1 Experimentation

This chapter will use the implementation of KNN-based graph-construction, label propagation and skeletonization to test how skeletonized label propagation compares to regular label propagation.

From early experiments it became apparent, that for skeletonization to add better accuracy the training data set would have to be very sparsely labeled. The results from the early experiments can be seen in appendix **??**, **??**, **??** and **??**, where the change in accuracy of the label propagation algorithm is marginal.

The label sparsity of the training data set is therefore narrowed to a range of $90\% \rightarrow 99.9\%$ removed labels. The narrowing is halted at 99.9% as a 99.99% removal of labels in the 60.000 sized training set would yield 6 nodes with labels, which does not cover all 10 unique labels of the MNIST data set.

This chapter will experiment with the different levels of label sparsity in the training data set, to compare the two models, and reflect on the results of the experiment.

1.1 Procedure of experiments

Both algorithms will use a graph created with knn-based graph construction. This graph is then skeletonized. The regular label propagation algorithm will use the original graph to propagate labels through, and the skeletonized label propagation algorithm will use the skeletonized graph.

The procedure for each algorithm is as follows:

1. Delete a percentage p of all labels in the train data set
2. Apply the algorithm to the training set, which will return the relabelled data points in the train set
3. Compare the true labels and the reconstructed labels. The clamped labels are not used in the comparison as these will always be equal.
4. Forget all labels in the test data set.
5. Use the reconstructed train data set to predict labels in the test set using the majority of labels in k nearest neighbors to determine each unknown label.
6. Compare the true labels with the predicted labels, that is, the accuracy of predicting the tested labels.

This approach outputs two different test metrics: reconstruction accuracy and testing accuracy. The reconstruction accuracy defines how well the labels within the graph have been assigned. Because the labels are embedded this metric might be skewed

as the `mde`-algorithm embeds nodes close together. To provide an independent metric, a second test is therefore made. Here the test set finds the K closest digits based on the Euclidean distance of two MNIST numbers in their 764 dimensional space. This is probably not the best attempt at making label propagation inductive, but as the test is the same for label propagation and skeletonized label propagation, making a comparison between the test results seems fair. Only the testing accuracy will be displayed in this chapter, but all results will be reference and placed in appendix.

Because the local-separator algorithm is not fully deterministic[BR20, page.12] and because the time to compute a skeletons for a highly connected graph is high, the skeletons are initially created in a range of $k = 2$ to $k = 40$ in the full training data set, and reused for the different levels of label sparsity. Although the difference between skeletons are marginal, this does make the comparisons between different experiments more fair.

1.2 Expectations

Before discussing expectations, an intuition is needed to translate how much the local-separator algorithm affects the results. From **??** a clear link between the connectivity of the graph and the amount of compression skeletonization does is established. If k is set to 1 in the graph-construction algorithm, the output graph of skeletonization will be the same as the input graph, and as k increases so will the amount of compression. This can be seen in Figure 1.1, showing the size of the skeletonized graph as k increases. In short, results will be more affected by skeletonization as a higher number of neighbors are used to construct the graph.

The expectation is that under the circumstances of a large, sparsely labeled training set and a high amount of neighbors, the MDS algorithm will be able to define clear clouds of similar points. The local-separator algorithm can use these to create a skeleton with separator nodes consisting of very similar digits and an edge-set consisting of the most similar nodes to each other. The harsh setting and the high amount of compression in combination, we hope, will equal, if not outperform, label propagation.

It could also be possible, that if the skeletonized graph gets very compressed, the implementation of the majority vote of the label in a node, when converting from pygel-graph to matrix-graph, could worsen results.

1.3 Parameters

The experiments will revolve around two parameters: the number of neighbors K and the amount of forgotten labels FP[1].

Choosing parameters is a case of qualified guesses as well as points of interest. Experiments and parameters for the specific hypothesis described in Section 1.2 will be tested, but to be more encompassing, a wider range of parameters are chosen. Through early

[1] Forgotten Percentage. If FP=80%, 80% of labels are removed

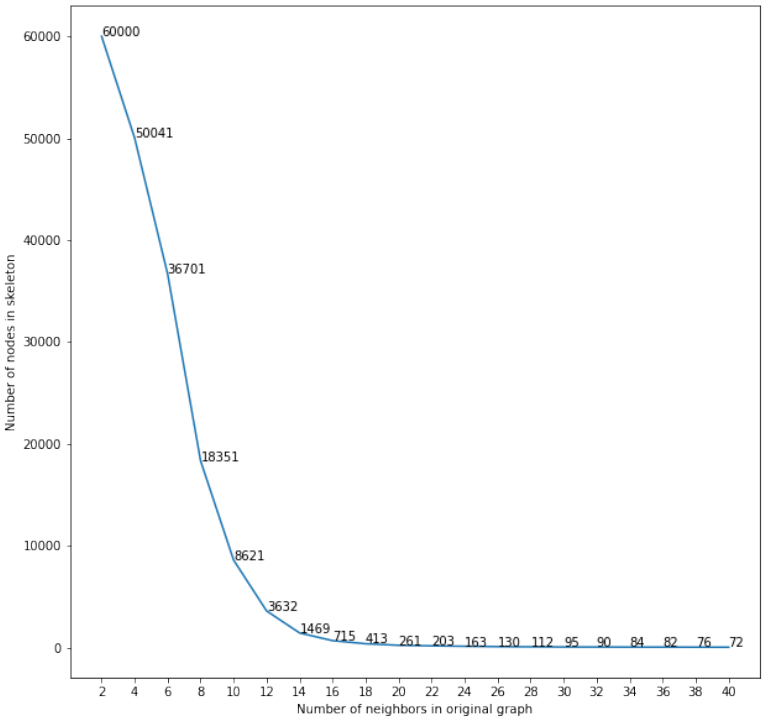


Figure 1.1: Size of skeletonized graph consisting of 60.000 nodes. Graph is constructed with the k-nearest neighbors algorithm at an increment of 2 between $k = 2$ and $k = 40$

testing, a high number of neighbors increases the time spend on skeletonizing quite significantly. Therefore, a ceiling of 40 neighbors is set, as the amount of compression seems to flatten out at around this value.

A quick summary of the ranges of parameters can be seen in Table 1.1.

Parameter	**Min**	**Max**
Number of neighbors K	1	40
Percentage of forgotten labels FP	90%	99.9%

Table 1.1: Summary of ranges of parameters

1.4 Varying number of neighbors

Increasing the number of neighbors in parity with skeletonization will show whether a compressed skeleton graph decreases, increases, or does not influence the result of

label propagation. If the accuracy decreases, skeletonization may not be helpful at all. If it increases, skeletonization has a positive effect on label propagation. If it does not change it does mean, that the skeletonization algorithm is good at keeping the features of the graph intact, but unfortunately does not improve the label propagation algorithm.

1.4.1 Results

At an earlier stage of experimentation the skeletonization algorithm was run and tested on a subset of the test set. These results showed, that there was almost no difference between including and excluding the MNIST data set. The graphs for the results can be seen in appendix **??**. The results of running the two algorithms using the whole test set shows a more positive outcome, and can be seen in Figure 1.2, and Table 1.2. Both tests ran on the same graphs and skeletons, the only difference is the size of the test data set.

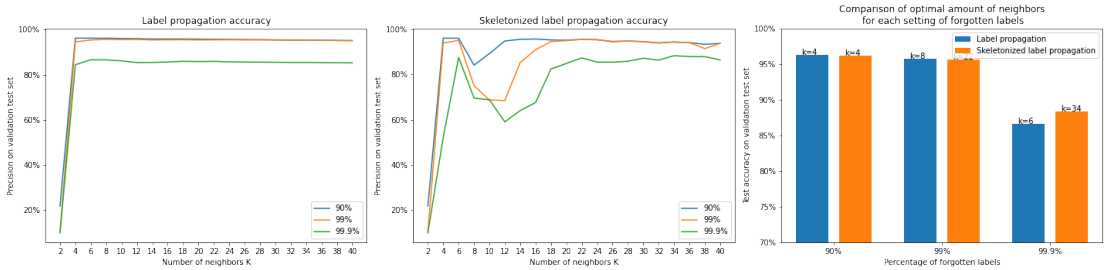


Figure 1.2: Validation set result. The first two figures show the result of an increased number of neighbors used to construct the graph, and right image shows a comparison of the best result for the two algorithms in each setting of sparsety of labels.

	Forgotten percentage	90%	99%	99.9%
Label propagation	Accuracy	96.26%	95.73%	86.64%
	Number of neighbors	4	8	6
Skeletonized label propagation	Accuracy	96.18%	95.61%	88.4%
	Number of neighbors	4	22	34

Table 1.2: Highest accuracy achieved and paired value of k. Values match the bar chart in Figure 1.2

These result show some interesting and surprising properties.

The label propagation algorithm needs more than 2 neighbors to produce a good result, but when this is saturated, the amount of neighbors affects the accuracy minimally, although there is a down-going trend. This is also visible as the shape of the three different settings are similar with an offset. The results also suggest, that the label propagation algorithm can deal with quite a significant absence of labels (99%) without it affecting the result of the algorithm. This can be seen, as the result of 90% and 99% unknown labels are practically identical, when number of neighbors is high.

The skeletonized label propagation is actually a bit surprising. Before conducting the experiments I had a suspicion that an increase in compression would worsen results, as the probability of squeezing different labels into the skeleton nodes would affect the result, especially at a high amount of labels. The results however suggests otherwise. As with label propagation the accuracy trends downwards as the number of neighbors is increased from 12 to 40. But biggest of surprises was the valley of low seen at all three levels of label sparsity. Initially the suspicion is, that labels of different labels are somehow put into the same nodes in this interval. To further explore this, and to estimate how the implementation of the majority node in each skeleton node affects the result, we look at the percentage of nodes with multiple labels in them. This can be seen in Figure 1.3.

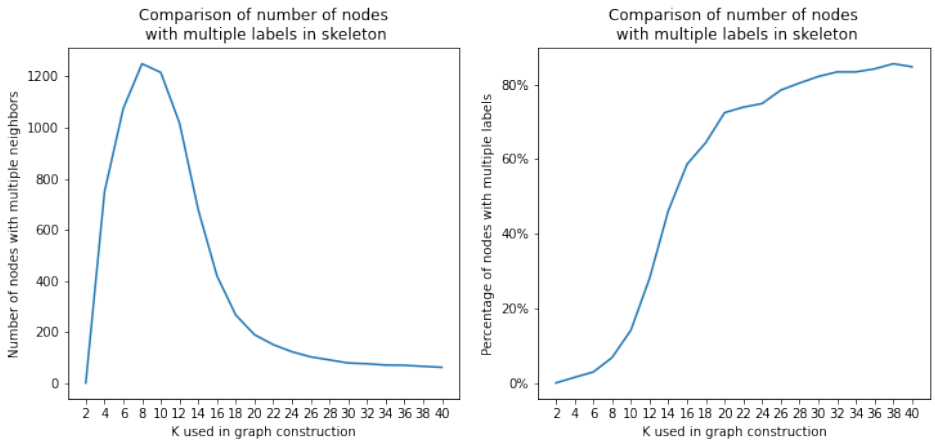
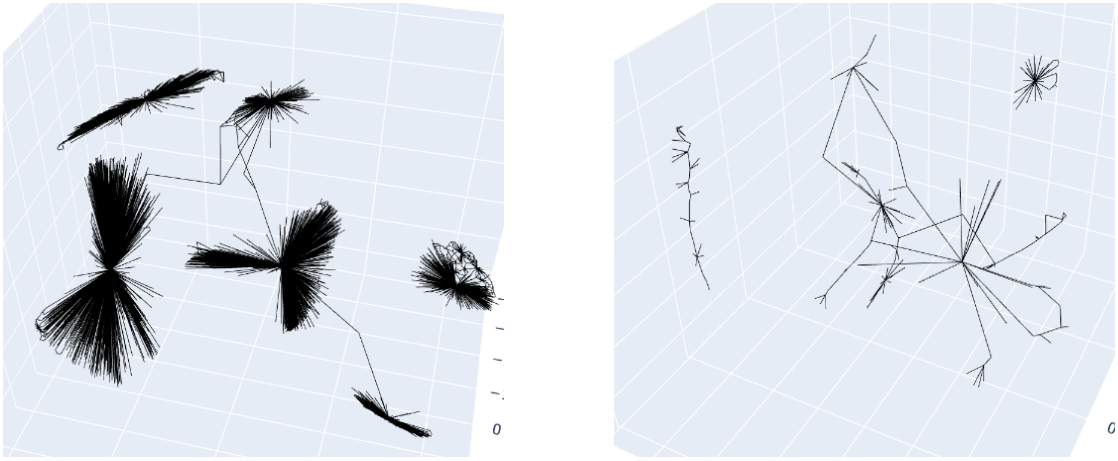


Figure 1.3: Percentage of nodes with multiple labels in them

To digest this, lets pick the information apart. First thing to notice is that there are no multiple labels in nodes when $k = 2$. This is not very surprising as Figure 1.1 shows that the skeleton has the same number of nodes as the original graph, and so no nodes have been compressed together. The next and most interesting aspect of this result is, that the number of nodes with multiple labels affect the result more than the percentage of nodes. This can be seen, as the spike in number of nodes with multiple labels align with the drop in accuracy of the skeletonized label propagation algorithm, in the interval between $k = 8$ and $k = 12$. This affect can be sees as ripples all three levels of label sparsity, where the decline of accuracy starts at around 6 neighbors, and with some delay first stabilises at around 20 neighbors.

The culprit of this behavior is likely the majority vote, as what is probably happening, is that the skeletonization algorithm has not been able to compress the graph enough to really guide the direction of the label propagation algorithm, and instead the connectivity from the original graph is intact, and within this true labels are changed. To explore this further, lets look at the structure of the skeletons. To keep the data manageable we look at the skeletons created at a high number of neighbors as they have less nodes, and

are therefore easier to look at and understand. With label sparsity set to 99.9% we see two skeletons reaching the lowest point and returning to a somewhat stable accuracy at $k = 12$ and $k = 22$. We expect the less compressed skeleton to be similar to the original graph, with edges between different regions of labels, and the very compressed skeleton to have removed a lot of these edges. Results can be seen in Figure 1.4.



(a) Skeleton produced from graph with k=12

(b) Skeleton produced from graph with k=22

Figure 1.4: Skeletons from graph construction

Unexpectedly the skeletons seem to tell a different story. The grouping of labels together is still a problem, but what seems to be the biggest problem is the reapereance of the star shape from **??**. As explained in **??**, the problem seems to be with centers of the graph being heavily connected. This creates a lot of separators, that either in the packing stage results in a singular node (the center of the stars), or it could happen in the skeleton extraction phase, where cliques with valency ≥ 3 are grouped together. This issue is a choice in the packing/skeleton-extraction algorithm, chosen and optimized for 3D-models, but seem to produce a less compelling output for this arbitrary graph of embedded data points. Looking at b in Figure 1.4, the star shapes are more of less dispersed, and the few stars that are left, are within the region of a specific label. To furthermore check if the stars make the results worse, lets look at exactly what types of digits are guessed wrong. Looking at the embedding the graphs are based on in **??** it is seen, that labels 0 and 1, are placed in distinct seperated clusters. This is reflected in both skeletons, as there are no edges between the two outside clusters. There is however placed a star in the middle of the cluster consisting of 3s, 5s, and 8s, and the cluster consisting of 4s, 7s and 9s. If the stars are the reason for the dropped accuracy this will be reflected, as these two clusters will guess wrong in at these labels. As the skeleton created with $k = 12$ seems to suffer most from stars, we will look at its distribution of true and predicted labels. Looking at the heatmap in Figure 1.5, as expected wrongly predicted labels are in the two clusters previously mentioned.

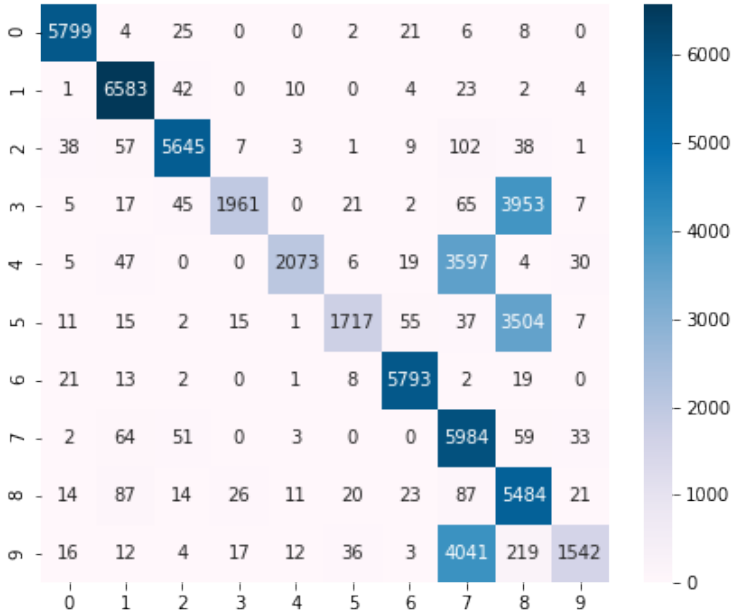


Figure 1.5: Heatmap of true labels and predicted labels. Rows are the true label of nodes, and columns are the predicted label.

1.5 Discussion

Despite its shortcomings the skeletonized label propagation algorithm improved results at a high level of sparsity, outperforming the regular label propagation algorithm at almost 2 percentage points. We also see that the number of neighbors for the optimal result was quite high at 34 neighbors. This suggests that not only did skeletonization not limit the label propagation algorithm, it actually boosted it. It would be interesting to see if resolving the issues at the lower amount of compression would boost the result of label propagation further.

The shortcomings of the skeletonized label propagation algorithm involves two factors, and the interaction between them, that seems to limit the accuracy: the majority vote in nodes with multiple labels, and the star shaped clusters. The star structure limits the paths for information to pass through the graph structure. We see a lot of spikes going from the center of the star, and the nodes at the ends of these spikes will have to pass through the center. Because the center of nodes is created from a big collection of nodes, at least one label will most certainly be placed in the separator. Because label propagation reinserts the values of all known labels (clamping) given at initialization of the algorithm, in the case that the center is assigned label, the information from the different spikes can't propagate through the graph as it is reset every iteration of the algorithm. In addition to this the information network of edges between nodes are completely removed, and replaced with stars instead, limiting the label propagation algorithm

even further.

Earlier we hypothesized a way to solve the issue of clamping by instead of deciding the label by majority a probability of each label is inserted as the label of the row, like the second step of label propagation. Whether this row of probabilities should be clamped or allowed to change is not immediately clear and would need some experimentation. The implementation can also be a bit tricky as labelled and unlabelled nodes would be a part of the separator, and assigning labels to these data points could not be done through label propagation.

The biggest problem seems to be the star structure reflected by the results in Figure 1.5. If instead of a star structure the skeletonization algorithm produced a small but connected cluster, the probability of skeleton nodes with a single collection of the identical labels might be higher. The most intuitive way to sabotage the creation of stars, would be to make the center less connected, forcing the skeletonization algorithm to spread out separator, and making the amount of compressed nodes in the separators more equal. This could be done by limiting the max distance two nodes are allowed to have to create an edge in the graph construction phase.

A popular method of constraining distance between nodes is ϵ-neighbor-based graph construction algorithms. This was not implemented in this thesis as KNN-based graphs outperform ϵ-neighbors based ones with better scalability according to [Son+21, page.5]. The major reason for disregarding this algorithm was that a poor choice of ϵ can be misleading and might produce worse results, and given the time constraint of the project, it thus seemed like a KNN-based approach was better suited. Looking at the result of this implementation it becomes quite apparent that an ϵ-neighborhood-based construction approach might even be a necessity to get consistent good results.