# Exploring the computational feasibility limits of perplexity in t-SNE for scenarios of limited working memory

Author: Dimitar Netzov[1] | Responsible Professor: Klaus Hildebrandt[1] | Supervisor: Martin Skrodzki[1]

[1]Delft University of Technology

## t-SNE, and the issues that come with it

Making sense of large datasets is a challenging task for researchers. An approach that leverages human intuition and pattern recognition is to visualize the often high dimensional data into two or three dimensions. This is dimensionality reduction, and a popular algorithm for it is t-distributed stochastic neighbor embedding (t-SNE).

- **The Good:** The reason for t-SNEs' popularity is its ability to model non-linear relationships in data. [It does so by trying to model the local structure]
- **The Bad:** An important parameter in t-SNE is the perplexity. It is effectively the size of the local neighborhood to be considered by the algorithm. Since the optimal size is dependent on the underlying dataset, there is no single "best" value for the perplexity.
- **The Ugly:** While some heuristics have been developed to offer guidance, in the end, selecting the perplexity value revolves around trial and error with intuition.

With the huge size of datasets used in practice, the process of tuning the perplexity is often very time-consuming. But aside from time, it also requires memory. This can become a problem as exceeding the available memory leads to even higher time penalties or worse - crashes.

## In this poster

To help researchers avoid slowdowns and crashes, this [poster/paper] explores the computational feasibility of combinations of data sizes and perplexities for a given hardware by answering these questions:

1. Can the algorithm be divided into sections based on memory consumption patterns?
2. What is the formula for the peak memory consumption with respect to data size $N$ and perplexity $P$?
3. Can the formula be derived via samples from runs with small N and P parameters?
4. How can the formula be improved?

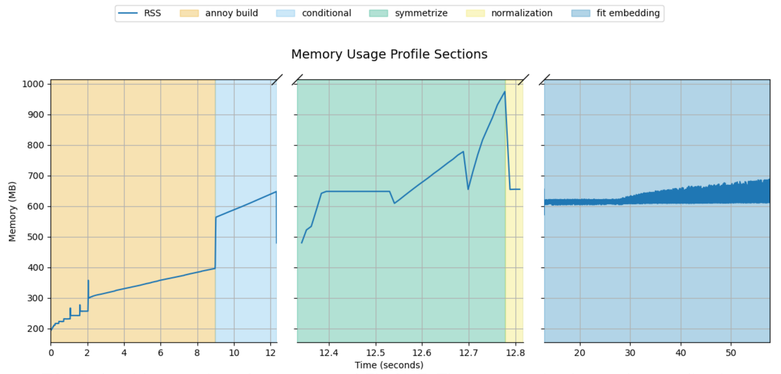
## Methodology

The implementation of t-SNE used in the experiments was openTSNE with MNIST as the dataset. All experiments were run on Windows 11 in a separate process to isolate memory usage. The actual peak usage was measured as the RSS of the process, monitored during execution. The following experiments/analyses were conducted:

- Real measurements: Measure peak RAM usage during execution for smaller perplexities and sample sizes to obtain a testing set.
- Theoretical estimate: Estimate the space complexity by analysing the algorithm's code.
- Compare measurements to estimates: See how well the theoretical estimate matches the measured values.
- Estimate the formulas from experimental data: Using regression and the training set obtained earlier, estimate the space complexity formula. Use it to validate theoretical findings.
- Verify: Test the theoretical and estimated formulas on high perplexity runs on the whole dataset.
- Improve: Test a modified version of the library algorithm to verify a lower memory consumption.

## Sections

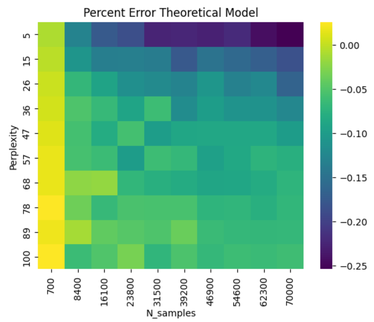
The algorith can be divided into five sections: Annoy KNN estimation, conditional matrix computation, symmetrisation, normalisation, and the fitting of the low-dimensional embedding.



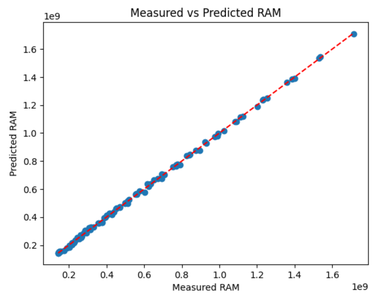

openTSNE algorithm sections for Annoy and FIt-SNE. The symmetrization and normalization are zoomed in on the time axis for visibility.

## Theoretical analysis

The theoretical formula found was $204N + 400N + C$, where C is a constant that heavily depends on the setup. In our case, $C = 126000000$




Mean percentage error for the small parameter samples, with the constant predicted by the analytical model



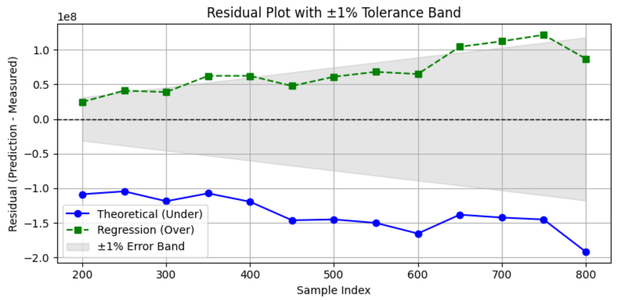

Reconstruction of testing set by the Lasso model

## Analytical model

To synthesize the memory usage formula from the test samples, the Lasso regression model was used on the test set. This resulted in the formula $208 \cdot N \cdot P + 1285 \cdot N + 126000000$. The reconstruction has a mean absolute percentage error of 2.3% and an $R^2$ value of 0.9993.

## Validation on larger values of N and P

To validate the theoretical and analytical approaches, their predictive power was measured for runs on the entire dataset, $N = 70000$, with perplexities up to $P = 800$.




Mean error of between the models versus peak memory usage, with 1% band marked in.

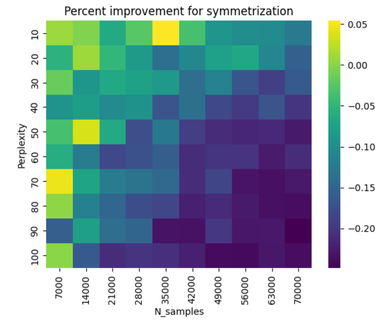
## Improvement of the symmetrization section

The symmetrization section introduces multiple copies of the P matrix due to the calculations being done on a single line. By splitting up said line, an improvement in memory usage can be seen for higher values of $N$ and $P$.

```
if symmetrize:
    P = (P + P.T) / 2
```

```
P += P.T
P /= 2
```

Original and modified code




Mean percent error between the default and modified implementation. Lower is better.

## Future Work

Future Work

1. Explore the impact that going into swap memory has on the execution times of the algorithm.
2. Explore the limits of using swap at which crashes occur.
3. Test the approximation method for other implementations of t-SNE
4. Derive a formula for the space complexity of the embedding algorithms.
5. Construct a tool that utilizes the results of this research to inform researchers of computational feasibility, given the parameters of their experiment.



D.P.Netzov@student.tudelft.nl