

Deep Neural-Gas Clustering for Instance Segmentation across Imaging Experiments

Philipp Grüning

Institute for Neuro- and Bioinformatics

University of Lübeck

Lübeck, Germany

0000-0003-2946-4020

Amir Madany Mamlouk

Institute for Neuro- and Bioinformatics

University of Lübeck

Lübeck, Germany

0000-0001-9709-1620

Abstract—CNNs are characterized in particular by the ability to independently learn suitable features from a given data set. However, the resulting latent space is optimized for the given training data. Especially for tasks that require a high generalization ability, like e.g. the segmentation of single cells in a microscopic image across various experiments, these specific solutions might not offer optimal results. In this work, we improve generalization with an additional unsupervised training step that operates in the latent space. First experiments with the Kaggle cell segmentation competition data show a strong improvement in the generalization of acquired knowledge when using a soft- and hard-competitive Neural-Gas algorithm for deep clustering with a standard CNN architecture.

Index Terms—semi-supervised learning, clustering, deep learning, instance segmentation

I. INTRODUCTION

When a human subject is asked to label the individual cells on a microscopic image, this typically does not pose a very difficult task, even for cell types and magnifications the subject has never seen before. For machine learning, however, it is crucial to have extensive training data that reflect the data distribution as good as possible.

Established bioimage analysis tools (like e.g. CellProfiler [1] or Fiji [2]) often provide powerful and easy to use solutions to segment single cell images, but they also end up to be specialized onto the given cell types from the training data. Due to the recent advances of deep learning in the field of biomedical imaging [3], the question arised whether it might be even possible to develop segmentation methods that can reliably process completely unknown images, regardless of cell types, staining, and magnification.

As a result, a number of competitions have recently emerged that are aimed precisely at this problem: one example was the MICCAI 2018 challenge [4] on multi-organ nucleus segmentation (7 organs, 21 623 annotated individual nuclei), another example was the Kaggle 2018 Data Science Bowl [5], a challenge to segment nuclei across different imaging experiments (30 experiments, 37 333 annotated nuclei). The motivation behind these initiatives is quite clear: Only without any human interaction will future microscopes be able to analyse cell tissues in a large scale at full machine speed.

A. Instance Segmentation

Instance segmentation can be seen as an extension to semantic segmentation, which is the pixel-wise classification of an image: instead of assigning a class to each pixel, we want to assign a value that indicates to which instance the pixel belongs. This segmentation task cannot be solved by a simple classification framework: the number of instances can vary for each input image and furthermore, there is no fixed order which index is assigned to which instance. The two most prevalent approaches to produce instance segmentation masks with CNNs [6] are detection with subsequent segmentation, as employed in the mask-RCNN [7]–[9], and semantic segmentation with post-processing [10], [11]. Like many other contributors, the winner of the Kaggle 2018 Data Science Bowl used an u-net [12] structure paired with a very deep classification network as backbone. Additionally, elaborated post-processing algorithms and ensembles were employed [13]. Therefore we decided to also use an u-net architecture for the experiments proposed in this work.

B. Unsupervised and Semi-Supervised learning, and Clustering

Although CNNs provide excellent results on a variety of computer vision tasks, the necessity for large training datasets is a downside of the approach. This poses a problem especially in the field of biomedical image segmentation: expert knowledge is often needed to produce reliable labels and most images contain a multitude of objects with intricate shapes, which in turn increases the time needed to accurately label one image. Hence, many biomedical projects cannot provide a desired number of labeled images. However, in many cases, unlabeled images exist in abundance. Accordingly, unsupervised or semi-supervised approaches that make use of this data gain more and more importance [14].

So called *self-supervision* [15]–[18] aims to generate models with high quality general purpose features by training the CNNs with auxiliary tasks, where labels can be easily computed. A typical task is for example to maximize the mutual information between images and their transformations [19] or between different features layers [20]. Learning features that model relations of image pairs well is a typical theme in self-supervision [21]–[24]. In so called *exemplar learning*, a

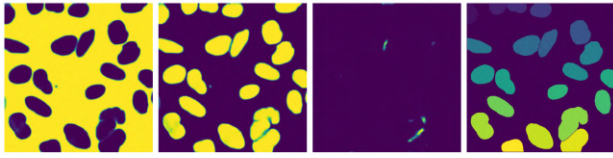


Fig. 1: **From semantic segmentation to instance segmentation.** Semantic segmentation output of the network: background (left), cell (middle left), cell border (middle right). Using a post-processing method based on the watershed algorithm, we can compute an instance segmentation mask (right).

training image is seen as a specific class. Other class members are transformed versions of this image and a network needs to learn features that are invariant to these transformations [25], [26]. Another approach is the generation of pseudo-labels, for example based on clustering in the feature space provided by a neural network. Caron et al. [27] proposed a semi-supervised learning framework including two steps: i) a CNN is trained on a labeled training set; ii) k -means clustering (k MC) is employed on the CNN’s feature representations of a set of unlabeled images. The cluster indices are used as pseudo-labels and subsequently, the CNN is trained fully supervised with the image-label pairs. Those two alternating steps are repeated several times. This method is called *deep clustering* and we based our proposed framework on it. Many recent works have identified unsupervised learning as a vital part for modern network architectures [28]–[33].

Although k MC is the first choice for many people when it comes to vector quantization, the algorithm comes with a major drawback: the random initialization paired with the *hard competitive* learning can lead to inhomogeneous distributions of the cluster centers, sometimes might even result in empty ones [27]. Martinetz and Schulten [34] proposed a *soft competitive* alternative called Neural-Gas (NG), where trivial solutions are not possible. Fritzke [35] extended the ideas of NG to the so called Growing Neural Gas (GNG), where no initial number of cluster centers k needs to be defined. Instead the algorithm automatically adds new cluster centers if necessary. In the following, we will use k MC, NG, and GNG to perform a deep clustering in the latent space of our CNN.

C. Contributions

We provide two contributions in this paper: first, we show that semi-supervised learning using deep clustering and pseudo-labels is viable, even in the context of instance segmentation. Although we only retrain the encoder part of an u-net, better generalization on unseen cell types and modalities can be achieved. Second, we investigate the impact of clustering quality, comparing k MC to NG and GNG. We show that although NG led to more robust solutions, k MC’s inhomogeneous clusterings might even be beneficial for this specific task.

II. METHODS AND EXPERIMENTS

A. Deep Clustering

In the context of instance segmentation, our aim is to enhance the generalization capability of a CNN for unlabeled images that greatly differ from the original training data. To this end, we train a model in two separate steps: the segmentation training and the cluster training.

In the segmentation step, we train a model for semantic segmentation, i.e. the pixel-wise classification of three classes: *background*, *cell*, and *cell-border*. Via a post-processing step, these segmentation masks can be transformed into instance segmentation masks (see Figure 1 for an illustration).

In the clustering step, we use a different, unlabeled dataset. For each image, we compute the latent space representation from the previously trained CNN to create a set of feature vectors. We reduce each feature vector’s dimension and then apply clustering. The given cluster index for each image denotes its pseudo-label and the encoder part of the CNN is trained to predict this label given the image as input. Note that this step only changes the encoder’s weights, while the decoder remains unchanged. Interestingly, even if the decoder is not adapted to the encoder, it can improve the instance segmentation results.

B. Data

For training and evaluation, we used the Kaggle Data Science Bowl 2018 training set from the first run, which is a subset of the Broad Bioimage Benchmark Collection dataset BBBC03v1 [5]. The set contains a high variety of 735 microscopic images of cell nuclei, that vary in cell type, image modality, resolution, and many more features. For example, cells from different organisms such as mice and flies are contained and different staining methods are used. Accordingly, the high variance of the respective images makes the dataset perfect for an analysis of generalization across domains.

We manually separated the data into 7 subsets of different types, that vary highly in their cardinality. An example image of each set is given in Figure 3. With 592 images, one set was very dominant, making up 80% of the total number of images. We called this set CT1 and the other 6 sets CT2 to CT7, respectively. We created a separate test set by randomly choosing two images from each cell type, amounting to 14 images for testing. This small number was due to some particularly small cell type sets that consisted of only seven or eight images. We used the training set of CT1 for segmentation training and solely in this step, the network did see actual ground truth labeling masks during training. For cluster training, we used the entire set CT2 to CT7. Since only pseudo-labels were employed, training with the test data is a viable approach.

C. Training Details

To enable mini batch training, we randomly cropped the input images to the size 256×256 (smaller images are zero padded first). Subsequently, we used the following data augmentation techniques with a probability of 50%: random

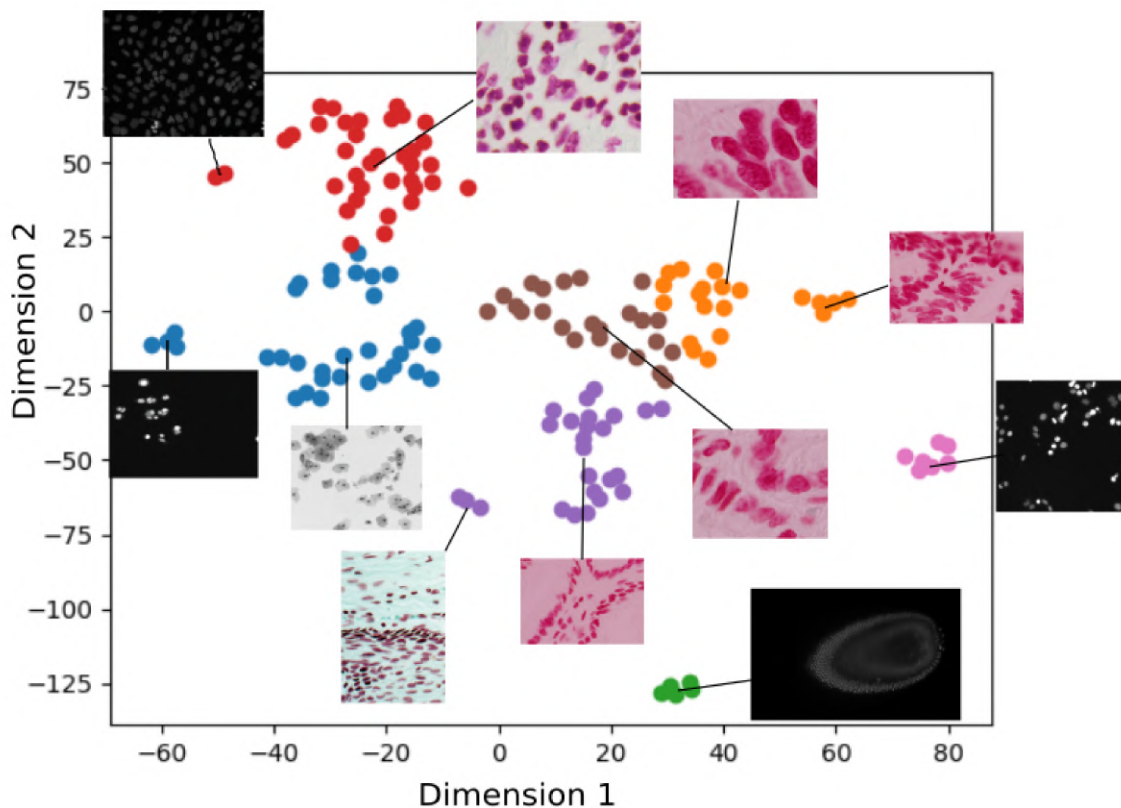


Fig. 2: **Latent space visualization.** T-SNE embedding of the 32 dimensional latent space. The computed clusters follow our intuitive division of the data based on image modality and cell type. However, further separation differences in cell size and image intensity are visible.

color channel swaps, color inversion, random scaling with a factor between .5 and 2 (25% probability), rotation of the image by a multiple of 90° , flipping along the vertical and/or horizontal axis.

We used an u-net-like structure with a resnet-50 encoder [36]. We defined the latent space to be the output of the last resnet block. For a $256 \times 256 \times 3$ input image, a $8 \times 8 \times 2048$ input tensor is produced.

For segmentation, we trained the CNN for 50 epochs, using the lamb optimizer [37], [38] with a learning rate of 0.001 beta parameters $\beta_0 = 0.9$ and $\beta_1 = 0.999$, and a batch-size of 8. We used cross entropy as our loss function.

For the proposed clustering extension, we drew 5000 images from the unlabeled dataset, with the described data augmentation regime. We applied global average pooling to the latent space output of the network. For each image, the $8 \times 8 \times 2048$ output tensor is reduced to a 2048 dimensional feature vector and the resulting data matrix of shape 5000×2048 is reduced to 5000×32 by employing principal component analysis (PCA). On this dataset, we used our different clustering algorithms and the resulting cluster indices were used as labels for the images.

In summary, we created a classification dataset containing

5000 images with k classes. The network architecture has been extended with a global average pooling layer and a linear layer that computes a k -dimensional output for a 2048 dimensional input. We trained the model for 20 epochs and from each 256×256 image we randomly cropped a 128×128 image patch. Apart from that, we used the same data augmentation functions as described above. We used a learning rate of 0.01, optimized with stochastic gradient descent (SGD) with a momentum of 0.5. Every 10 epochs, we reduced the learning rate by 0.1.

The input of the post processing algorithm is a softmax tensor of shape $h \times w \times 3$, with each pixel denoting the likelihood of belonging to one of the three classes *background*, *cell*, or *cell-border*. The instance labels were generated via a watershed segmentation. Starting points were pixels where the likelihood of being a cell and not a cell-border was greater than 80%. Additionally, background starting points were pixels with a value below 20%. The height map is a binarized version of the cell output map with a threshold of 30%. All instances computed by the watershed algorithm that had an average cell likelihood below 60% were set to background.

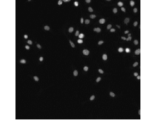
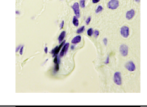
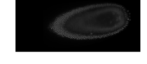
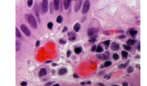
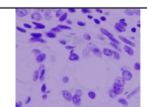
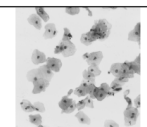
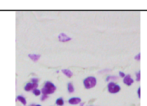
	cell type 1	592 images
	cell type 2	8 images
	cell type 3	7 images
	cell type 4	8 images
	cell type 5	66 images
	cell type 6	16 images
	cell type 7	38 images

Fig. 3: **Dataset partitioning.** We manually separated the Kaggle Data Science Bowl 2018 training set into seven individual subsets that differ in staining methods, cell type, cell nuclei, and image modality. We use cell type 1 (CT1) as training data for segmentation.

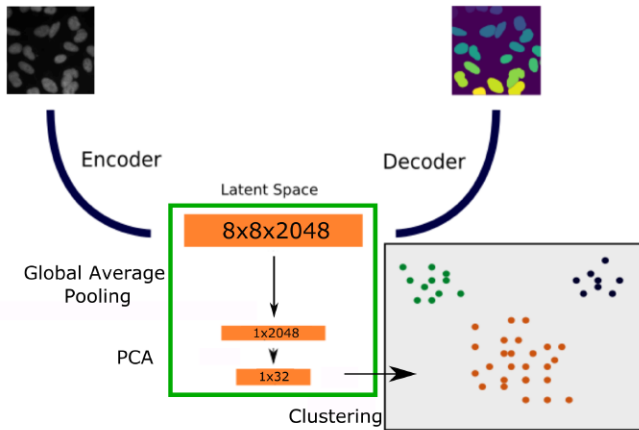


Fig. 4: **Deep Clustering with Segmentation.** Our architecture contains a typical encoder-decoder structure known from the u-net. At the end of the encoder (resnet-50 backbone), we transform a 256×256 input image to a 2048 dimensional feature vector. We reduce the dimension to 32 via PCA. We create pseudo-labels using clustering.

D. Clustering algorithms

We compare three different methods of clustering on the described latent space vectors: neural gas (NG), pattern-by-pattern k -means (k MC), and growing neural gas (GNG) [39]. The centroids for k MC and NG are adapted in a pattern-by-pattern fashion. In an epoch, a centroid adaptation is computed for each datapoint. Given a datapoint \vec{x} , the neural gas update rule for a centroid \vec{w}_i is defined as:

$$\Delta \vec{w}_i = \epsilon(t) h(\vec{x} - \vec{w}_i), \quad (1)$$

$$\vec{w}_i = \vec{w}_i + \Delta \vec{w}_i. \quad (2)$$

The update is weighted by $h \in [0, 1]$, which depends on the rank $r(\vec{w}_i, \vec{x})$. E.g. for the nearest centroid to \vec{x} , $r(\vec{w}_i, \vec{x}) = 0$, for the second nearest $r(\vec{w}_i, \vec{x}) = 1$ etc. h is computed as follows:

$$h(\vec{w}_i, \vec{x}, t) = \exp(-r(\vec{w}_i, \vec{x})/R(t)). \quad (3)$$

R is reduced depending on the current epoch t :

$$R(t) = R_0 \cdot \left(\frac{R_{fin}}{R_0}\right)^{\frac{t}{T}}, \quad (4)$$

with T being the total number of training epochs, with R_0 and R_{fin} defining the start and end value. ϵ is a learning rate parameter that is reduced similarly with respect to t :

$$\epsilon(t) = \epsilon_0 \cdot \left(\frac{\epsilon_{fin}}{\epsilon_0}\right)^{\frac{t}{T}}. \quad (5)$$

As hyperparameters, we use $\epsilon_0 = 0.1$, $\epsilon_{fin} = 0.0001$, $R_0 = 2 \cdot k$, $R_{fin} = 0.001$, $T = 200$ for NG. For k MC, we change $R_0 = R_{fin} = 0.001$. Note that, for each centroid with $r(\vec{w}_i, \vec{x}) > 0$, the corresponding weight h is zero. Hence for k MC, only the nearest centroid is updated (hard competitive). Opposed to that, NG can update several centroids (soft competitive). However, with increasing t , the weights for centroids that are not the nearest neighbor are reduced.

Different from NG, GNG has no parameters that change over time. Furthermore, there is no need to determine the number of cluster centers *a-priori*. Similar to NG, GNG is updated for every datapoint \vec{x} . Edges between pairs of cluster centers \vec{w}_i and \vec{w}_j indicate a neighborhood relationship between centers. The edges form or disappear in the course of training. For each input \vec{x} the nearest neighbor \vec{w}_{s1} is determined. It is updated similar to Equation 1, with $h = 1$ and $\epsilon(t) = \epsilon_b$. All neighbors to \vec{w}_{s1} are updated with $h = 1$ and $\epsilon(t) = \epsilon_n$. For each \vec{w}_i an error variable is kept that is increased every time \vec{w}_i is a nearest neighbor to \vec{x} :

$$error(i) = \|(\vec{w}_i - \vec{x})\|^2. \quad (6)$$

If the number of steps is a multiple of λ , the unit \vec{w}_q with the highest error value is determined. A new unit is created halfway between \vec{w}_q and its neighbor with the highest error value \vec{w}_f :

$$\vec{w}_r = 0.5(\vec{w}_f + \vec{w}_q). \quad (7)$$

For GNG, we use the hyperparameters $\epsilon_n = 0.1$ for the learning rate of the nearest neighbor, $\epsilon_b = 0.001$ is the learning

rate for centroids that are connected to the nearest neighbor via an edge. The maximum edge age is 50, every time after $\lambda = 100$ iterations a new centroid is added, the split error decay rate is 0.5, the error decay rate 0.995, and the minimal distance for updates is 0.2.

We trained one neural network in the segmentation step. Extracted a clustering dataset as described in Section II-C. For different numbers of cluster centers $k \in \{5, 50, 100, 500, 1000\}$, we created pseudo labels using k MC and NG. We computed the GNG clustering starting with $k = 5$. Here, the number of centroids increased to values around $k = 500$. For each k and each clustering method, we did 48 test runs altering the random seeds. This effectively changed the initial starting positions of each centroid and the order of datapoints in each epoch. We evaluated each run on the presented test datasets (see Section II-B). The *known* test set, contained images with a cell type that was also used for the segmentation data (CT1). The *unknown* test set contained only cell types that were not present in the segmentation dataset (CT2-7). We computed the adapted mean average precision (MAP, as presented in the Kaggle challenge [40]) of a ground truth instance segmentation and a prediction from our network.

III. RESULTS

In order to obtain a reference value, first we trained a network, which approaches the segmentation problem by means of a classical u-net [12]. For details on architecture and training please refer to Sections II-A and II-C. This approach can already solve the problem very well. Thus, most successful participants of the Kaggle Challenge have used very similar architectures. However, since we wanted to focus on changes caused by modifications in latent space, we have restricted ourselves on a basic network without any exhaustive post-processing steps.

We wanted to learn more about the complexity and structure of the latent space. Thus, we transformed the 32D feature space into a 2D projection using T-SNE [41] and looked for the input images to see how they were oriented in this space. Figure 2 shows a visualization of some images and their localization in latent space. Similar images actually seem to collapse into clusters and very unusual images are also clearly separated from the others.

Next, we systematically clustered the data in latent space using the three methods presented in Section II-D (k -means (k MC), Neural Gas (NG), and Growing Neural Gas (GNG)), retrained the encoder, and evaluated how the quality of instance segmentation had changed. Table I displays the MAP we obtained for the traditional approach without any clustering in the latent space (we will refer in the following to this value as the *baseline*). The MAP did not change significantly after clustering. As expected, the segmentation was quite difficult, especially with unknown images. Without CT1 (known images), the MAP was significantly lower than the MAP for the set CT1 only.

Nevertheless, there were significant differences when looking at the final segmentation. Figure 5 displays the results of

name	CT1	without CT1	all
baseline	0.41	0.21	0.23
k MC (5)	0.40 ± 0.03	0.19 ± 0.01	0.22 ± 0.01
NG (5)	0.40 ± 0.02	0.19 ± 0.01	0.22 ± 0.01
k MC (50)	0.41 ± 0.03	0.21 ± 0.01	0.23 ± 0.01
NG (50)	0.42 ± 0.02	0.21 ± 0.01	0.24 ± 0.01
k MC (100)	0.42 ± 0.03	0.22 ± 0.01	0.24 ± 0.01
NG (100)	0.43 ± 0.02	0.22 ± 0.01	0.25 ± 0.01
k MC (500)	0.42 ± 0.02	0.23 ± 0.01	0.26 ± 0.01
NG (500)	0.41 ± 0.02	0.23 ± 0.01	0.25 ± 0.01
k MC (1000)	0.41 ± 0.03	0.23 ± 0.01	0.26 ± 0.01
NG (1000)	0.41 ± 0.02	0.23 ± 0.01	0.25 ± 0.01
GNG (5)	0.42 ± 0.03	0.23 ± 0.01	0.25 ± 0.01

TABLE I: Mean average precision (MAP) on the test data set of models with different strategies for pseudo-labelling.

the instance segmentation for three different cell images. The first column shows the original images, the second column the corresponding ground truth. The third column gives the results with CNN without clustering, the fourth column provides the results with clustering (here k MC).

Next, we wanted to investigate the variance of the individual solutions and the influence of the quality on the free parameter k . For this purpose, we calculated the MAP and compared it to the baseline, both for CT1 and CT2-7. The results for CT1 are shown in Figure 6. Although the mean MAP always stayed close to the baseline, there were some configurations that got both better and worse. This applied to all three procedures. A completely different picture emerged for the development of the MAP for the unknown CT2-7 image series, however: as can be seen in Figure 7, the MAP after clustering in latent space gradually improved with increasing k . From $k = 500$ on, almost all runs provided an improvement to the baseline.

Finally, we wanted to examine the two methods k MC and NG for the different runs in terms of the consistency of the results. For this purpose we considered the the standard deviation of normalized entropy: the larger the standard deviation, the more diverse the solution spaces of the cluster method. In Figure 8, we can clearly see that the solutions for NG were more similar among each other for all k than for k MC.

IV. DISCUSSION

We have extended an established framework for segmenting cells with the aim of further improving the generalization capability of the architecture. For this purpose, we clustered the learned features of the latent space using unsupervised learning processes. Using the cluster indices as classification pseudo-labels, we retrained the encoder part of our segmentation CNN and then evaluated how the performance changes for different parameterizations.

The basic idea behind this has been that the learned features might be optimal for describing the training data, but especially for strongly deviating data, there may also be alternative features that are even better suited for generalization.

Following this idea, we expected to see a deterioration in segmentation performance in the first scenario rather than without vector quantization (VQ). However, it has been shown

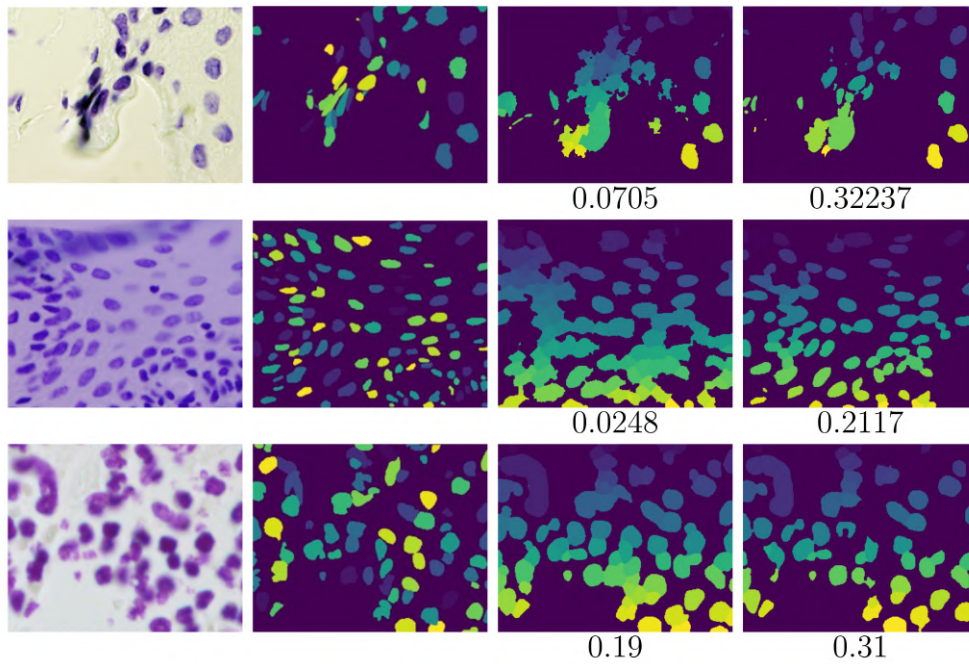


Fig. 5: **Result comparison.** Input image (left), corresponding ground truth label (middle left), network output before cluster training (middle right), network output after cluster training. Although only the network’s encoder is trained on classification with pseudo-labels, the overall segmentation quality can be improved in many cases.

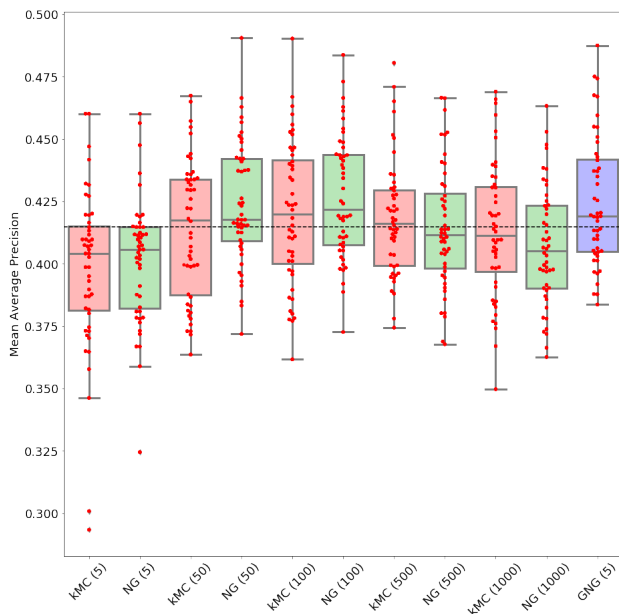


Fig. 6: **Test error on the known cell types.** For $k = 5$, the effect of the clustering randomly fluctuates around the ground truth (black dashed line). The NG solutions for $k = 50$ and $k = 100$ as well as the GNG solution show a slight trend towards improvement, although we would not have expected this for this scenario.

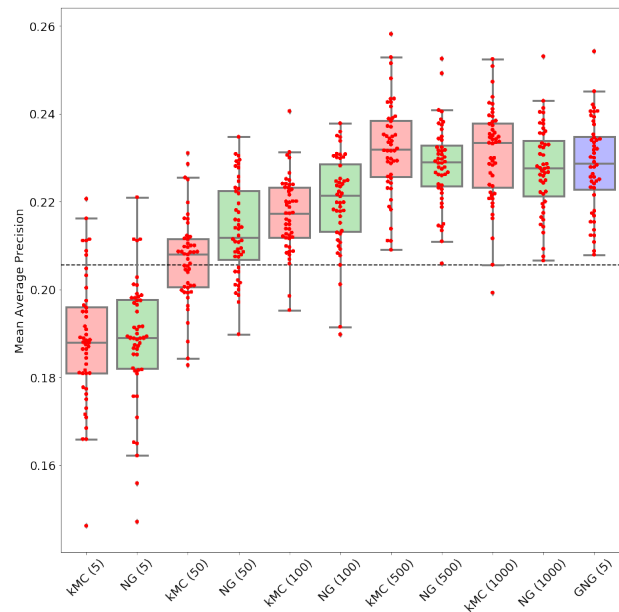


Fig. 7: **Test error on the unknown cell types.** The cluster solutions with $k = 5$ usually provide poor generalization performance for the unknown cell types. For $k \geq 50$, all approaches led to a systematic improvement in the generalization capability of the network.

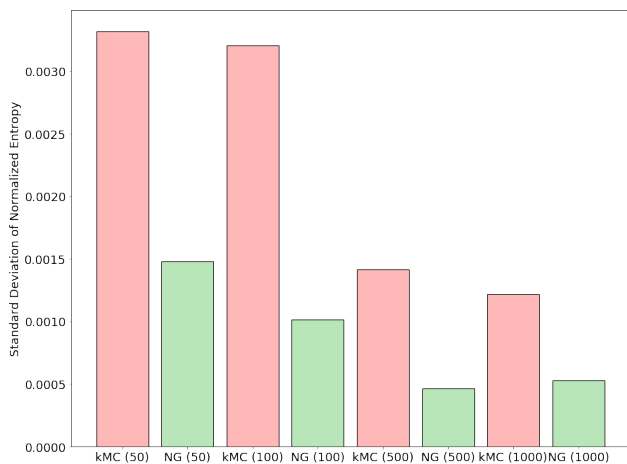


Fig. 8: Standard deviation of the normalized entropy. Comparison of the entropy variance based for the different cluster k and is therefore only comparable to a limited extent with the other approaches. For all configurations, NG shows much more stable solutions (smaller variance) than k MC. The variance for NG no longer decreases for $k = 1000$ compared to $k = 500$.

that, surprisingly, there can even be an improvement. For some configurations ($k = 50$), we observed a trend towards a better solution than before.

Interestingly, we could hardly see any difference between k -means clustering (k MC) and Neural-Gas (NG). Because NG behaves very robust and reproducible due to the initial large softness (large neighborhood radius) and ends hard-competitive in the late learning phases, the solution space of the NG is a real subset of the k -means solutions.

Nevertheless, the variance did not differ significantly for all k . This seems to indicate that the cluster centers are in stable configurations and the internal degrees of freedom influence the results. This also indicates that there might be only a few "real" clusters ($< k$) in the training data. Within a single data cluster, the codebook vectors are converging to a reproducible configuration, but at the same time, they can rotate freely within the given distribution. In such a scenario, this would apply equally to both VQ methods, k MC and NG.

For the second data scenario, however, we expected that there would be an improvement on the unknown cell images. Indeed, for all tested VQ approaches (k MC, NG, and GNG) our proposed method led to an increase of the MAP, thus increasing the generalization capability of the network. This result shows that clustering in latent space basically has exactly the intended effect: the network has achieved better generalization properties. Again, it was interesting to see that some of the k MC runs actually even outperform NG (see Figure 7). These differences can only be due to the fact that there are small outliers at the k MC that can no longer be placed in an optimal position. Such outliers could land - unknowingly - at those positions where later a large number of test images can land. Thus, a not-so-optimal configuration on the training data for k MC might lead to a better representation on unknown data that vary strongly from the given samples.

This assumption is also supported by the development of the entropy distribution respectively its variance (see Figure 8). Here, as expected, the NG is developing more robustly and with less variance than k MC for all the configurations examined.

In summary, it can be said that clustering in the latent space definitely has the potential to improve the generalization capability of a deep network. However, our experiments do not result in any clear recommendations regarding the three methods examined. It is a well-known fact that NG delivers a nearly optimal solution regarding the mean squared error on the training data [42]. But it seems that generalization might benefit from imperfect solutions. Therefore, an ensemble of k MC solutions should be examined in future work whether the latent space can possibly be better approximated by such an approach. Due to its design, GNG ends with a large parameter k and is therefore only comparable to a limited extent with the other approaches.

We showed that deep clustering might be a viable approach in the field of instance segmentation. Our proposed method already improved the generalization abilities on various biomedical images, even though so far the decoder remained untrained to the adapted features.

REFERENCES

- [1] C. McQuin, A. Goodman, V. Chernyshev, L. Kamensky, B. A. Cimini, K. W. Karhohs *et al.*, "Cellprofiler 3.0: Next-generation image processing for biology," *PLoS biology*, vol. 16, no. 7, 2018.
- [2] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nature methods*, vol. 9, no. 7, pp. 676–682, 2012.
- [3] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. Van Valen, "Deep learning for cellular image analysis," *Nature methods*, pp. 1–14, 2019.
- [4] N. Kumar, R. Verma, D. Anand, Y. Zhou, O. F. Onder, E. Tsougenis *et al.*, "A multi-organ nucleus segmentation challenge," *IEEE transactions on medical imaging*, 2019.
- [5] J. C. Caicedo, A. Goodman, K. W. Karhohs, B. A. Cimini, J. Ackerman, M. Haghghi *et al.*, "Nucleus segmentation across imaging experiments: the 2018 data science bowl," *Nature methods*, vol. 16, no. 12, pp. 1247–1253, 2019.
- [6] A. O. Vuola, S. U. Akram, and J. Kannala, "Mask-rcnn and u-net ensembled for nuclei segmentation," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019, pp. 208–212.
- [7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
- [8] H.-F. Tsai, J. Gajda, T. F. Sloan, A. Rares, and A. Q. Shen, "Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning," *SoftwareX*, vol. 9, pp. 230–237, 2019.
- [9] J. W. Johnson, "Adapting mask-rcnn for automatic nucleus segmentation," *arXiv preprint arXiv:1805.00500*, 2018.
- [10] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak *et al.*, "Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images," *Medical Image Analysis*, vol. 58, p. 101563, 2019.
- [11] R. Hollandi, A. Szkalicity, T. Toth, E. Tasnadi, C. Molnar, B. Mathe *et al.*, "A deep learning framework for nucleus segmentation using image style transfer," *bioRxiv*, p. 580605, 2019.
- [12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

- [13] S. Seferbekov. (2018) Kaggle data science bowl 2018 1st place solution. [Online]. Available: <https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741>
- [14] L. Yang, Y. Zhang, J. Chen, S. Zhang, and D. Z. Chen, "Suggestive annotation: A deep active learning framework for biomedical image segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 399–407.
- [15] O. J. Hénaff, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv preprint arXiv:1905.09272*, 2019.
- [16] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 5050–5060.
- [17] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2051–2060.
- [18] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision*. Springer, 2016, pp. 69–84.
- [19] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.
- [20] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler *et al.*, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.
- [21] V. R. de Sa, "Learning classification with unlabeled data," in *Advances in neural information processing systems*, 1994, pp. 112–119.
- [22] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.
- [23] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, "Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2547–2555.
- [24] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5879–5887.
- [25] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in neural information processing systems*, 2014, pp. 766–774.
- [26] M. A. Bautista, A. Sanakoyeu, E. Tikhoncheva, and B. Ommer, "Cliqecnn: Deep unsupervised exemplar learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3846–3854.
- [27] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, "Unsupervised pre-training of image features on non-curated data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2959–2968.
- [28] O. Siméoni, M. Budnik, Y. Avrithis, and G. Gravier, "Rethinking deep active learning: Using unlabeled data at model training," *arXiv preprint arXiv:1911.08177*, 2019.
- [29] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.
- [30] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.
- [31] A. Iscen, G. Toliás, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5070–5079.
- [32] X. Cao, B.-C. Chen, and S.-N. Lim, "Unsupervised deep metric learning via auxiliary rotation loss," *arXiv preprint arXiv:1911.07072*, 2019.
- [33] M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl, "Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 31–41.
- [34] T. Martinetz and K. Schulten, "A Neural-Gas Network Learns Topologies," *Artificial Neural Networks*, vol. I, pp. 397–402, 1991.
- [35] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in neural information processing systems*, 1995, pp. 625–632.
- [36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [37] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli *et al.*, "Large batch optimization for deep learning: Training bert in 76 minutes," in *International Conference on Learning Representations*, 2019.
- [38] B. Mann. (2020) pytorch-lamb. [Online]. Available: <https://github.com/cybertronai/pytorch-lamb>
- [39] Y. Shevchuk, "Neupy: neural networks in python," 2019.
- [40] K. Booz Allen Hamilton. (2018) Kaggle data science bowl 2018. [Online]. Available: <https://www.kaggle.com/c/data-science-bowl-2018/overview/evaluation>
- [41] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [42] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'neural-gas' network for vector quantization and its application to time-series prediction," *IEEE transactions on neural networks*, vol. 4, no. 4, pp. 558–569, 1993.