Integrative Imaging and Molecular Diagnostics Lab Project report

Leon H. Kloker
Institute for Computational and Mathematical Engineering
Stanford University
leonkl@stanford.edu

1 Brief overview of project

The goal of the project is two combine the information on cell morphology contained in H&E stained images with the biomarker data from Multiplexed Immunofluorescence (mxIF) images in order to classify the cells in the tumor microenvironment into more fine-grained classes. As the activation levels of biomarkers in nucleus and cytoplasm of a cell contain additional information on the functionality of the cell, we expect that the combination of these two image modalities will allow for a much more detailed classification of the cells and thus yield a more comprehensive picture of the tumor.

2 Coregistration of H&E and mxIF images

The first essential step in the project is two coregister each H&E core with the corresponding mxIF core. This needs to be done in order to determine which biomarker activation levels from the mxIF data correspond to which cell in the HE image. In its essence, the coregistration is done by finding an coordinate transformation consisting of linear scaling, shifting and rotation around a center, such that the transformed cell coordinates in the mxIF image align with the cells in the HE image.

Hossein has already fine-tuned parameters of OpenCV image transformations such that both images are aligned as close as possible. Based on the values of these parameters and the definition of the applied OpenCV transformations, I wrote a python script `coregistration.py` to automize the coregistration. This script takes the values of the fine-tuned transformation parameters and the original mxIF cell data and transforms all the cell coordinates contained in the original mxIF data to the coordinates in the H&E image. Hence, after the coregistration, the activation levels of the biomarkers contained in the mxIF data can be assigned to the corresponding cell in the H&E image.

Script: `coregistration.py`

- Necessary command line arguments
  - -tf <.csv file containing the fine-tuned transformation parameters>
  - -d <directory containing all the mxIF .csv files which should be coregistered>
- Optional command line arguments
  - -i if specified, a dot is drawn in the H&E core at each transformed mxIF cell coordinate and the resulting image is saved in directory `coregistered_cores`

Winter 2023, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

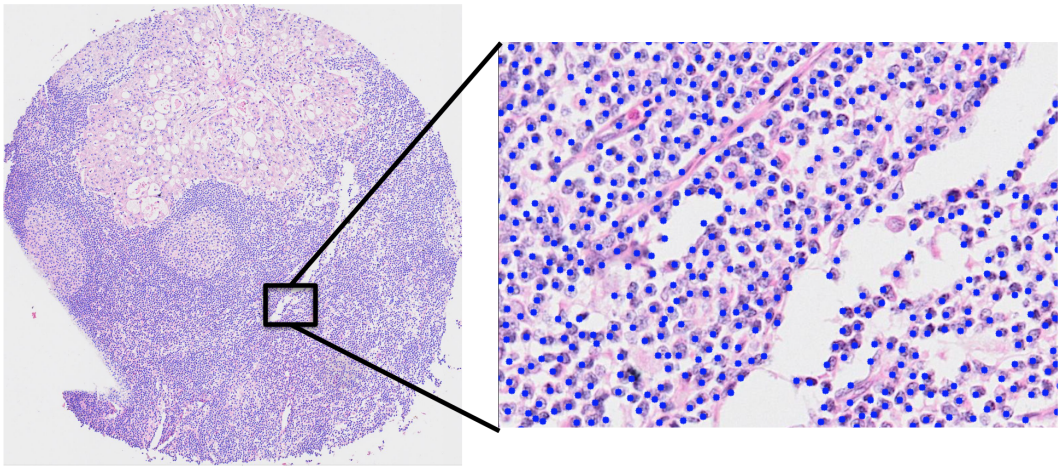

Figure 1: H&E image with blue dots drawn at transformed mxIF cell coordinates. We can see that the transformed cell coordinates align well with the cells visible in the H&E image.

-s if specified, the mxIF .csv files with updated cell coordinates are saved in `<original mxIF directory>_coregistered`

3 Autoencoder baseline model

In order to get a meaningful dense representation of each cell, we train an autoencoder on 64x64 H&E image patches centered at the transformed cell coordinates. The architecture for the baseline model is shown in figure 2. The same is done simultaneously to the mxIF data vector that contains mean, max and standard deviation values of the activations for biomarkers PD1, FOXP3, CD20, CD3 and PANCK in nucleus and cytoplasm. In order to do so, a feed forward autoencoder consisting of 6 dense layers with dimensions 64, 32, 16, 32, 64 and 30 (length of the mxIF data vector) with batch normalization in between each layer and a dropout rate of 0.5 is trained. Both, the convolutional autoencoder for the H&E cell images and the dense autoencoder for the mxIF cell data are trained for 1000 epochs on all cells in the cores H-15, A-15, B-11, A-7, F-9, G-3, G-11, H-7 and I-13, meaning the training set contains over 150000 cells.

As training takes roughly a week with such a big dataset, figure 5 shows the training progress after 120 epochs where both autoencoders are trained in parallel as one model. The dropout in the dense autoencoder prevents the model from overfitting as visible in the figure.

We can visually observe how well the autoencoder is able to reconstruct a cell image. Figure 4 shows the original cell image next to its reconstruction after being passed through the encoder and decoder.

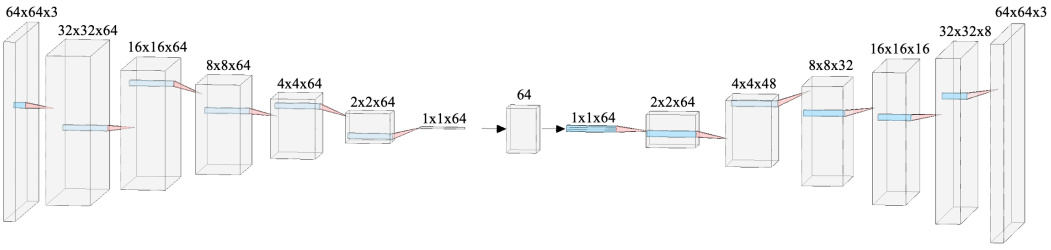

Figure 2: Architecture diagram of the convolutional autoencoder. Every normal and transpose convolutional layer has kernels of size 3, stride 2, zero-padding and a ReLU activation, except for the last layer of the decoder which has a sigmoid activation. Moreover, every convolutional layer is followed by a batchnorm layer except for the last layer of both encoder and decoder. The black arrows correspond to a dense layer and the reshaping of the data before and after the latent space.

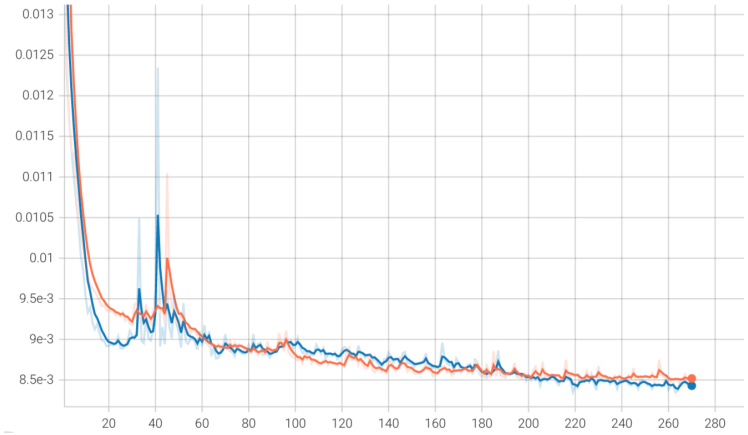

Figure 3: The figure shows the training process of the combined convolutional and dense autoencoder after 120 epochs. The orange curve is the train loss whereas the blue curve represents the validation loss and the loss function is mean squared error for both H&E image and mxIF data reconstruction.

4 Clustering

After well performing convolutional and dense autoencoders have been trained, we can use their encoder to get a latent representation of the cell image and mxIF data. As a baseline model, we just concatenate the two latent representations of the same cell to a 80-dimensional latent vector. When we do this for the entire training set, we obtain around 150000 latent vectors, which now can be clustered into K different clusters that hopefully correspond to different cell type and functionality. The clustering algorithm employed here is Kmeans and we use 10 different clusters. In the future, the amount of clusters can be changed to maximize the silhouette score of each cluster such that every cluster actually represents a meaningufl cell class.

5 Next steps

In order to get a more meaningful latent representation, we could also incorporate a cell segmentation mask into the H&E cell images, as as of now, the latent representation also depends a lot on the environment the cell is in as the 64x64 pixel patch often also contains neighbouring cells. In some cases, however, I might also be beneficial to have some of the immediate environment included for spatial context. Moreover, instead of only using two separate autoencoders, we will build an autoencoder that interweaves the mxIF and H&E data to a rich latent representation. This can be done by training another autoencoder on the concatenated latent representation of the separate encoders such that the autoencoder learns how much mxIF or H&E information is needed to properly reconstruct both the cell image and the mxIF values. This will also yield a better clustering, as the clustering with the current approach depends a lot on the ratio of latent dimensions of the H&E and

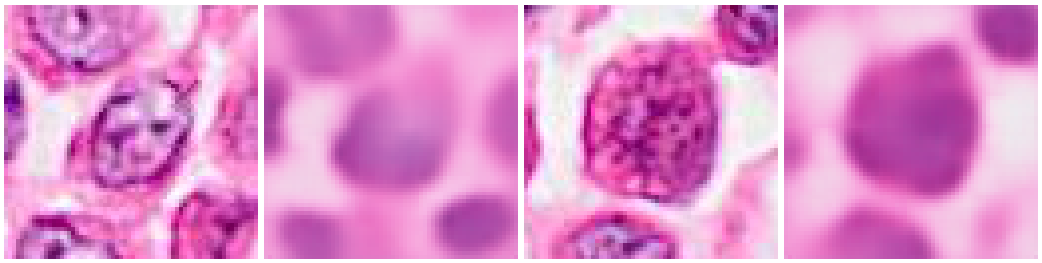

Figure 4: The first and third pictures are the original 64x64 H&E image of the cells and the second and fourth pictures are the corresponding reconstruction after the image is processed by the convolutional autoencoder. The model used for these pictures has only been trained for 120 epochs as the other model is still training as this is written.

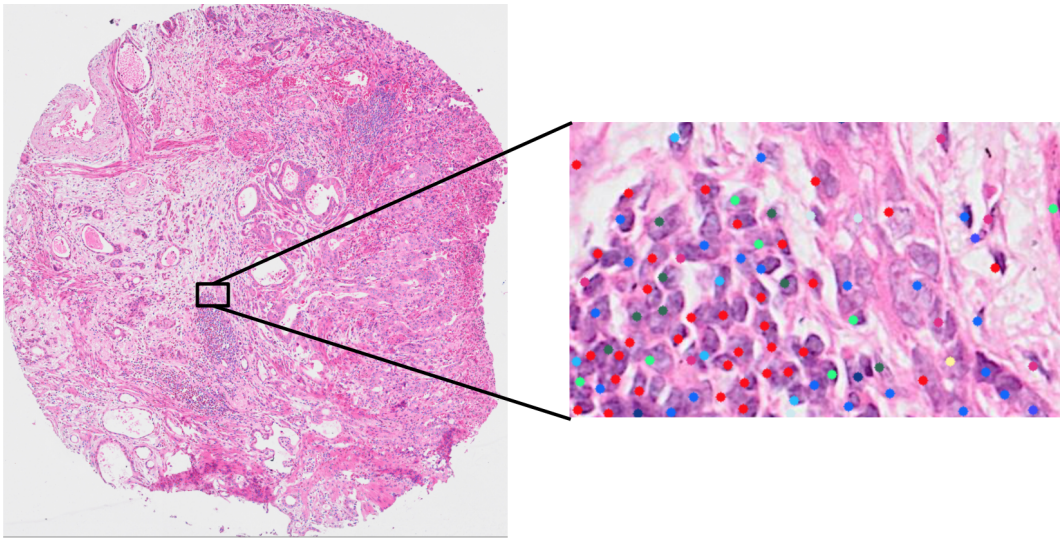

Figure 5: The figure shows a H&E core where every cell coordinate was colored according to the cluster it belongs to.

mxIF data. Finally, the amount of clusters to use can also be optimized by looking at the silhouette score of every cluster such that every cluster is well separated and corresponds to a meaningful cell class.