Utilizing Autoencoders for Analysis and Classification of Microscopic In Situ Hybridization Images

Authors: Aleksandar A. Yanev[1], Galina D. Momcheva[2], Stoyan P. Pavlov[3,4]

[1]High School of Mathematics "D-r Petar Beron", Varna, Bulgaria
[2]Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
[3]Department of Anatomy and Cell biology, Medical University of Varna, Varna, Bulgaria
[4]Advanced Computational Bioimaging, Research Institute of Medical University "Prof. Dr Paraskev Stoyanov" Varna, Bulgaria
E-mails: aleksandar.yanev.2004@gmail.com; (ORCID: 0009-0007-4051-1699)
gmomcheva@math.bas.bg (ORCID:0000-0003-0726-2022)
stoyan.pavlov@mu-varna.bg (ORCID:0000-0002-9322-2299)

Abstract: Currently, analysis of microscopic In Situ Hybridization (ISH) images is done manually by experts. Precise evaluation and classification of such microscopic images can ease experts' work and reveal further insights about the data. In this work, we propose a deep-learning workflow to detect and classify areas of microscopic images with similar levels of gene expression. Analysis of the data is done by employing a type of ANN – Deep Learning Autoencoders – suitable for unsupervised learning. The model's performance is optimised by balancing the latent layers' length and complexity and fine-tuning hyperparameters. The results are validated by adapting the mean-squared error (MSE) metric and comparison to expert's evaluation. Reconstruction of the whole-scale microscopic images is used to summarise and visualise the results.

Keywords: Artificial Neural Networks, Deep Learning Autoencoders, Image Analysis, Unsupervised Learning, Fuzzy Clustering

1. Introduction

In Situ Hybridization (ISH) is a method for the recognition and localization of specific nucleotide sequences in the nucleic acids (DNA and RNA) in cells and tissues [1]. Applications of this technique on microscopic images include the identification of viruses, diagnosis and grading of cancer, cytogenetics, and analysis of gene expression. In the chromogenic variant of the reaction (chromogenic in situ Hybridization – CISH), the Hybridization product is visualised by a coloured precipitate that can be easily observed, and documented using a bright-field microscopic imaging system [2]. Positive signals correspond to cells that actively produce (express) the investigated gene products[2], [3].

The CISH results are influenced by the tissue preparation, conditions of Hybridization and colour reaction development, and the imaging parameters. This substantial variability obstructs the standardised analysis of large batches of such data. Thus, the "gold standard" for gene expression grading in CISH-stained tissue slides is assessment by an expert, i.e. visual inspection of the slides at various scales and subjective grading of the strength, density, and/or distribution of the staining using more or less arbitrary ordinal scales for strength (e.g. "negative", "low", "moderate" and "strong"; or "-", "1+", "2+" and "3+"), and patterns of expression ("ubiquitous", "regional" or "scattered") [2], [4].

In this study, we attempt to develop a workflow for automated, fast, reliable, and reproducible analysis of CISH images. We explore the properties of the proposed approach: methods used for extracting meaningful data for the algorithm's training, length and complexity of the used autoencoder, limitations and prospects for future improvements.

2. Data Preparation

The lateral dimensions of microscopic whole-slide CISH images usually span tens of thousands of pixels. As our goal is to classify areas of the image with similar gene expression levels, we must train the autoencoder on smaller regions (tiles) cut from the complete image. The choice of tile size depends on the goal and scale of the evaluation, e.g., entire slice, brain area, subregional evaluation. In this study, we focus on the supracellular subregional level and thus have chosen square tiles with side 150 μm (on the scale of the original image – 0.5 μm/px, this corresponds to 300 px). We have chosen a 75 μm (150 px) overlap between adjacent tiles to ensure that the transition between them is well represented and to reduce the distortion of the results due to boundary phenomena.

The whole-slide microscopic images include large areas without tissue that introduce classification bias and increase the computational time and the memory footprint of the autoencoder. That is why we create binary tissue pixel masks (0 if the pixel is part of the background, 1 – if it belongs to the tissue) employing consecutive Gaussian blur, automatic triangle threshold, and morphological "Fill holes". For mask creation, we down-scaled 20 times the whole-slide images and apply Gaussian blur (α=10), followed by an automatic triangle threshold and morphological fill holes algorithm to remove small tissue defects. Low-intensity imaging artefacts located outside the tissue slice were excluded by morphological reconstruction of the mask from a seed created by a large-radius erosion (r≈300px). In the final stage, we resize the mask image to its original scale with a Lanczsos algorithm, extract the coordinates of the "tissue pixels", and define tiles for these coordinates.[5]

3. The Autoencoder

3.1. Encoder output size

The output size (the number of features extracted) is one of the deciding factors for the model performance and accuracy [6]. We have chosen to reduce the 300x300px tiles to just two floating-point numbers, which comes with its benefits and limitations.

The current best model completes its training, classifies all tiles, and reconstructs the classifications into colour-coded image maps in less than two hours. Another advantage is that with just two floating numbers, we can visualise thousands of tiles on 2D planes and search for meaningful correlations in the data. In this way, biology experts without a computer science background can easily look for new relations in the available data. On the other hand, the extracted features (the latent layer of the autoencoder) contain limited information that is not suitable for the reconstruction of meaningful images in the decoder segment of the network.

3.2. Evaluation of loss and choice of an optimiser

We perform an unsupervised classification with the freedom to vary the number of assigned classes. Testing with two of the most common functions for image analysis– MSE (mean-squared error) and MAE (mean absolute error), shows that the MSE usage enables the model to converge approximately 20% faster. Programmatic evaluations must be double checked due to imperfect mask generation. To demonstrate that this does not impede the correct classification of the tiles, we performed two tests — a real-world and a programmatic one.

The real-world test includes expert evaluation of the resulting whole CISH images and approval that the labels match the manual evaluation - areas of matching levels of gene expression belong to the same clusters appointed by the neural network.

The programmatic test used the preceding work of Pavlov, Atanasov, and Momcheva [7], [8], who extracted the exact coordinates for the two CISH images on which the autoencoder is trained. Evaluation of the autoencoder using the perfect dataset shows a convergence of the loss function at a

similar value, thus proving that the outlier background tiles do not change the result drastically. A comparison of the results reveals that after being trained on each data set, the classification outlines the same regions as having similar levels of gene expression. This test demonstrates the masking algorithm's ability to remove enough background data as well as the robustness of the autoencoder against small amounts of background data without bias in the results.

Loss measurement and the actual accuracy of the model are also closely related to the optimiser function. The properties of our image data, e.g. the presence of rare features, strongly point to the usage of adaptive optimisers. In choosing an optimiser function, we have considered benchmarks of optimisers [9] concerning our data. After testing, although with a difference of less than 0.02, Adam optimiser [10] outperforms RMSprop [11], Adagrad [12], and Adadelta [13].

3.3. Batch sizes and epochs

Batch sizes are usually a predefined parameter that sets the number of data points – in our case, tiles – needed before an update of the neural network's parameters. Larger batch sizes reduce computational time as parameters update less frequently, but when combined with the number of epochs, disbalance may lead to underfitting of the model. We have found no significant difference in convergence when batch sizes are between 30 and 60 combined with 70 epochs for the autoencoder training.

3.4. Layers

One of the main problems encountered was the inability of the model to extract meaningful features. The solution was the continued deepening of the model – 14 layer blocks in total (Fig. 1). An autoencoder with similar architecture but only 8 layers (encoder and decoder of 2 convolutional and 2 linear layers each) still manages to extract features that are neither as complex or well represented as needed. In conclusion, we chose a model with an encoder that chronologically uses 4 convolution and max pooling blocks with the ReLU activation function, followed by 3 linear blocks with LeakyReLU activation function. The decoder's architecture comprises 3 linear blocks with the LeakyReLU, followed by 3 transpose convolution blocks with ReLU and one with sigmoid activation function. The 'flatten' and the 'unflatten' functions represented in the figure change the data dimensions but are not considered in the estimation of the neural network depth. We discuss the purpose and properties of these blocks in the following paragraphs.

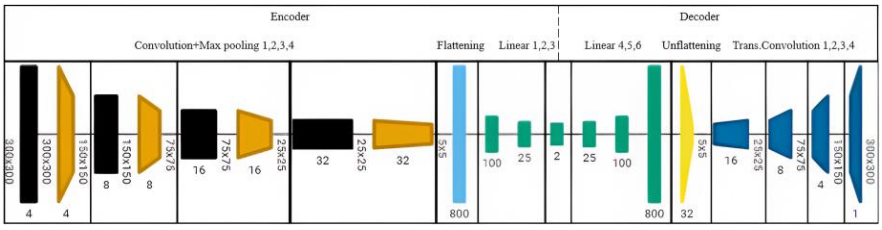

Figure 1. Scheme of the chosen autoencoder created with Graphiz [14]

3.4.1. Linear blocks

When using linear blocks, we encountered a problem with the effectiveness of the activation functions[15]. Initially, we used a compromise between ReLU and sigmoid functions that provided semicorrect results. However, after additional testing, we encountered the so-called "Dying ReLU" problem [16] related to a number of neurons never firing and thus acting as dead weight. Replacing the "normal" ReLU-s with Leaky ReLU versions seems to resolve the issue.

In the encoder, we use three linear layers that reduce the values extracted by the preceding convolutional blocks stepwise from 800 to just 2 – first from 800 to 100, next from 100 to 25, and finally from 25 to 2. Distributing the size reduction into three layers improves the quality of the

features, because each neuron's significance is calculated on multiple levels and therefore has greater weight in the final calculation.

The architecture of the decoder uses the same three linear layer blocks with reversed dimensions –2 to 25, 25 to 100, and 100 to 800.

3.4.2. Convolution and max pooling

Convolution [17] is a standard method used for image recognition and processing. Given our data, if the convolution kernels are too big, the most significant pixels for feature estimation may be disregarded, and a single high-value cell may influence the categorisation of an entire region. That is why for balance and to retain useful values even after convolution, we have decided to use four layers with smaller kernel sizes, each followed by a ReLU function (here we did not observe the mentioned problems with the standard ReLU activation) and a max pooling layer (Table 1).

Table 1. Convolution block layers

Block number	Convolution kernel size (in_channels, out_channels, kernel_size, padding=1)	Activation function	Max pooling layer size
1	(1,4,3)	ReLU	(2,2)
2	(4,8,3)	ReLU	(2,2)
3	(8,16,3)	ReLU	(3,3)
4	(16,32,3)	ReLU	(5,5)

The decoder's activation functions and the transpose convolutional layers (the reverse function of the convolution layers) use the same parameters as the encoder track. We must mention two technical differences: the absence of any form of pooling in the decoder and the presence of a sigmoid function instead of a ReLU in the last transpose convolution. This modification prevents significant jumps in output values that may disrupt the clustering we perform next.

4. Clustering of results

Our autoencoder provides two floating-point values that represent the input tile. The gene expression in our images is not clearly defined into separate classes – like in a "dog or cat" guesser. Therefore, to examine the relationships between different tiles we cluster them with a Fuzzy C-means algorithm [18], [19]. The clustering uses the following stepwise approach:

- Let N be the number of data points, with x_i denoting the vector representing the i-th point. Choose the number of clusters K. Let c_j be the vector representing the centre of cluster j. Choose a fuzziness parameter m (generally $1.25<m<2$). Randomly initialize a partition matrix U $[u_{i,j}]$ such that $\sum_{k=1}^{K} u_{i,k} = 1$ where $u_{i,j}$ is the membership value of data point x_i in cluster j.
- Then repeatedly do the following:
  1. Calculate each cluster centre from (2) (centroids)

(2) $$c_j = \frac{\sum_{i=1}^{N} u_{i,j}^{m} * x_i}{\sum_{i=1}^{N} u_{i,j}^{m}}$$

  2. Update each $u_{i,j}$ from (3) (centroids)

(3) $$u_{i,j} = \frac{1}{\sum_{k=1}^{K} \frac{\|x_i - c_j\|^{\frac{2}{m-1}}}{\|x_i - c_k\|^{\frac{2}{m-1}}}}$$

  3. Stop the process at iteration t when $\|\mathrm{U}^{t} - \mathrm{U}^{t-1}\|$ stops changing significantly (a threshold <0.005 produces well-clustered results in our case) or when desired

Fig. 2 shows the number of centres and the value of the associated FPC (fuzzy partition coefficient; a metric for the model performance describing the data [20], [7], [8]). In our results, the FPC decreases with the number of clusters (Fig.2). This behaviour is expected as there are no apparent clusters in our data– the tiles form a cloud that demonstrates an almost continuous gradient in the calculated features (Fig.2).

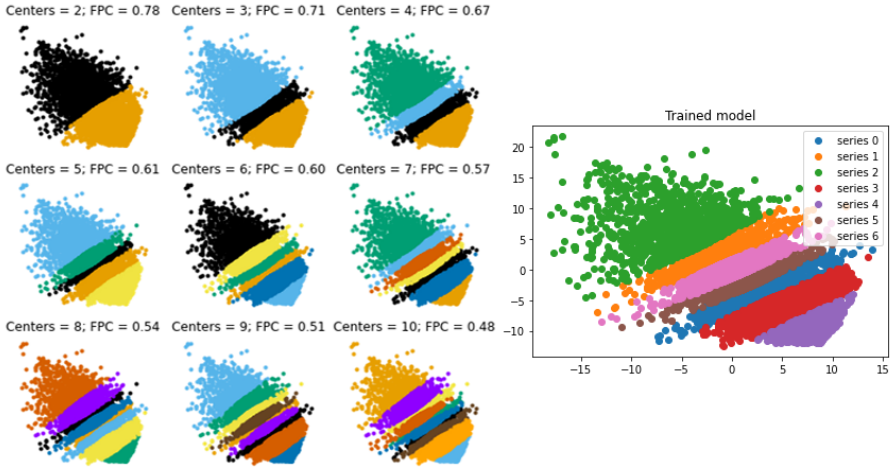

Figure 2. The distribution of tiles on the 2D plane and their clustering depending on the number of centroids

5. Analysis of the results

Despite the lack of obvious clusters, we chose to examine the 7-cluster model (Fig. 2, left). This number is close to the number of labels used in the real-life evaluation of CISH images. Additionally, as the number of groups increases, it becomes much harder to classify the tiles correctly by hand. To facilitate visual inspection, we reconstructed spatial colour maps of the clustering by mapping the colour-coded by class tiles to their coordinates in the original microscopic image. (Fig. 3)

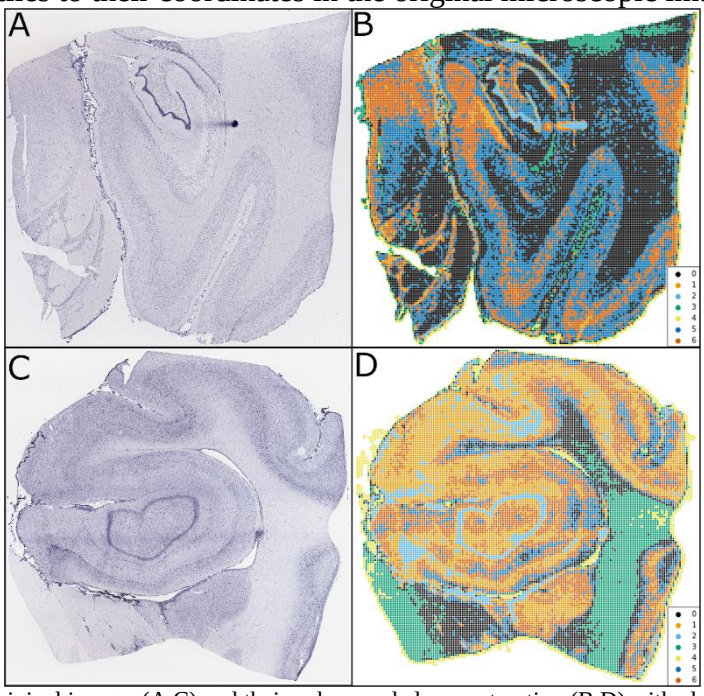

Figure 3. Original images (A,C) and their colour-coded reconstruction (B,D) with classified tiles

The apparent linearity in the two-dimensional feature space transfers into the well-visible correspondence between the colour-coded classes and regions with particular gene expression

patterns in the unprocessed images (Fig. 3). The reconstructions outline and demonstrate that the algorithm classifies regions with similar staining properties in the two images into the same class.

6. Future plans and prospects

This paper introduces the concept that the latent layer of Autoencoders can be used as a feature space for the unsupervised classification of staining patterns in microscopic CISH images.

The team is currently working on design of a user-friendly interface for practical application, and strategies for the deployment of the training algorithm and the trained network. The masking algorithm may be improved by implementing a more complex approach to reduce the "empty" tiles, thus reducing computational time. Also, we plan to develop an interface for: (i) the definition of either random selection of training tiles or predefined coordinates of a region of interest for analysis; (ii) the definition of scale of the analysis, i.e., size of the processed tiles; (iii) manipulation of the size of the latent layer and the number of extracted features.

As the expert evaluation of other types of microscopic images follows very similar logic, it will be fascinating to experiment how the model will behave when applied to images of different microscopic and macroscopic modality.

As the real-world data which we are using is usually noisy and of low contrast, experimenting with optical density instead of pixel values or employing a combination of median filtration and contrast enhancement may further enhance the model’s performance.

Finally, a feature-rich latent layer from stable encoder trained on real-world data may be combined with a model for the random generation of features and included in a GAN for generation of augmented data that can be used in different scenarios for training and validation of supervised ML algorithms.

R e f e r e n c e s

[1] Jensen, E., “Technical Review: *In Situ* Hybridization: AR Insights,” *The Anatomical Record*, vol. 297, no. 8, pp. 1349–1353, Aug. 2014, doi: 10.1002/ar.22944.

[2] Chu, Y.-H., H. Hardin, R. Zhang, Z. Guo, and R. V. Lloyd, “In situ hybridization: Introduction to techniques, applications and pitfalls in the performance and interpretation of assays,” *Seminars in Diagnostic Pathology*, vol. 36, no. 5, pp. 336–341, Sep. 2019, doi: 10.1053/j.semdp.2019.06.004.

[3] Eichele, G. and G. Diez-Roux, “High-throughput analysis of gene expression on tissue sections by in situ hybridization,” *Methods*, vol. 53, no. 4, pp. 417–423, Apr. 2011, doi: 10.1016/j.ymeth.2010.12.020.

[4] Visel, A., C. Thaller, and G. Eichele, “GenePaint.org: an atlas of gene expression patterns in the mouse embryo,” *Nucleic Acids Res*, vol. 32, no. suppl_1, pp. D552–D556, Jan. 2004, doi: 10.1093/nar/gkh029.

[5] Walt, S. van der, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, *et al.*, “scikit-image: image processing in Python,” *PeerJ*, vol. 2, p. e453, Jun. 2014, doi: 10.7717/peerj.453.

[6] Venkatesh, B. and J. Anuradha, “A Review of Feature Selection and Its Methods,” *Cybernetics and Information Technologies*, vol. 19, no. 1, pp. 3–26, Mar. 2019, doi: 10.2478/cait-2019-0001.

[7] Pavlov, S., G. Momcheva, P. Burlakova, S. Atanasov, D. Stoyanov, *et al.*, “Feasibility of Haralick’s Texture Features for the Classification of Chromogenic In-situ Hybridization Images,” in *2020 International Conference on Biomedical Innovations and Applications (BIA)*, Varna, Bulgaria, Sep. 2020, pp. 65–68. doi: 10.1109/BIA50171.2020.9244282.

[8] Pavlov, S., G. Momcheva, P. Burlakova, S. Atanasov, D. Stoyanov, *et al.*, “Gabor Features for the Classification and Evaluation of Chromogenic In-Situ Hybridization Images [accepted for publication],” in *Contemporary Methods in Bioinformatics and Biomedicine and Their Applications*, vol. 374, Cham: Springer, 2020, pp. 375–383. doi: 10.1007/978-3-030-96638-6_39.

[9] Schmidt, R. M., F. Schneider, and P. Hennig, “Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers,” 2020, doi: 10.48550/ARXIV.2007.01547.

[10] Kingma, D. P. and J. Ba, “Adam: A Method for Stochastic Optimization,” 2014, doi: 10.48550/ARXIV.1412.6980.

[11] Tieleman, T. and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[12] Duchi, J., E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2121–2159, Jul. 2011.

[13] Zeiler, M. D., “ADADELTA: An Adaptive Learning Rate Method,” 2012, doi: 10.48550/ARXIV.1212.5701.

[14] Ellson, J., E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, “Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools,” in *Graph Drawing Software*, M. Jünger and P. Mutzel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 127–148. doi: 10.1007/978-3-642-18638-7_6.

[15] Nwankpa, C., W. Ijomah, A. Gachagan, and S. Marshall, “Activation Functions: Comparison of trends in Practice and Research for Deep Learning,” 2018, doi: 10.48550/ARXIV.1811.03378.

[16] Lu, L., Y. Shin, Y. Su, and G. E. Karniadakis, “Dying ReLU and Initialization: Theory and Numerical Examples,” 2019, doi: 10.48550/ARXIV.1903.06733.

[17] Dumoulin, V. and F. Visin, “A guide to convolution arithmetic for deep learning,” 2016, doi: 10.48550/ARXIV.1603.07285.
[18] Bezdek, J. C., R. Ehrlich, and W. Full, “FCM: The fuzzy c-means clustering algorithm,” *Computers & Geosciences*, vol. 10, no. 2–3, pp. 191–203, Jan. 1984, doi: 10.1016/0098-3004(84)90020-7.
[19] Warner, J., J. Sexauer, Scikit-Fuzzy, Twmeggs, Alexsavio, *et al.*, “JDWarner/scikit-fuzzy: Scikit-Fuzzy version 0.4.2.” Zenodo, Nov. 14, 2019. doi: 10.5281/ZENODO.802396.
[20] Kim, D.-W., K. H. Lee, and D. Lee, “Fuzzy cluster validation index based on inter-cluster proximity,” *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2561–2574, Nov. 2003, doi: 10.1016/S0167-8655(03)00101-6.

Author addresses:
Stoyan P. Pavlov (Email address: stoyan.pavlov@mu-varna.bg; *ORCID:*0000-0002-9322-2299)
Department of Anatomy and Cell biology
Medical University Varna
Prof. Marin Drinov Str. 55
9002 Varna
Bulgaria

Aleksandar A. Yanev (Email address: aleksandar.yanev.2004@gmail.com; *ORCID*: 0009-0007-4051-1699)
High School of Mathematics D-r Petar Beron Varna
bul. Aleksander Stamboliyski
9000 Varna
Bulgaria

Galina D. Momcheva (Email address: gmomcheva@math.bas.bg; *ORCID:* 0000-0003-0726-2022)
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
Acad. Georgi Bonchev Str., Block 8
1113 Sofia
Bulgaria