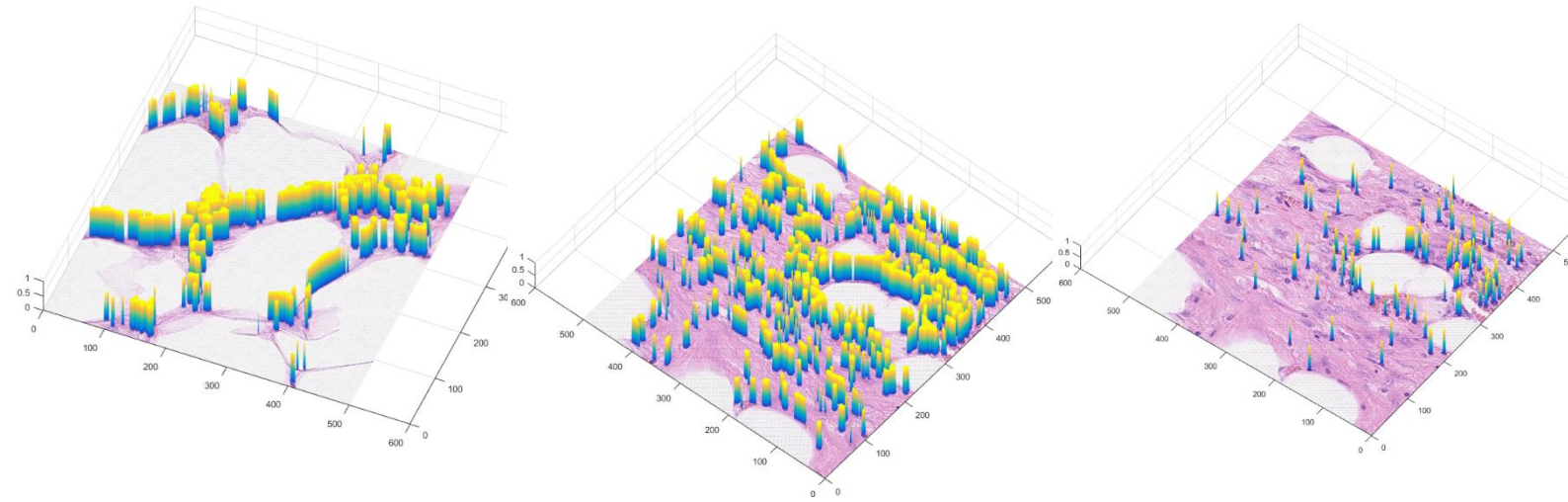
IMG4-IMG48


Detection: img55 simpleAutoencoders.h5
Detection: img55 stackedAutoencoders.h5
Detection: img55 variationalAutoencoders.h5
Detection: img55 variationalAutoencoders_noKL.h5
Detection: img55 variationalAutoencoders_stacked.h5
Detection: img55 variationalAutoencoders_stacked_noLayerwise.h5
Detection: img81 simpleAutoencoders.h5
Detection: img81 stackedAutoencoders.h5
Detection: img81 variationalAutoencoders.h5
Detection: img81 variationalAutoencoders_noKL.h5
Detection: img81 variationalAutoencoders_stacked.h5
Detection: img81 variationalAutoencoders_stacked_noLayerwise.h5
Detected
True

IMG55-IMG81



IMG 89



Raw data of vA_noKL and vAs in img2 and img48

3. Detection Precision:

To quantify the classifier's detection precision, we sampled 13400 nuclei 17*17*3 images and 13400 17*17*3 background images from the training and test sets. The precision is calculated by the formulation below. DP is detected positive nuclei samples, DN is detected negative background samples, and S is number of all samples (26800). Threshold is set with 0.8, 0.2 respectively.

$$\text{Precision} = \frac{DP + DN}{S} \quad :$$

	Training Detected Positive	Training Detected Negative	Test Detected Positive	Test Detected Negative	Training Precision	Test Precision
sA	12309	10899	12519	10801	86.59%	87.01%
stA	12204	11408	12558	11298	88.10%	89.01%
vA	12042	11712	12463	11545	88.63%	89.58%
vAs	11590	11688	11958	11523	86.86%	87.62%
vAs_noLW	11948	11457	12354	11326	87.32%	88.36%
vA_noKL	12186	12112	12356	11728	**90.66%**	**89.86%**

Table 1. Precision of different autoencoders.

Section VI Conclusion

Through the experiments above, we concluded two significant results. First, the new developed variational autoencoder (**vA**) can perform better than the traditional autoencoders. The variational autoencoders can also be stacked (**vAs**) for deep learning, but none like other neural networks, the **vAs** lacks of effective training methods to improve performance, and the layer-by layer greedy training will only help slightly. It performs even worse than **vA** and **stA**. In my opinion, the introduction of generative model, and sampling procedure may make the training more uncertain, and the stochastic gradient descent algorithm may trap in the local optima more easily. Therefore, it is very necessary to develop a new or improved training methods for such generative model based condition.

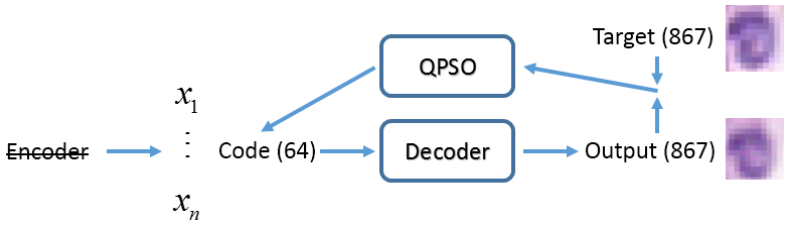
Second, the **vA** without KL divergence constrain outperforms extremely well compared with others. KL divergence is an essential part of the **vA**, which was added in the loss function to evaluate the divergence between latent distribution and normal distribution. I think this constrain may work well on the condition when input is Gaussian distribution, but failed contrarily. Therefore, it is very important to build a new adaptive model for the latent variables in practical environment.

In general, this experiment only reveals the performance and intractable problems of autoencoders, and proposed several possible reasons for it. Although it is proved that those algorithms can handle with detection tasks, but we still need plenty of expanded experiments to support our theory in the future, which involves great scientific challenges.

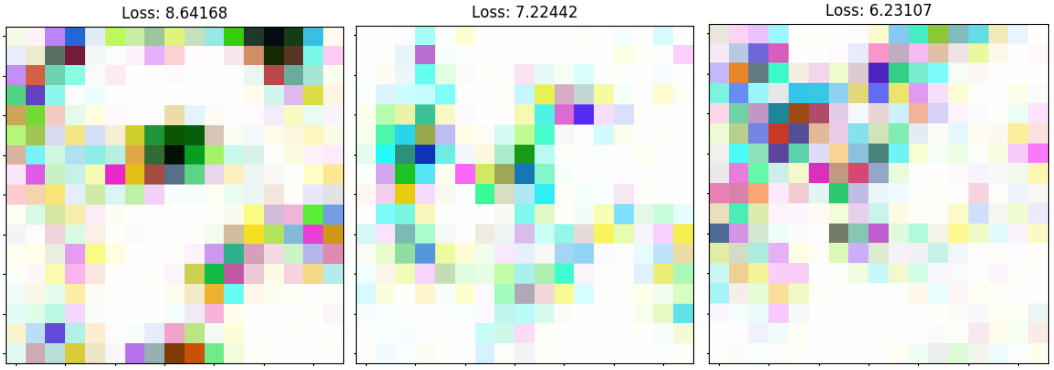
Section VII Digression

As we know, the autoencoders was created to reconstruct its input, and it has two main parts: encoder and decoder. The autoencoder will attempt to find an ANN that could best “encode” the input through training. But, let us think in an opposite way, how about finding a “code” that could best reconstruct the input through the determined decoder?

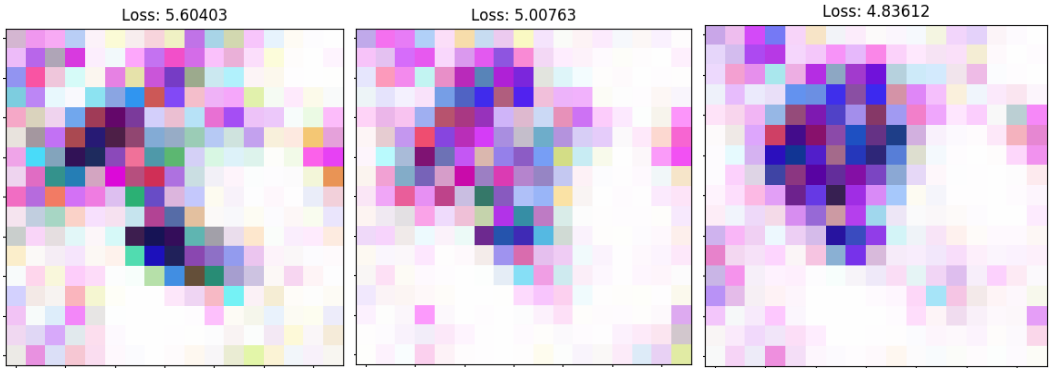
For instance, assume we have an encoder, with 867 dimension input, and 64 dimension in encoded space. Through the stochastic gradient descent training, the autoencoder’s weights will be tuned. But we only care about the discrepancy between input and output, and we rarely know much about the coded space! How is the “code” space organized? How would the changes in code affect the output? While, here, I did a simple but interesting experiment. Structure was shown below.



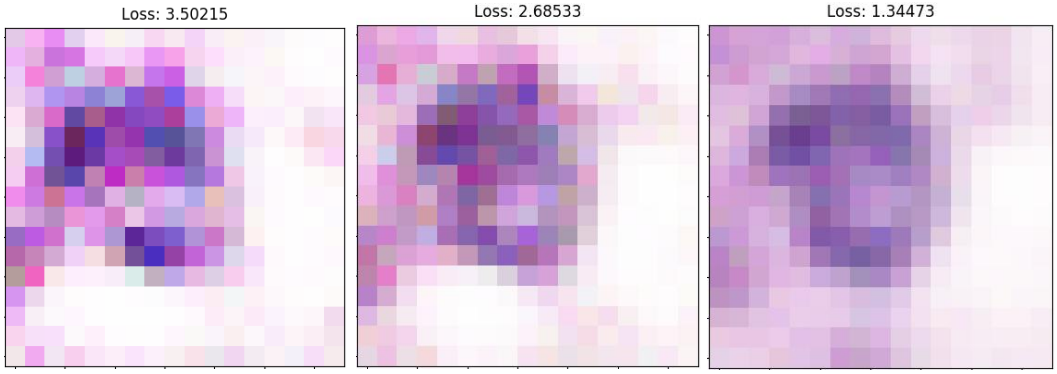
In the "code" space, which is 64 dimension here, we attempt to find a "code" solution that could best represent the output to the target, which is different from finding an ANN with same task. The optimization algorithm here we adopt is Quantum Particle Swarm Optimization algorithm (QPSO), with swarm population and beta set with 200 and 1~0.5 linear respectively, you can know more about this algorithm in my paper [2]. After 1000 iteration, the approximate solution was found.



Iteration 1, 6, 13

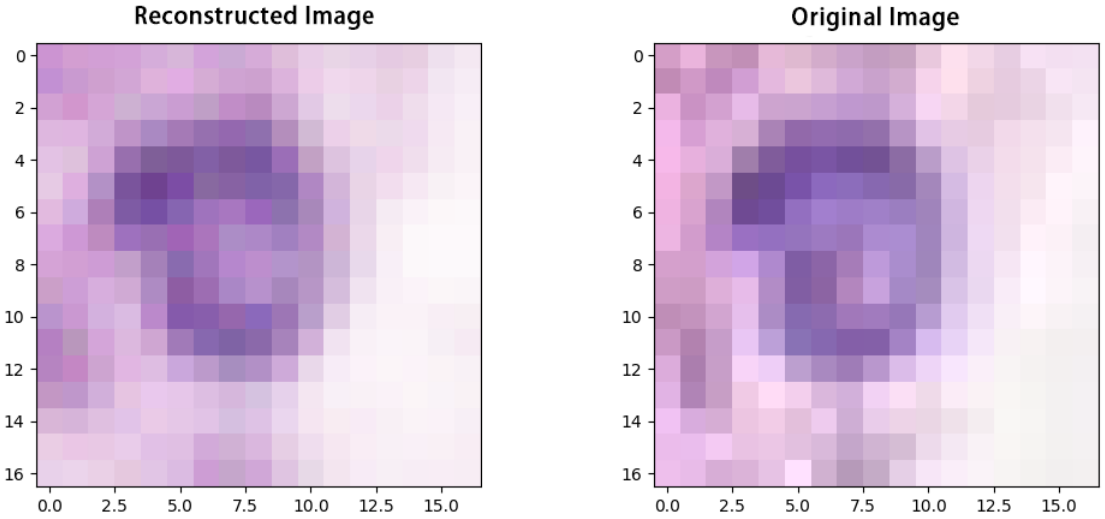


Iteration 20, 36, 62



Iteration 186, 359, 600

Finally, the image was reconstructed.



You can check the animation of this process on: https://www.youtube.com/watch?v=sYnfk186Vmc

You may ask, what is the meaning of finding a best qualified "code"? In my opinion, This is a very obvious generative procession, which creates data from conditional distribution, and there may be two contribution in the future. First, as we've discussed above, the stacked variational autoencoders lack of effective training method, so finding a potential "code" can be fused in the procession of the training to accelerate convergence, even help escape local optima. Because this algorithm could provide a prior solution to SGD. Second, the usage of autoencoder was usually restricted to the encoder part, and the decoder seems to be only useful in the training stage. Take the above detection experiment for example, the decoder was totally abandoned. But our experiment shows that the decoder may help with encoding something efficiently, they can work in similar but distinctive ways. In general, the exploration about this is only a start, I plan to expand the experiment and find more solid theory support.

Acknowledgement:
All of the source code can be downloaded from: https://github.com/zgbkdlm/vae_experiments/. Please follow the license.

References:

[1] K. Sirinukunwattana, S.E.A. Raza, Y.W Tsang, I.A. Cree, D.R.J. Snead, N.M. Rajpoot, 'Locality Sensitive Deep Learning for Detection and Classification of Nuclei in Routine Colon Cancer Histology Images,' *IEEE Transactions on Medical Imaging, 2016 (in press)*

[2] Naigong. Y. and Zheng. Z. An Improved Quantum-Behaved Particle Swarm Optimization Algorithm with Dynamic Chaos and Beta Distribution Strategy. *Proceedings of 2016 4th International Conference on Control Engineering & Information Technology (CEIT-2016)*. Tunisia, 2016