Clustering Plotted Data by Image Segmentation

Tarek Naous[1], Srinjay Sarkar[2], Abubakar Abid[3], James Zou[4]
[1] American University of Beirut, [2] VinAI Research, [3] Hugging Face, [4] Stanford University
tnn11@aub.edu.lb, v.sarkar@vinai.io, abubakar@hf.co, jamesz@stanford.edu

Abstract

Clustering is a popular approach to detecting patterns in unlabeled data. Existing clustering methods typically treat samples in a dataset as points in a metric space and compute distances to group together similar points. In this paper, we present a different way of clustering points in 2-dimensional space, inspired by how humans cluster data: by training neural networks to perform instance segmentation on plotted data. Our approach, Visual Clustering, has several advantages over traditional clustering algorithms: it is much faster than most existing clustering algorithms (making it suitable for very large datasets), it agrees strongly with human intuition for clusters, and it is by default hyperparameter free (although additional steps with hyperparameters can be introduced for more control of the algorithm). We describe the method and compare it to ten other clustering methods on synthetic data to illustrate its advantages and disadvantages. We then demonstrate how our approach can be extended to higher-dimensional data and illustrate its performance on real-world data. Our implementation of Visual Clustering is publicly available as a python package that can be installed and used on any dataset in a few lines of code[1]*. A demo on synthetic datasets is provided*[2].

1. Introduction

Numerous applications require the classification of unlabeled samples in a dataset into disjointed clusters such that samples within the same cluster are similar, yet samples in different clusters differ meaningfully. Many such clustering algorithms have been developed satisfying different desiderata for applications in fields such as image processing [1] [10], biomedicine [13], and spatial data [3].

The most commonly used clustering algorithms, such as K-means clustering [5], Gaussian mixture clustering [4], and DBSCAN [12], treat samples as points in a metric (usually Euclidean) space and group together points based on distances to other points or to computed exemplars. For example, the K-means clustering algorithm identifies optimal centroids in the metric space to which the distance of all samples in the dataset is minimized. The Gaussian mixture algorithm assumes the data is sampled from a mixture of Gaussians and produces clusters in the data to maximize the likelihood, which happens when data points are close to the centers of the Gaussian distributions. DBSCAN is [2] a density-based clustering algorithm that does not assume the number of clusters for the given dataset but considers a group of points belonging to the same cluster if there are a certain number of points in the neighborhood of the selected point. The clusters are expanded by recursively considering distances to all other points.

Since most of these algorithms involve measuring distances between points, they scale poorly to large datasets with millions or billions of samples. In our work, we introduce a completely different kind of clustering algorithm, designed for two-dimensional large datasets. Our approach, which we call *Visual Clustering*, is inspired by how humans cluster data: rather than computing distances, we segment data points into clusters based on the shape of large regions within the dataset. We simulate this process by training neural networks to perform instance segmentation on plotted data. Our approach has several advantages over traditional clustering algorithms: **(1)** Because the main step in the algorithm is running a prediction from one neural network, it is much faster than most existing clustering algorithms and scales easily to datasets with millions or billions of samples. **(2)** As we show on many kinds of datasets, it agrees strongly with human intuition for clusters, more so than many other clustering algorithms. **(3)** The core algorithm is hyperparameter free, although we suggest additional steps with hyperparameters that can be introduced for more control of the algorithm.

Clustering has been applied in the literature to solve various problems in deep learning and computer vision such as unsupervised image segmentation [14], facial landmark detection [9], and image grouping [1, 2]. However, no previous work has leveraged the fast inference time of trained neural network models to perform clustering. Developing

[1] https://github.com/tareknaous/visual-clustering
[2] https://huggingface.co/spaces/CVPR/visual-clustering

[1] In this paper, an adaptive clustering algorithm for image segmentation was proposed. The algorithm is designed to segment images into regions that have similar statistical characteristics based on the clustering of the pixels in the image. The authors used a Gaussian mixture model (GMM) to represent the pixel intensities in the image, and the clustering algorithm is based on an expectation-maximization (EM) algorithm.

[2] DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It is a popular clustering algorithm used in machine learning and data mining for identifying clusters of points in a dataset based on their spatial density.

Unlike K-means and Gaussian mixture clustering, which group data points based on their distance from the cluster centers, DBSCAN groups data points based on their local density. It identifies dense regions of points in the dataset and assigns them to clusters. Points that do not belong to any cluster are considered noise.

StatQuest Video: https://youtu.be/RDZUdRSDOok

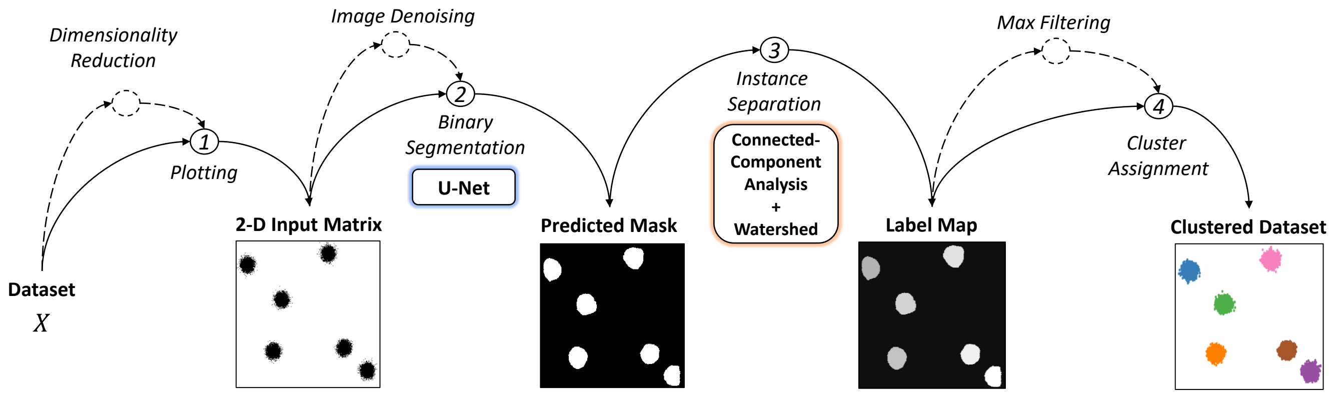

Figure 1. Diagrammatic view of our Visual Clustering algorithm. Dotted lines infer optional steps. The algorithm first creates a matrix representation of the dataset that is used as input to a binary segmentation model (U-Net). Connected component analysis and watershed are applied to the predicted binary mask to separate the different instances in the image, resulting in a label map from which cluster assignment is finally performed.

deep learning models that can replace classical algorithms has been studied in the literature for various problems such as sorting [15], solving mixed-integer problems [8], or even replacing index structures in data management systems [7].

Further, there are several deep learning-based approaches proposed for clustering [6]. However, these approaches rely on the optimization via gradient descent of two losses for any dataset that needs to be clustered, which necessitates a large computational time. In comparison, our approach requires simply one inference run (i.e. one "forward pass" through the model), which is significantly faster. Additionally, those methods are mostly tailored to the problem of image clustering. While transfer learning can be applied to extract latent features and by-pass initial optimization and training process in such approaches, this can only be done on data where a common correlation structure among features exists, such as image or text data. For general tabular data, such a correlation structure among features is unknown and varies across datasets. Hence, if those deep learning-based approaches were applied to such tabular datasets, they will still require a very large computation time. On the other hand, our proposed method is the first to provide a visual approach based on a deep learning-based segmentation model trained in a supervised fashion to perform clustering on numerical datasets (which could be tabular data or embeddings of data such as text or images), which has computational time advantages regardless of the type of data.

2. Methods

2.1. Core Algorithm

Our proposed Visual Clustering algorithm is illustrated in Figure 1 and consists of four main steps that we describe in this section. Consider the dataset $X \in \mathbb{R}^{m \times 2}$ that we would like to cluster. We start with the plotting step where X is represented in a two-dimensional matrix form, denoted by $I(X)$. This is done by first linearly shifting the values of both features to be $\in [0, 256]$ and then filling a zero-initialized 256×256 matrix by a value of 1 for each sample in the dataset according to its coordinates. In case the dataset was high-dimensional, we apply Principal Component Analysis (PCA) and use the first two principal components as features. The matrix, which can be visualized as an image, is then fed as an input to the second step of binary segmentation where a pre-trained binary segmentation model is used. We adopt the U-Net architecture [11] for binary segmentation and train it in a supervised fashion on images of plotted datasets, which were synthetically generated, and their binary masks. The U-Net model predicts a pixel-level binary mask $\hat{M}(I(X))$.

[3] Sample Code: https://pym.dev/p/33xg3/

The predicted binary mask by the trained U-Net model contains information on where cluster areas are located. However, the binary mask alone does not indicate how many clusters there are in the image. Hence, the next step in our approach is separating the instances (or clusters) present in the binary mask. To do this, we apply Connected Component Analysis on the predicted mask followed by a Watershed transformation for instance separation. This results in a label map L where pixels belonging to the same cluster are assigned the same label value. The final step in our approach is cluster assignment, where we assign a cluster label to each sample in the dataset based on its location in the label map.

[4] https://www.youtube.com/watch?v=ticZclUYy88

[5] https://en.wikipedia.org/wiki/Watershed_(image_processing)

2.2. Training the Binary Segmentation Model

To train the U-Net model for binary segmentation, we generated 1,000 synthetic datasets of blob-shaped clusters. Each numeric dataset is used to create a data sample to train the binary segmentation model where the input is the plotted dataset and the label is the segmentation map which where each cluster has its segment. To automatically generate the label (segmentation map) for each dataset, we computed the convex hull of each cluster in the dataset. The convex hulls are then used to form a binary mask label. When the hulls of two clusters were intersecting beyond a threshold of 30%, they were joined together to form one cluster. If the intersection was below the threshold, the hulls were subtracted to separate the clusters in the binary mask. The U-Net model achieved a test-set Intersection-Over-Union (IOU) of 88.7%. While only blob-shaped clusters were considered in the training process, the learned model can segment clusters independently from their shape as will be shown in the results of Section 3.

2.3. Handling Inseparable Clusters with Image Denoising

The developed binary segmentation model segments the image in a way that identifies disconnected clusters. If two or more clusters intersect, the model is likely to combine these clusters into one. This holds, especially in real-world datasets where clusters are more likely to be inseparable. To avoid this problem, we introduce an optional pre-processing step of image denoising to filter out low-density areas in the image and emphasize high-density areas. This helps disconnect cluster regions that seem to be connected by sparse data points. We specifically denoise the image through a median[6] filter, where increasing the size of the filter will eliminate low-density areas and emphasize high-density ones more severely.

2.4. Handling Unassigned Points with Max Filtering

When assigning labels to each point in the dataset in the final step of the algorithm, many points will fall in regions that are outside but near the cluster area. These may be points that deviate from where the majority of the points in the cluster are located and thus would be ignored by the binary segmentation model. Unassigned points may also be low-density areas that were filtered out in the image denoising step if used. To address this, we perform an optional maximum filtering operation on the label map to increase the area of each cluster, helping cover nearby unassigned points. The degree to which we would like to increase the cluster areas is controlled by the size of the max filter, where a larger filter size will result in larger cluster areas.

[7]
What if Max Filter size is increased too much?
Will it not start combining two clusters into one?

3. Results

3.1. Clustering Performance on Synthetic Datasets and Computation Time Comparison

The performance of our Visual Clustering approach is compared to multiple classical clustering algorithms on synthetic datasets of various cluster shapes in Figure 2. Although the segmentation model used in our algorithm was only trained on blob-shaped clusters, it can successfully segment clusters independently of their shape. On datasets that have more complex patterns, such as circle or moon-shaped clusters, our algorithm provides clustering results that are more in-line with human intuition compared to the results of K-means, Affinity Propagation, or Gaussian Mixture. While other algorithms such as DBSCAN or Spectral Clustering agree with human intuition, they suffer from a large computation time and cannot be used to cluster large datasets efficiently.

	Number of Samples					
Algorithm	10K	50K	100K	500K	1M	2M
Visual Clustering	0.292	0.571	0.909	3.686	7.222	14.096
K-Means	0.155	0.541	1.103	5.470	9.519	18.956
Affinity Propagation	175.35	∞	∞	∞	∞	∞
MeanShift	3.482	101.82	∞	∞	∞	∞
Spectral Clustering	0.052	0.509	0.796	7.455	20.001	53.559
Ward	1.994	27.965	93.564	∞	∞	∞
Agglomerative Clustering	1.177	12.154	39.886	∞	∞	∞
DBSCAN	0.093	0.357	0.837	7.604	20.009	52.648
Optics	16.515	∞	∞	∞	∞	∞
BIRCH	1.298	7.390	14.320	∞	∞	∞
Gaussian Mixture	0.089	0.358	0.726	3.047	5.949	11.962

Table 1. End-to-end computation time (in seconds) comparison of Visual Clustering with classical clustering algorithms for an increasing number of samples. ∞ indicates a computation time beyond 3 minutes and hence was not included. Visual Clustering is as fast as Gaussian Mixture and faster than all the other classical algorithms on large datasets.

The computation time of the segmentation model, instance separation algorithms, and filtering techniques are independent of the number of samples in the dataset. As our algorithm is a completely vision-based approach, it provides a great computation time advantage. However, the computation time of the plotting and cluster assignment steps in our algorithm increases linearly with the number of samples. In Table 1, we show an end-to-end computation time comparison between visual clustering (including the plotting time) and classical algorithms for an increasing number of samples. Visual Clustering achieves a very fast computation time that is almost identical to Gaussian Mixture, which was the fastest classical algorithm among the ones tested, and faster performance than K-Means clustering. Visual Clustering also outperforms all the rest of the classical algorithms in terms of computation time. Therefore, Visual Clustering achieves a compromise between slow classical algorithms like Affinity Propagation that can cluster complex patterns in a human intuitive manner while having a very fast computation time like Gaussian Mixture or K-Means.

3.2. Performance on Real-world Datasets

We evaluate Visual Clustering on three real-world datasets obtained from the UCI repository among which one

[6] A median filter is commonly used to remove noise from an image. It replaces each pixel value in the image with the median value of its neighboring pixels.

In this paper, they are using a median filter as a denoising step to filter out low-density areas in the image and emphasize high-density areas where the clusters are located.

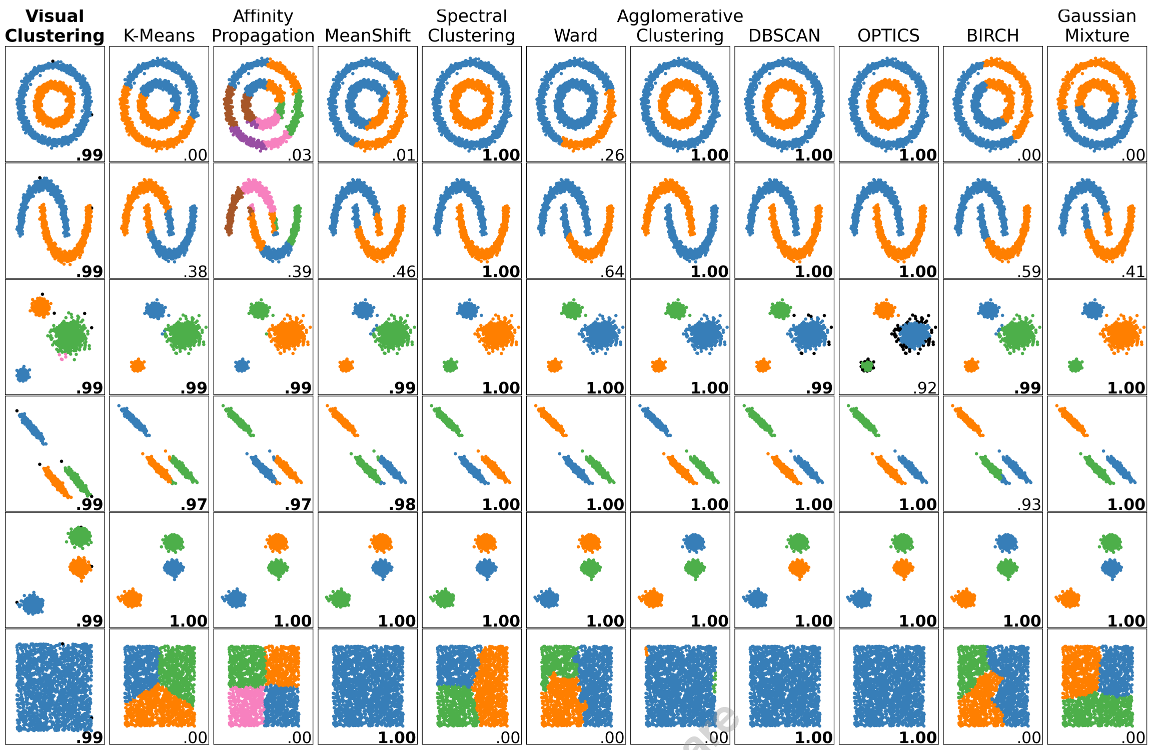

Figure 2. Comparison of Visual Clustering with classical clustering algorithms on synthetic datasets with several cluster shapes. The bottom right numbers indicate the Adjusted Mutual Information Score between the ground truth labeling and the predicted labeling. Scores that are above 0.95 are highlighted in bold. Visual Clustering and DBSCAN are the only two algorithms that achieve nearly perfect matches with ground-truth labeling on all synthetic datasets.

is two-dimensional and the two others are of higher dimension. The clustering results on those datasets are shown in Figure 3. In real-world datasets, clusters are more likely to not be visually separable. For the first two datasets (**a** and **b**), the image denoising step shows its effectiveness in emphasizing high-density regions in the dataset and eliminating low-density regions. This helps the binary segmentation model capture more clusters as they become visually separable. The majority of the points that belong to low-density regions were then recaptured by performing max filtering on the label map before cluster assignment. It is also noticed in those real-world datasets that outliers exist, which are a few sparse points that lie very far from the main clusters, and that are usually desired to not be considered as belonging to any cluster. While it is difficult for classical approaches to capture such outliers, it is observed that Visual Clustering performs well from this aspect and avoids forming clusters for such points, which offers an advantage when clustering is needed while trying to avoid outliers.

The third dataset (dataset **c**) presents a more challenging case for Visual Clustering where most of the points in the plot are connected through the same pattern. While visually it would be intuitive to identify the three main lines shown more clearly after denoising as three different clusters, the algorithm instead considers them as one cluster. This is because Visual Clustering relies on binary segmentation and connect component analysis which makes it difficult to identify several clusters on connected patterns, which could or could not be desirable output based on the domain expertise of users. In this respect, our future work will focus on improving the Visual Clustering algorithm to enable further flexibility in segmentation in a way that provides an ability in placing multiple clusters on connected patterns.

4. Conclusion

We introduced Visual Clustering, a fast clustering algorithm based on a trained image segmentation model. Visual Clustering is inspired by how humans cluster data: by plotting datasets in 2D and identifying groups of similar points. Our experiments on real and synthetic datasets and comparisons to ten classical clustering algorithms show that Visual Clustering achieves clustering results that are in-line with human intuition, which is highlighted by very high adjusted mutual information scores, and does so in a fast computation time that outperforms almost all the rest of the classical algorithms, making it efficient for usage on very large datasets.

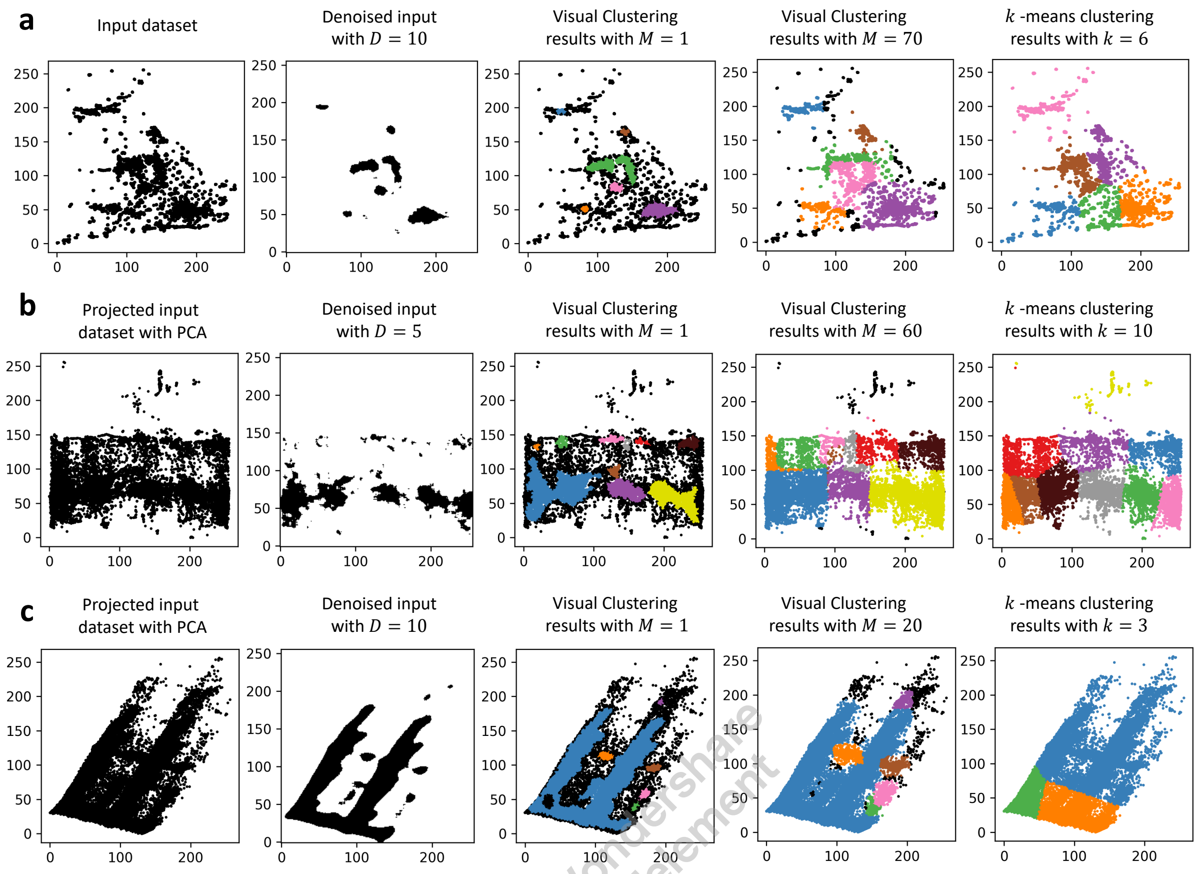

Figure 3. Results of the Visual Clustering algorithm on three real-world datasets compared with k-means clustering. D stands for the kernel size of the median filter. M stands for the kernel size of the max filter. Dataset **(a)** consists of urban road accidents coordinates. It contains 2 features and 360,177 samples. Dataset **(b)** consists of geo-magnetic field data for indoor localisation. It contains 13 features and 58,374 samples. Dataset **(c)** consists of individual household electric power consumption data. It contains 7 features and 2,075,259 samples. Both datasets **(b)** and **(c)** are reduced to two dimensions using PCA. We note that we cannot conclude which algorithm provides the best clustering since no ground truth labels are available in these datasets.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 1

[2] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5879–5887, 2017. 1

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996. 1

[4] Xin Jin and Jiawei Han. *Expectation Maximization Clustering*, pages 382–383. Springer US, Boston, MA, 2010. 1

[5] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 695–697. Springer US, Boston, MA, 2017. 1

[6] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, and Stefan Decker. Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*, 22(1):393–415, 2021. 2

[7] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, pages 489–504, 2018. 2

[8] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020. 2

[9] Xuan-Bac Nguyen, Duc Toan Bui, Chi Nhan Duong, Tien D Bui, and Khoa Luu. Clusformer: A transformer based clustering approach to unsupervised large-scale face and visual landmark recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10847–10856, 2021. 1

[10] Thrasyvoulos N Pappas and Nikil S Jayant. An adaptive clustering algorithm for image segmentation. In *International*

Conference on Acoustics, Speech, and Signal Processing,, pages 1667–1670. IEEE, 1989. 1

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[12] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017. 1

[13] Rui Xu and Donald Wunsch. Clustering algorithms in biomedical research: A review. *Biomedical Engineering, IEEE Reviews in*, 3:120 – 154, 02 2010. 1

[14] Lei Zhou and Weiyufeng Wei. DIC: deep image clustering for unsupervised image segmentation. *IEEE Access*, 8:34481–34491, 2020. 1

[15] Xiaoke Zhu, Taining Cheng, Qi Zhang, Ling Liu, Jing He, Shaowen Yao, and Wei Zhou. NN-sort: Neural network based data distribution-aware sorting. *arXiv preprint arXiv:1907.08817*, 2019. 2