CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Towards Unknown Traffic Driving Pattern Discovery with Active Learning

Master's thesis in Computer science and engineering

JARL SANNA, WENNERBLOM JULIA

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Towards Unknown Traffic Driving Pattern Discovery with Active Learning

JARL SANNA, WENNERBLOM JULIA

CHALMERS

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Towards Unknown Traffic Driving Pattern Discovery with Active Learning
JARL SANNA, WENNERBLOM JULIA

Supervisor: Morteza Hagir Chehreghani, Department of Computer Science and Engineering
Advisor: Sadegh Rahrovani, Volvo Cars Corporation
Examiner: Morteza Hagir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Towards Unknown Traffic Driving Pattern Discovery with Active Learning
JARL SANNA, WENNERBLOM JULIA
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The promise of autonomous vehicles is frequently discussed and the traffic landscape is expected to change drastically with the technology of AD. Therefore rigorous testing is essential for the reliance on and trust in the system. The vast amounts of labelled data for testing is not a trivial thing to obtain. One potential approach to this issue is Active Learning as its purpose is to produce a robust data set with minimal human interaction. The aim of this project is to examine the effectiveness of active learning for annotation of scenario data collected by a Volvo Cars Corporation (VCC) vehicle. Active learning trains a classifier on a small initial annotated data set and uses it to determine which unlabelled data points need annotation by a human. The classifier is then retrained with the updated annotated set until the budget of queries is spent. In this study, active learning is performed on the latent space produced by multivariate Time Series t-Distributed Stochastic Neighbor Embedding (mTSNE), Recurrent Autoencoder (RAE) and Variational Recurrent Autoencoder (VRAE). Investigations are made into which embedding, classifier and query strategy is most suitable for the task of performing active learning on VCC's trajectory data. A study is also performed on the impact of different degrees of class imbalance in the data. Area Under the Curve (AUC) and F1 score with regards to number of queried points are used as measures of performance. In many cases, active learning has proven an effective tool. We can conclude that the mTSNE embedding with the Support Vector Machines algorithm (SVM) as a classifier outperforms the other models, with both high AUC and F1 score in addition to a low run time and high stability. Entropy querying is observed as the most suitable query method. The separability of the mTSNE generated latent space provides a less complex model, although the mTSNE transformation itself is very computational heavy. RAE also performs well, though combined with a Neural Network (NN) it struggles with detecting the smaller class as the class imbalance increases. VRAE proves to be a suboptimal choice of embedding, since it performs worse than the two others. We conclude that for mTSNE, 50 queries is sufficient to reach a high AUC and F1 score for most class imbalances, and for RAE, that number is 125. The potential of active learning to act as an unknown class detector was also investigated using RAE and VRAE embedded data. Cut in was regarded as the unknown class, and performance was measured in terms of number of queried cut ins. The results show that for a budget size up to 200 queries RAE with SVM classifier queries the most cut ins, while for a larger budget sizes VRAE with SVM queries the most cut ins.

Keywords: Active Learning, Unknown Detection, Annotation, Time Series Analysis, mTSNE, SVM, Neural Network, query.

Acknowledgements

We want to give a big thank you to our supervisors Morteza Chehreghani, Sadegh Rahrovani and Maria Svedlund for their support, hard work and brilliant insights.

Julia Wennerblom and Sanna Jarl, Gothenburg, June 2021

Contents

List of Figures

1
Introduction

Autonomous driving is a new and exciting technology that offers an array of opportunities for society. It could be a way to reduce accidents in traffic, ease gridlock issues and allow for more comfortable and productive commutes. Every year, around 240 people die in traffic in Sweden [6] and 1.35 million world wide [7]. Many of the deaths related to traffic accidents are attributed to the human factor. A large portion of the world population also commute to work every day, taking hours per week away from spare time. Regular vehicles could be improved with autonomous functions that enable humans to use this time to get some extra sleep, read emails, or engage in other activities.

Volvo Cars Corporation (VCC) produces cars with assisted driving functions, and in order to comfortably integrate these improvements in society they must be safe and well tested. Testing these functions on the road is estimated to require hundreds of million of failure free kilometers driven [8], and is therefor an impractical approach. It is instead desirable to move the testing to a virtual environment, which requires a large set of robust, annotated data. Some types of data are easy to find annotated (names in a population, viewership numbers for a TV program, population size), however for some tasks, such as assisted driving functionalities, the annotation is quite expensive and complicated. The cost comes in the form of extensive human interaction, where an expert might be required to label the data correctly, the data could be difficult to classify, or it could take a long time to go through. The data concerning autonomous vehicles comes in the form of time series, and the complicated nature of this data opens up for the risk of automated annotation algorithms missing rarer classes, or misclassifying fringe cases. On top of this some cases are rare on the road and it could take a long time to find multiple examples of them, so it is important to find them, both for verification and generation purposes.

For the objective of producing this high quality, low cost data set, active learning could be used, either as a classification tool, or as an extra step in the verification process of the labels. The purpose of this method is to produce a reliable annotated data set with minimal human interaction, while still keeping a person involved in the process. An expert only needs to manually annotate cases with high informational value, and the involvement of humans ensures a higher robustness of the annotation and completeness of the scenario catalogue. A classifier will then label the bulk of the data based on that information.

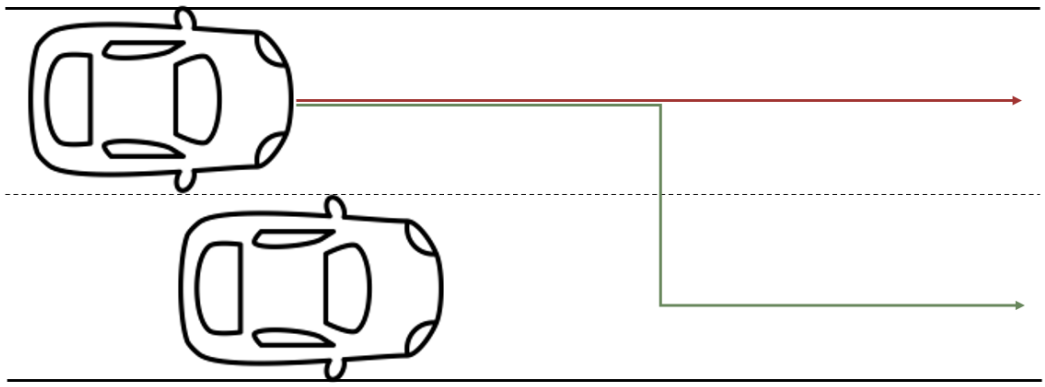


Figure 1.1: Illustration of two of the classes in the data. The red line is a left drive by and the green line is a cut in [1].

1.1 Purpose

The goal of this project is to implement and investigate the effectiveness of active learning on the VCC trajectory data and to what extent it can be used to detect unknown classes. Finding new classes is important for AD verification as it contributes to scenario catalogue completeness. The data contains scenarios collected by a vehicle driving on a highway; the classes are ”cut in”, ”left drive by” and ”right drive by”, as shown in Figure 1.1. The problem will be approached by transforming the data to latent space with multivariate Time Series t-Distributed Stochastic Neighbor Embedding (mTSNE), Recurrent Autoencoder (RAE) or Variational Recurrent Autoencoder (VRAE) and then perform simulations of active learning. A Support Vector Machine (SVM) and a Neural Network (NN) will be used as classifiers in the latent space, while entropy, margin and random querying will be used for querying. The performance of the model will be evaluated based on Area Under the Curve (AUC) and F1 score with regards to budget size. RAE and VRAE embedded data will be used for unknown class detection, with either an NN or SVM classifier. Number of queried cut ins will be used as a measure of performance.

1.2 Limitations

The scope of this project is to implement an active learning algorithm for classifying driving scenarios. Some different query strategies will be investigated and evaluated, however, looking at combinations of different query strategies is beyond the scope of the project. The classification methods will be limited to SVM and a Neural Network, and no combination of the methods will be attempted. No detailed investigation will be performed to find the optimal architecture for the Neural Network after finding one that works well enough for our purpose. Similarly no detailed investigation will be performed to find the optimal mTSNE perplexity or

SVM parameters after finding ones that work. The data from VCC will not be examined further to improve the quality. The results of the investigation might not be applicable to other types of data sets, since it is performed specifically on driving trajectories. Unknown class detection will also be carried out, using RAE and VRAE embedded data, together with the same classifiers and query strategies already mentioned. Anomaly detection will not be part of the scope.

1.3 Structure of the Report

In Chapter 2, ”Theory” the underlying theoretical background to the project is presented. Chapter 3, ”Methods”, describes the approach to the problem and the tools used. In Chapter 4, ”Results”, the results of the studies are presented and Chapter 5, ”Conclusion”, draws conclusions from the results.

2

Theory

The transformation from time series to more easily managed representations in latent space can be done using a Recurrent Encoder, a Variational Recurrent Encoder or multivariate Time Series t-Distributed Stochastic Neighbor Embedding. Once in latent space, the data can be efficiently and correctly labelled with active learning. This will be elaborated on in the following sections.

2.1 Active learning

Active learning is a semi-supervised technique that aims to label data efficiently to minimize the involvement of humans. The idea is to retrieve the most informative data points and send them to an expert for annotation [9]. The amount of queried points is dictated by the budget. This idea is explained further in Algorithm 1.

Algorithm 1: Active learning

Result: Labels a number of data points and classifies the rest.
train classifier on small annotated data set;
while *budget not empty* **do**
    classify unlabelled data;
    calculate informativeness;
    query most informative point(s) to an expert;
    add queried point to annotated data set and remove from unlabelled data set;
    retrain classifier on annotated data set;
end

There are several ways a query regarding the data points can be asked. The common goal for all strategies is to obtain as much information as possible with as few queries as possible.

2.1.1 Query strategies

There are three commonly used query strategies: random, margin and entropy querying [10]. Let U denote the unlabelled data set. Random is a strategy where a

uniformly distributed informativeness is assigned:

$$I_i^R \sim \text{unif}(0,1), \quad for\ i \in U. \tag{2.1}$$

This means that an object is drawn at random and queried. The second strategy is margin, which computes the informativeness for every unlabelled object $i \in U$ as

$$I_i^M = -[P_C(\hat{y} = \tilde{c}_1|\mathbf{x}_i) - P_C(\hat{y} = \tilde{c}_2|\mathbf{x}_i)], \tag{2.2}$$

where $\mathbf{x}_i$ is the data point, $\hat{y}$ is the predicted label, P_C is the probability of $\hat{y} = \tilde{c}_j$ given $\mathbf{x}_i$, $\tilde{c}_1$ and $\tilde{c}_2$ are the most probable respectively the second most probable classes for $\mathbf{x}_i$ to belong to, as predicted by classifier C. Examples of classifiers that could be used are SVM or a Neural Network. Note that since I_i^M is negative, it is maximized in active learning, meaning that the point for which I_i^M is closest to zero is queried.

The third query strategy is entropy, which assigns the informativeness based on the entropy of the predictive distribution:

$$I_i^E = -\sum_c P_C(\hat{y} = \tilde{c}) \log(P_C(\hat{y} = \tilde{c})). \tag{2.3}$$

The entropy can be viewed as the total amount of information in the entire distribution. In active learning, the point with the highest I_i^E is queried.

2.2 SVM

To solve the constrained optimization problem, define the hyper plane

$$f(x) = \langle w, x_i \rangle + b, \tag{2.4}$$

where x_i is the data and w and b are the slope and intercept, to linearly divide the space into two domains [11]. The problem can be rewritten as

$$\min_{w,\gamma} : \frac{||\mathbf{w}||^2}{2} \tag{2.5}$$

$$\text{Subject to: } f(x) = \begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon \\ \langle w, x_i \rangle + b - y_i \leq \epsilon, \end{cases}$$

where y_i is the label and ϵ is the accepted error. In some cases, the problem cannot be solved without allowing for errors, so we introduce the slack variables ξ_i and ξ_i^* and modify the equations to become

$$\min_{\mathbf{w},\gamma} : \frac{||\mathbf{w}||^2}{2} + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) \tag{2.6}$$

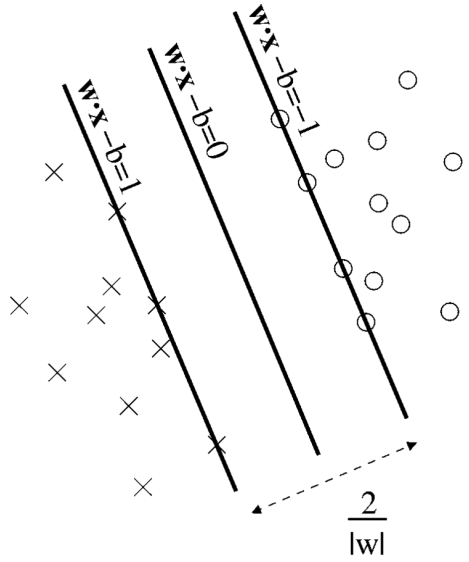


Figure 2.1: Illustration of the SVM boundaries [2].

$$\text{Subject to: } f(x) = \begin{cases} y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0. \end{cases}$$

C is a parameter for regulating the flatness of f.
In the case of nonlinearity in the data, the equations must be modified. In order to solve the problem, the data is mapped to a feature space using the transformation

$$\phi : R^p \to R^d, \tag{2.7}$$

where R^p is the p-dimensional original space, and R^d is the d-dimensional feature space. ϕ can be difficult to compute, so the kernel trick is often used, where the kernel is defined as

$$k(x, x') = \phi(x)\phi(x') \tag{2.8}$$

and is generally easier to compute. There are several kernels, however, the radial basis function (rbf) kernel is the relevant one in this case. The gaussian form of the rbf kernel can be written as

$$\exp(-\gamma||x - x'||^2), \text{ [12]}. \tag{2.9}$$

2.3 Neural Network

Artificial Neural Networks consist of *perceptrons*, ordered in layers, see Figure 2.2. In each perceptron, a calculation is performed according to

$$a_j = \sum_{i=1}^{D} w_{ji}^{(k)} x_i + b_j^{(k)} \tag{2.10}$$

where $w_{ji}^{(k)}$ is the weight and $b_j^{(k)}$ is the bias for the input x_i in the j:th perceptron in layer k, and D is the dimensionality of the input [13]. Next, the activation function

$$z_j = h(a_j) \tag{2.11}$$

is used. There are many options for the activation function, for instance the Rectified Linear Units (ReLU) function [14]:

$$h(x) = \max(0, x). \tag{2.12}$$

The output is calculated with the softmax function [13]

$$p(C_k|x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{2.13}$$

where $p(C_k|x)$ is the probability of predicting class C_k given x.

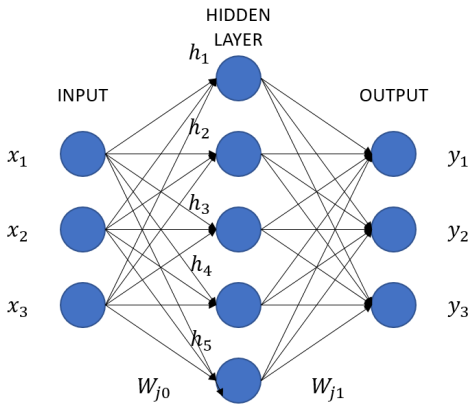


Figure 2.2: Illustration of a fully connected artificial Neural Network.

In order for the network to learn, the error is calculated in the form of a certain loss function. The cross entropy loss function is defined as

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} -\log p(\tilde{y} = \tilde{y}_n|\mathbf{x}_n, \mathbf{w}), \tag{2.14}$$

where E_n is the error for a data point $\mathbf{x}_n$, $\tilde{y}$ is the predicted label for $\mathbf{x}_n$ and $\tilde{y}_n$ is the true label [15]. The weights $\mathbf{w}$ are updated according to

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)}), \tag{2.15}$$

where ∇E_n is the gradient of the error function for a data point $\mathbf{x}_n$ and η is the learning rate [13].

2.4 Evaluation metrics F1 score and AUC

In a classification problem, a predicted label can be either correct or incorrect, as illustrated in the confusion matrix in Figure 2.3. The information in the confusion

		Predicted class: Positive	Predicted class: Negative
Actual class	Positive	TP	FN
	Negative	FP	TN

Figure 2.3: Binary confusion matrix.

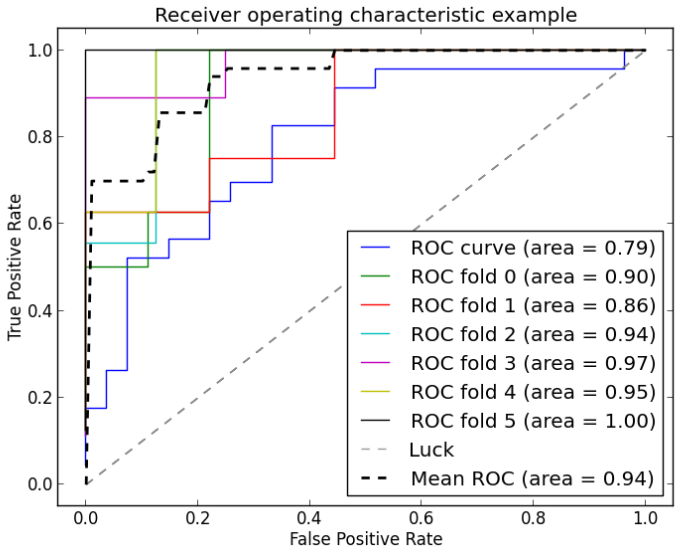


Figure 2.4: Example of a ROC curve [3].

matrix is frequently used for evaluation of a model, two common metrics being AUC and F1 score.

The ROC curve plots the TP rate against the FP rate, as shown in Figure 2.4, with the objective of TP rate being close to 1 and FP rate being close to 0 [16]. The grey dashed line is the performance of a truly random classifier. From this curve we can calculate the Area Under the Curve (AUC). For an optimal classifier, the AUC will be 1.

To understand F1 score, we define the following quantities [16]:

$$\begin{aligned} \text{Precision (P)} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{ and} \\ \text{Recall (R)} &= \frac{\text{TP}}{\text{TP} + \text{FN}}. \end{aligned} \tag{2.16}$$

F1 is then defined as

$$F1 = \frac{2 \cdot P \cdot R}{P + R}. \tag{2.17}$$

F1 score can take values from 0 to 1, with 1 being the best.

2.5 Multivariate Time Series t-Distributed Stochastic Neighborhood Embedding

mTSNE is a method for transforming data of various size to the latent space [17]. This method is a combination of stochastic neighbour embedding and t-distributed neighbour embedding. High dimensional euclidean distances between data points are converted into probabilities that represent similarities. The similarity of point x_j and x_i is given by the conditional probability

$$p_{j|i} = \frac{exp\left(-||x_i - x_j||^2/2\sigma_i^2\right)}{\sum_{k \neq i} exp\left(-||x_i - x_k||^2/2\sigma_i^2\right)} \tag{2.18}$$

that x_i would pick x_j as its neighbour if both points where drawn in proportion to their probability density under a gaussian distribution centered at x_i. This means that for points close to x_i, $p_{j|i}$ will take a higher value and a lower value for points further away. For the low dimensional counterparts y_i and y_j of the high dimensional points x_i and x_j, the low dimensional conditional probability $q_{j|i}$ can be computed in the exact same was as Equation 2.18. If the low dimensional points correctly models the high dimensional points, $p_{j|i}$ and $q_{j|i}$ will take the same value. Stochastic neighbourhood embedding (SNE) aims to find a low dimensional representation that minimizes the difference between $p_{j|i}$ and $q_{j|i}$. In order to minimize the difference, the KL divergence is minimized. The cost function is described by Equation 2.19.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{2.19}$$

If the KL divergence is very hard to optimize, then t-distributed neighbourhood embedding is a method to go around this problem.

2.6 Recurrent Autoencoder

Recurrent Neural Network is a type of network that is used to learn sequences, with the big advantage of taking input of variable length [18]. A common choice of architecture is the Long Short Term Memory (LSTM) cell, [19], since this excels at learning long sequences. Recurrent Autoencoder is a network that uses two Recurrent Networks. The architecture of RAE is illustrated in Figure 2.5, and contains an encoder and a decoder, which are Recurrent Networks comprised of LSTM-cells. The encoder forces the data into a lower dimensional space (latent space), essentially extracting features of the data. From there, the decoder uses the extracted features

to reconstruct the sequence. The loss is calculated with mean squared error (MSE) between the original sequence and the reconstructed one, and the weights in the network are adjusted accordingly [20].

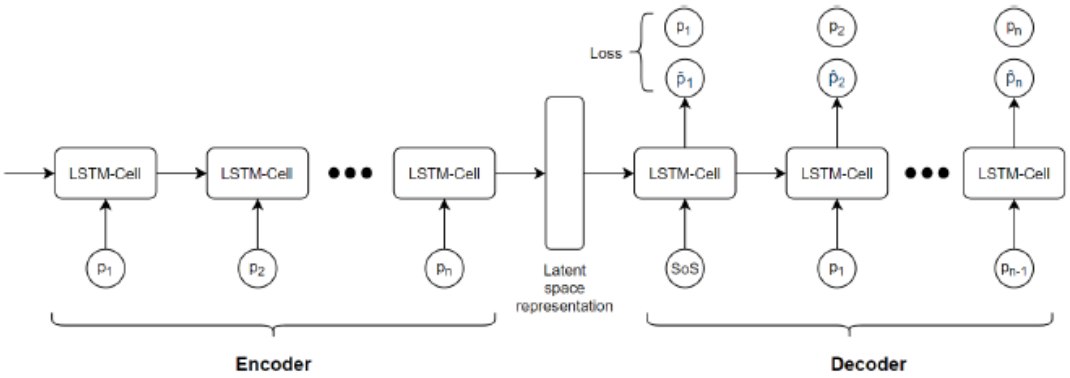


Figure 2.5: The general structure of Recurrent Autoencoder [4].

The LSTM-cell [19] is shown in Figure 2.6 and is defined by the equations

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.20}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.21}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{2.22}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.23}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.24}$$
$$h_t = o_t \odot \tanh c_t. \tag{2.25}$$

t is the time step, h is the hidden state, c is the cell state, x is the input, i is the input gate, f is the forget gate, g is the cell gate, o is the output gate, σ is the sigmoid function and $\odot$ is the Hadamard product. W and b are weights and biases.

The gates are used to create the strong long term memory. In Figure 2.6, from the left, the gates are forget gate, input gate, cell gate and output gate. The forget gate is used to choose what old information in the cell state to forget, where a value of 0 means keep nothing, and 1 means keep everything. The cell state is then updated accordingly. The input gate decides what new information to remember and the cell gate updates the cell state. After this the output gate decides what to output, and the new hidden state is calculated.

2.6.1 Variational Recurrent Autoencoder

VRAE is very similar to RAE, with an important bayesian trick. With a regular autoencoder, there is nothing to enforce that the latent space is cohesive, that is that data points similar to each other in the original space are encoded close to each

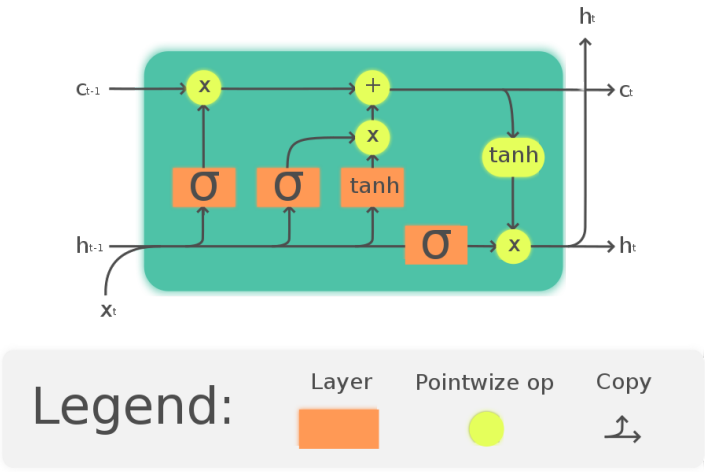


Figure 2.6: Schematic image of the LSTM-cell [5].

other in latent space. The goal of VRAE is to enforce this. Instead of being encoded as a point in latent space, the data is encoded as a gaussian distribution [21]. From this distribution a point is drawn and used as input to the decoder. A way around this for the network is to make the standard deviation very small and the mean any number, and we would return to the case of RAE. Therefor it is important to ensure that the mean and standard deviation stays within bounds. This is enforced by using the KL divergence as a loss, combined with MSE.

3
Methods

The project is executed by performing simulations of active learning on VCC's trajectory data, described in section 3.4, according to Algorithm 1. Instead of the step "query most informative point(s) to a human", labels generated from a set of basic, class defining rules are used as ground truth. Specifics regarding the class definitions can be found in [4]. The simulations are executed using Python's machine learning toolbox Pytorch and it's statistical toolbox sklearn [22]. mTSNE, RAE and VRAE are used for dimension reduction and active learning is performed on the latent space produced. The performance of the different methods is evaluated on AUC and F1 score.

3.1 Dimension Reduction

Data in the form of time series is difficult to analyze, so in order to perform active learning the data is transformed to a latent space. This is done using the tools developed in [4] (mTSNE and Recurrent Autoencoder), as well as a network developed by us (Variational Recurrent Autoencoder). Once in latent space, the data is analyzed with active learning.

Both autoencoders consist of an encoder and a decoder built from LSTM-cells. VRAE has an extra layer that encodes the transformed variables to distributions in latent space, and then draws a sample from the distribution as input to the decoder. VRAE and RAE both have two stacked LSTM-cells, 64 features in the hidden state, and 64 features in the latent space. The mTSNE perplexity was set to 37.5.

The output of mTSNE is 2-dimensional while the output of the autoencoders is 64-dimensional. For the autoencoders, the hidden states are used as the data for clustering. Figure 3.1 shows mTSNE embedded data on a data set containing 10% cut ins. Red, white and blue dots correspond to cut-in, right drive by and left drive by respectively.

3.2 Active Learning

Several of the active learning architecture options are tested in the project, in order to find the alternative most suitable for this data, see Figure 3.2 for details on the options. The querying strategies examined are entropy, margin and random

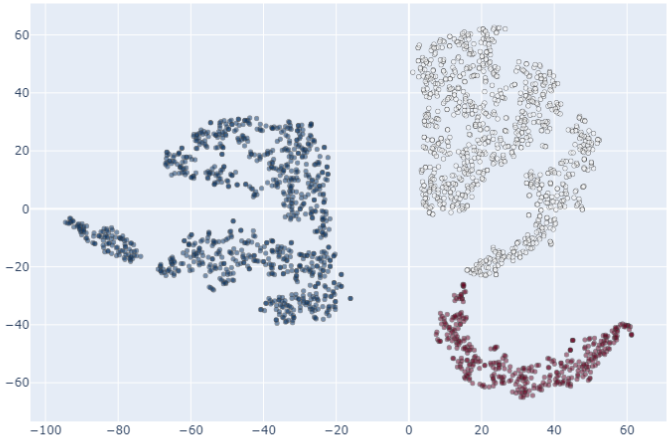


Figure 3.1: The clusters formed in the latent space with data containing 10% cut ins. Note that the three clusters are separable.

Figure 3.2: Flow chart of the steps involved in active learning.

querying. The classification methods used are SVM and a fully connected Neural Network consisting of 2 layers with 128 and 256 neurons respectively. For VRAE, a network with 5 layers with 64, 128, 256, 128 and 64 neurons respectively is also used, this due to poor performance of the smaller network. Each layer is batch normed and activated with ReLU. For SVM, C is set to 1 and γ is determined by the function $1/(n_{features} \cdot \sigma_x^2)$, where σ_x^2 is the variance of the data x. The results are evaluated on area under the curve (AUC) and F1 score for number of queried points.

3.3 Class distribution

The distribution of the classes in the unprocessed data is roughly equal amounts of left and right drive by, and approximately 10% cut ins. For this reason, the main analysis is performed on data with this class distribution. However, an investigation into the impact of class distribution is also conducted with $\alpha = 33, 10, 5$ and 1, where α is the percentage of cut ins in the data set. The case $\alpha = 33$ will be referred to as ”balanced set”. The mTSNE latent space for these data sets can be found in Figures A.1, 3.1, A.2 and A.3 respectively.

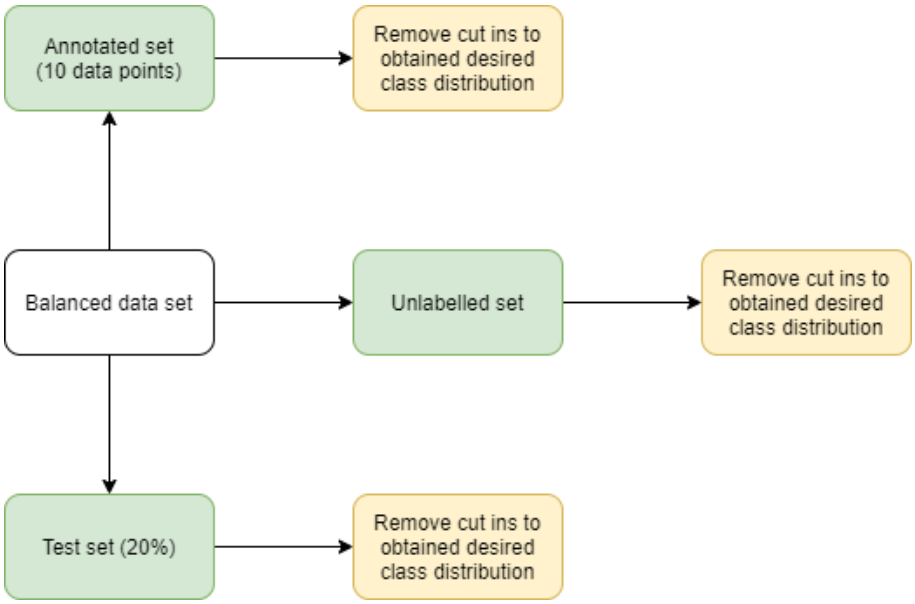


Figure 3.3: Flow chart over the data split process.

3.4 Data

The data used in this project is in the form of time series and collected by a car driving in traffic, the ego. The data is provided and collected by Volvo Cars Corporation. From the available parameters, longitudinal and latitudinal position of the tracked vehicles are extracted. This means that the trajectories analyzed are relative to the ego vehicle. The classes of trajectories are cut in, left drive by and right drive by.

An overview of the data split can be seen in Figure . The data sets are divided into three sets with balanced classes: annotated set, unlabelled set, and test set. The annotated set is used for training the classifiers, the unlabelled set is the set points are queried from, and the test set is used to evaluate performance. These sets are kept as fixed as possible across all simulations. For the unbalanced data sets, the same sets are used, but some cut ins are removed to get the correct distribution. This means that the data sets for those sets are smaller, but otherwise identical. The annotated set, however, only has 10 points, so when cut ins are removed from it, other points take their place. The size of the data sets can be found in Table 3.1.

Table 3.1: Table of the number of points in each data set.

Data set	Annotated set	Unlabelled set	Test set
Balanced set	10	2211	615
$\alpha = 10$	10	1769	492
$\alpha = 5$	10	1563	435
$\alpha = 1$	10	1489	415

3.5 Unknown class detection

The same data set containing 10% cut is used to perform unknown class detection, where cut in is treated as an unknown class. RAE and VRAE embeddings are considered for this purpose. For unknown detection the autoencoders are retrained on left drive by and right drive by trajectories only. When a cut in is given as input during active learning the aim is to query this point. To evaluate performance the average number of queried cut ins over five runs with the same budget is used as a measure.

3.5.1 Data set

After transformation to latent space by either RAE or VRAE, the classes are relabelled so that left drive by label 0, right drive by label 1 and cut in has label 2. This was necessary in order for the NN to handle the input as it requires the labels to start from 0. All cut ins are separated from the the data set and relabelled as 0, then appended to the end of the unlabelled set. In this way it is possible to keep track of all cut ins. Once a cut in is queried, it is removed from the data set and not given back to the algorithm.

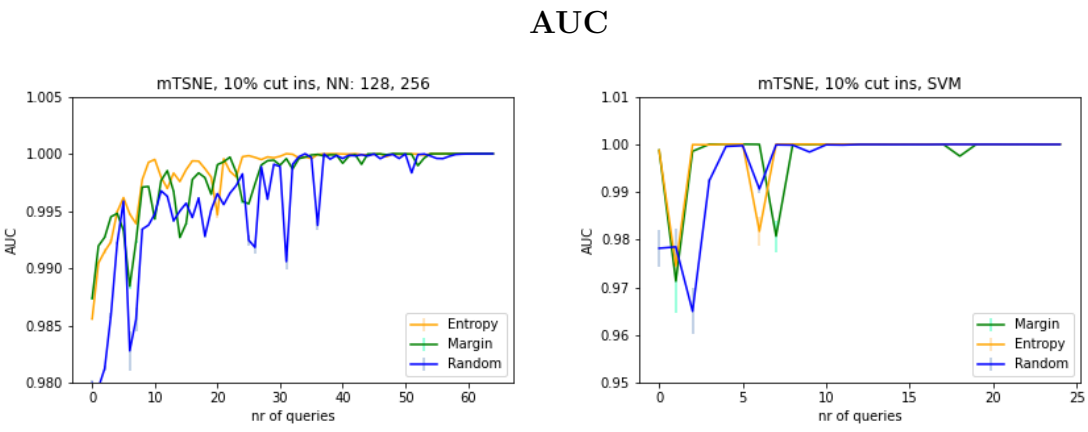
4

Results

The impact on performance of three different factors are considered: embedding, classifier and query method. The classifiers used are SVM and a Neural Network. The different embeddings of the data are mTSNE, Recurrent Autoencoder and Variational Recurrent Autoencoder. Three different query strategies are considered, random, margin and entropy querying. Balanced data was investigated, as well as unbalanced data sets consisting of 10%, 5% and 1% cut ins.

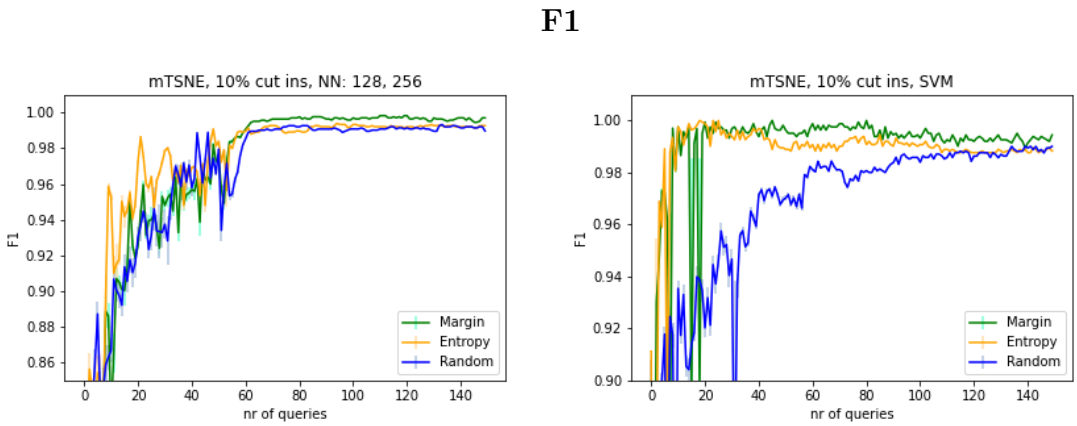
4.1 mTSNE

Two different classifiers were used: SVM and a Neural Network. The AUC for different number of queried points can be found in Figure 4.1. The F1 score for number of queried points can be found in Figure 4.2. The data plotted is an average over 10 runs, and the variance is represented by the lighter colored lines. The data sets consist of equal amount of left drive bys and right drive bys, and 10% cut ins.



(a) Neural Network **(b)** SVM

Figure 4.1: AUC for different budget size with data embedding from mTSNE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



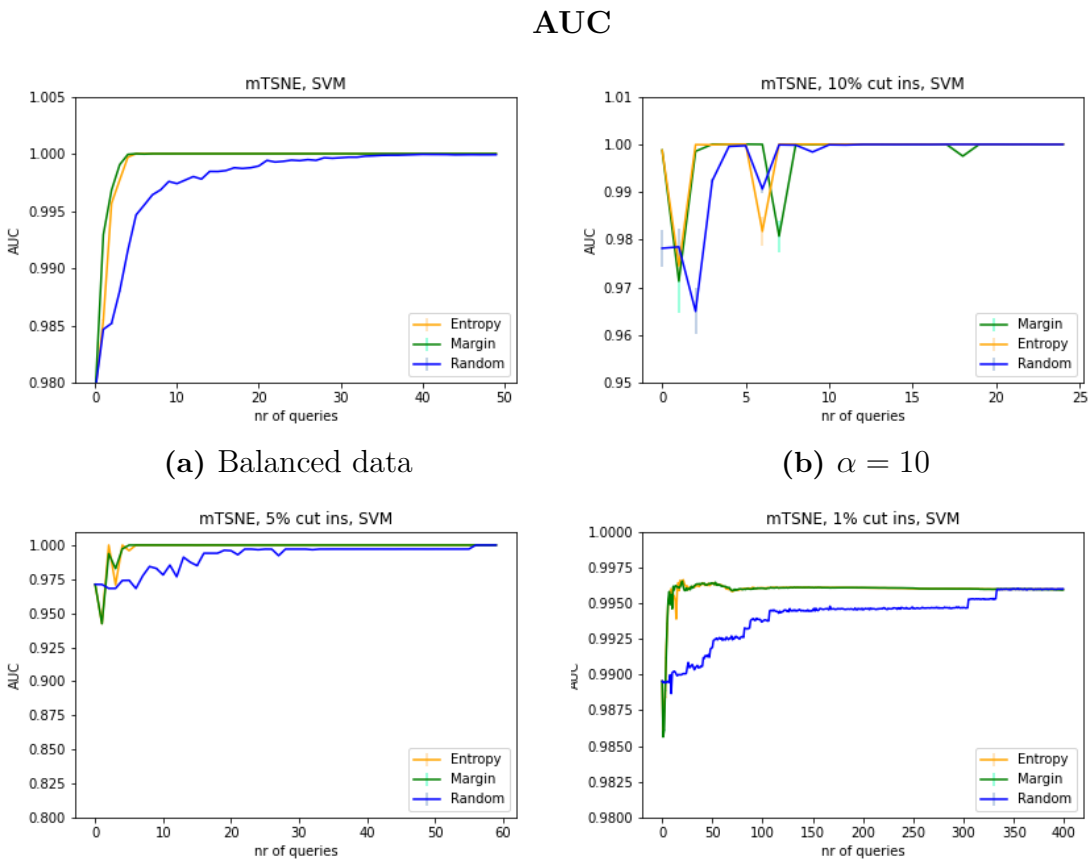
(a) Neural Network **(b)** SVM

Figure 4.2: F1 score for cut ins for different budget size with data embedding from mTSNE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.

As seen in Figures 4.1 and 4.2, margin and entropy gives a slightly higher AUC and F1 score as random using an NN classifier. Using the SVM classifier, margin and entropy achieve a high F1 score much faster than random. The jumps seen in Figure **??**, could be due to SVM configurations.

4.1.1 Class distribution

The impact of the class distribution was investigated by performing active learning on data with varying α, where α is the percent of cut ins in the data. $\alpha = 33, 10, 5$ and 1 was used. AUC for classification with SVM on the different distributions can be found in Figure 4.3. F1 score for cut ins for the different distributions can be found in Figure 4.4. The data plotted is an average over 10 runs, and the variance is represented by the lighter colored lines. The results for the Neural Network can be found in Appendix B in Figure B.1 and B.2.



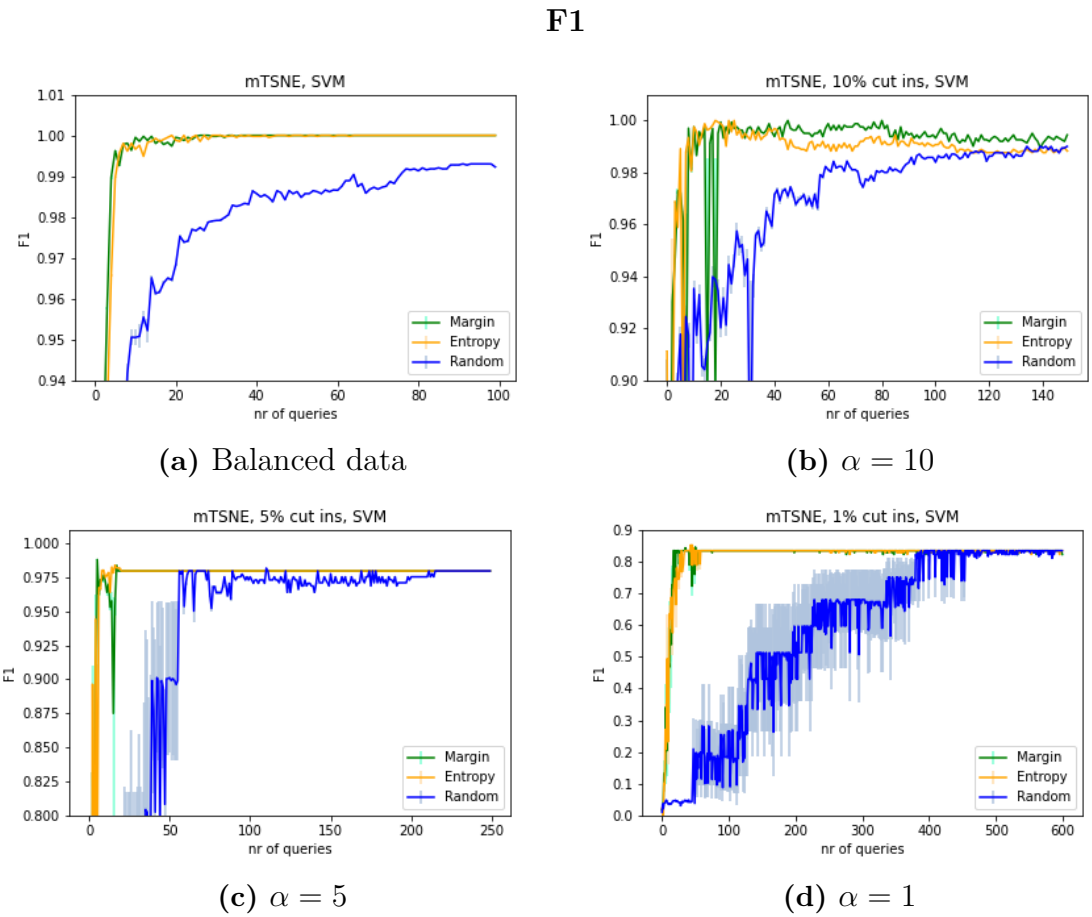
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure 4.3: AUC for different class distributions with data embedding from mT-SNE, using SVM as classifier. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

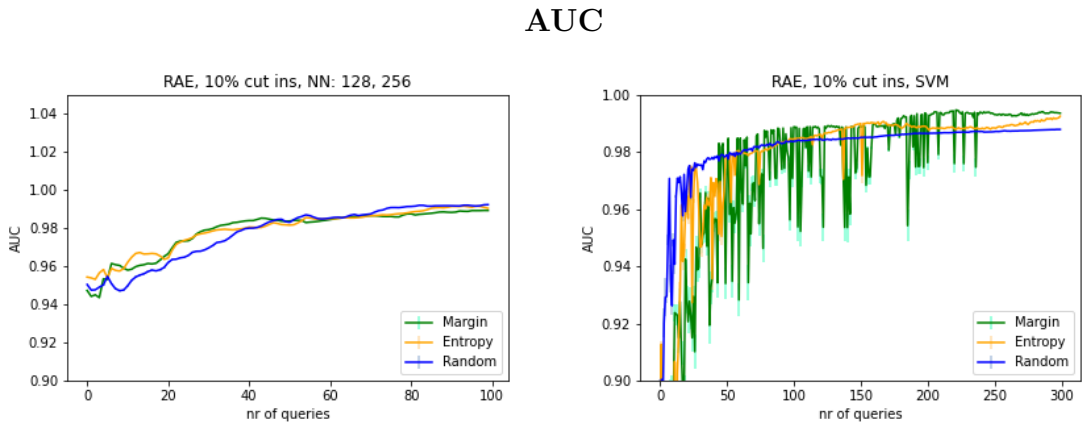
(d) $\alpha = 1$

Figure 4.4: F1 score for cut in for different class distributions with data embedding from mTSNE, using SVM as classifier. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data. Note the scale of the horizontal axes.

As shown in Figures 4.3 and 4.4, AUC and F1 score for margin and entropy saturates faster than random for all four data distributions. In Figure 4.4d, it is seen that for $\alpha = 1$, the F1 score saturates already at around 0.8.

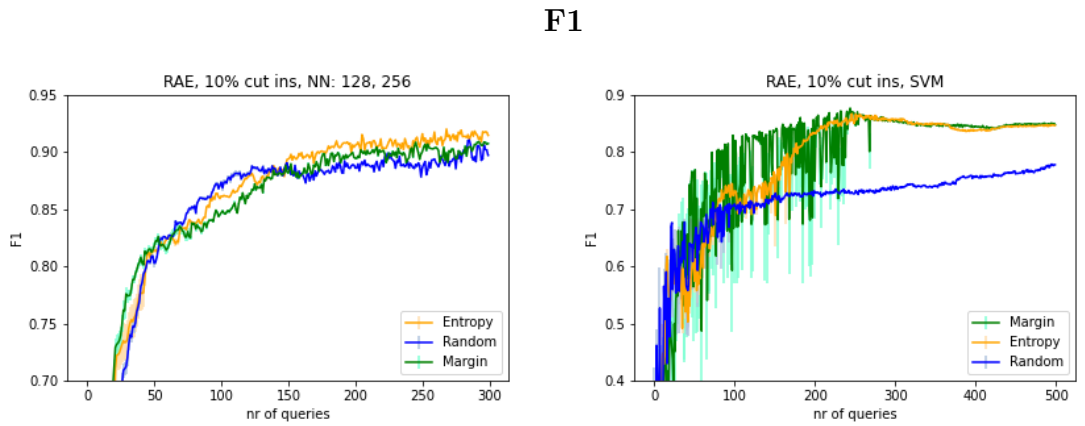
4.2 Recurrent Autoencoder

Two different classifiers were used: SVM and a Neural Network. The AUC for different number of queried points can be found in Figure 4.5. The F1 score for number of queried points can be found in Figure 4.6. The data plotted is an average over 10 runs, and the variance is represented by the lighter colored lines. The data consists of equal amount of left drive bys and right drive bys, and 10% cut ins.



(a) Neural Network **(b)** SVM

Figure 4.5: AUC for different budget size with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Neural Network **(b)** SVM

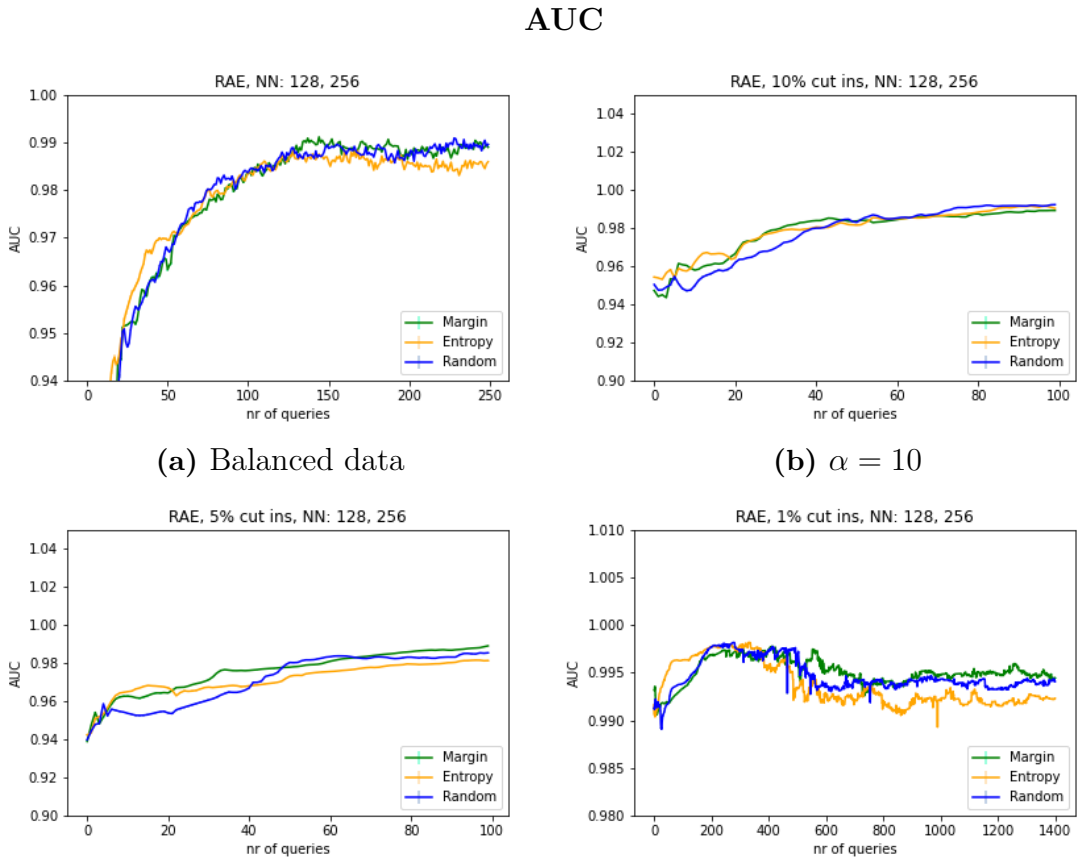
Figure 4.6: F1 score for cut ins for different budget size with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.

In Figures 4.5b and 4.6b, large fluctuations are observed using the SVM classifier. Looking at the NN classifier in Figures 4.5a and 4.6a, there is no significant difference in AUC or F1 score among the three query strategies.

4.2.1 Class distribution

The impact of the class distribution was investigated by performing active learning on data with varying α, where α is the percent of cut ins in the data. $\alpha = 33, 10, 5$ and 1 was used. AUC for classification with the Neural Network on the different distributions can be found in Figure 4.7. F1 score for cut ins for the different distributions can be found in Figure 4.8. The data plotted is an average over 10

runs, and the variance is represented by the lighter colored lines. The results for SVM can be found in Appendix B in Figures B.3 and B.4.



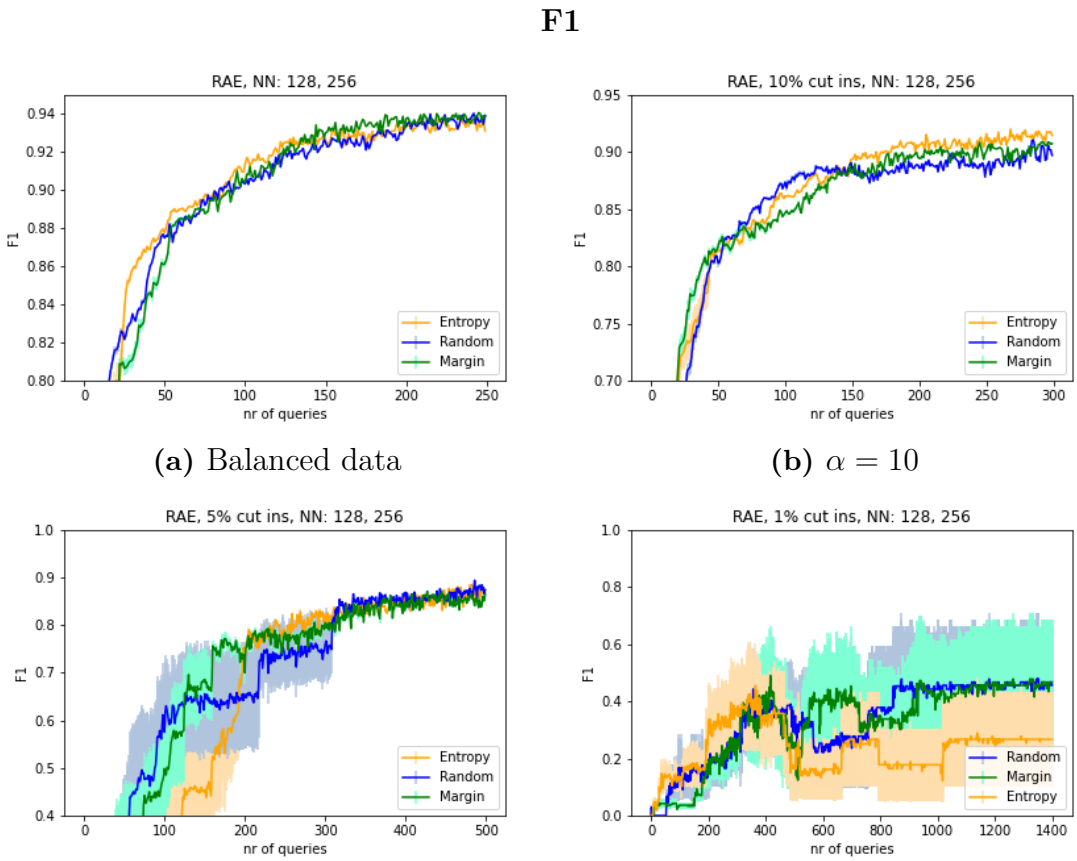
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure 4.7: AUC for different class distributions with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

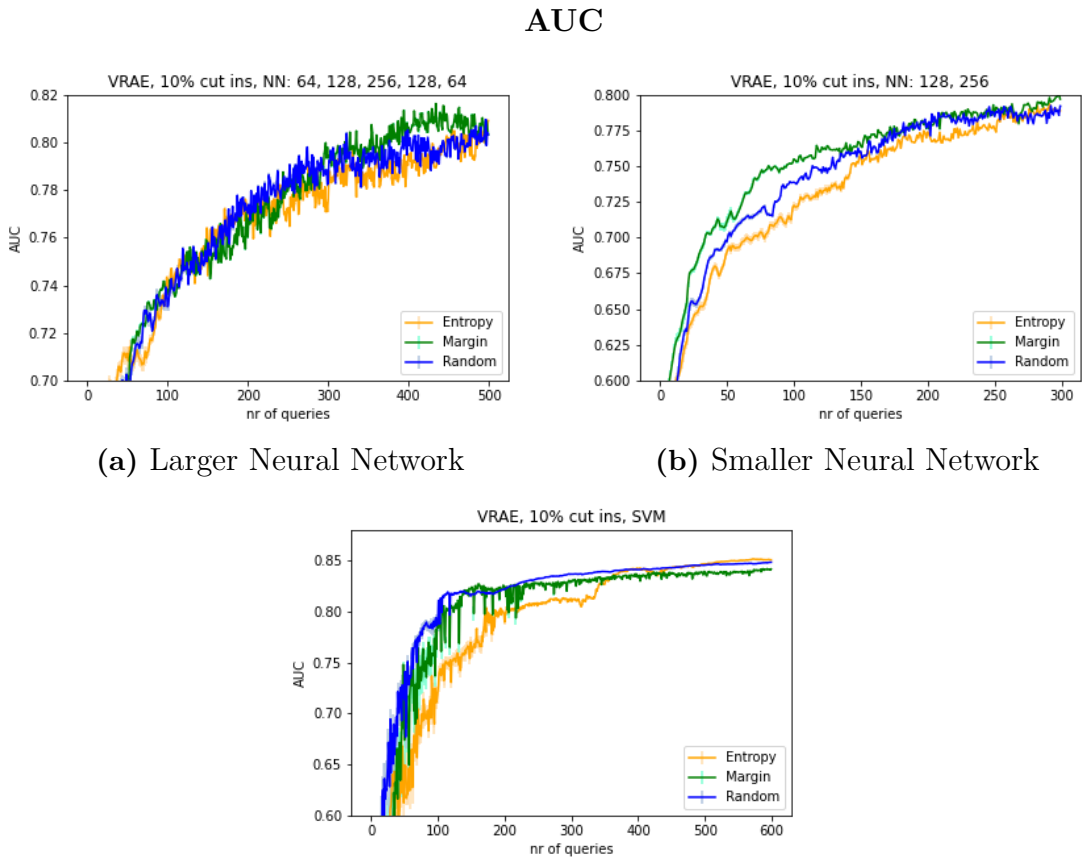
(d) $\alpha = 1$

Figure 4.8: F1 score for cut in for different class distributions with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data. Note the scale of the horizontal axes.

From Figures 4.7 and 4.8, it is shown that only balanced data and the data set with $\alpha = 10$ reaches an AUC and F1 score above 0.9. Overall no significant difference between the query strategies is observed, but in some cases either margin or entropy give a slightly better performance than random, as at least one of the two saturates faster or reaches a higher score.

4.3 Variational Recurrent Autoencoder

Three different classifiers were used: SVM, a Neural Network with five layers, and a Neural Network with two layers. The AUC for different number of queried points can be found in Figure 4.9. The F1 score for number of queried points can be found in Figure 4.10. The data plotted is an average over 10 runs, and the variance is represented by the lighter colored lines. The data consists of equal amount of left drive bys and right drive bys, and 10% cut ins.

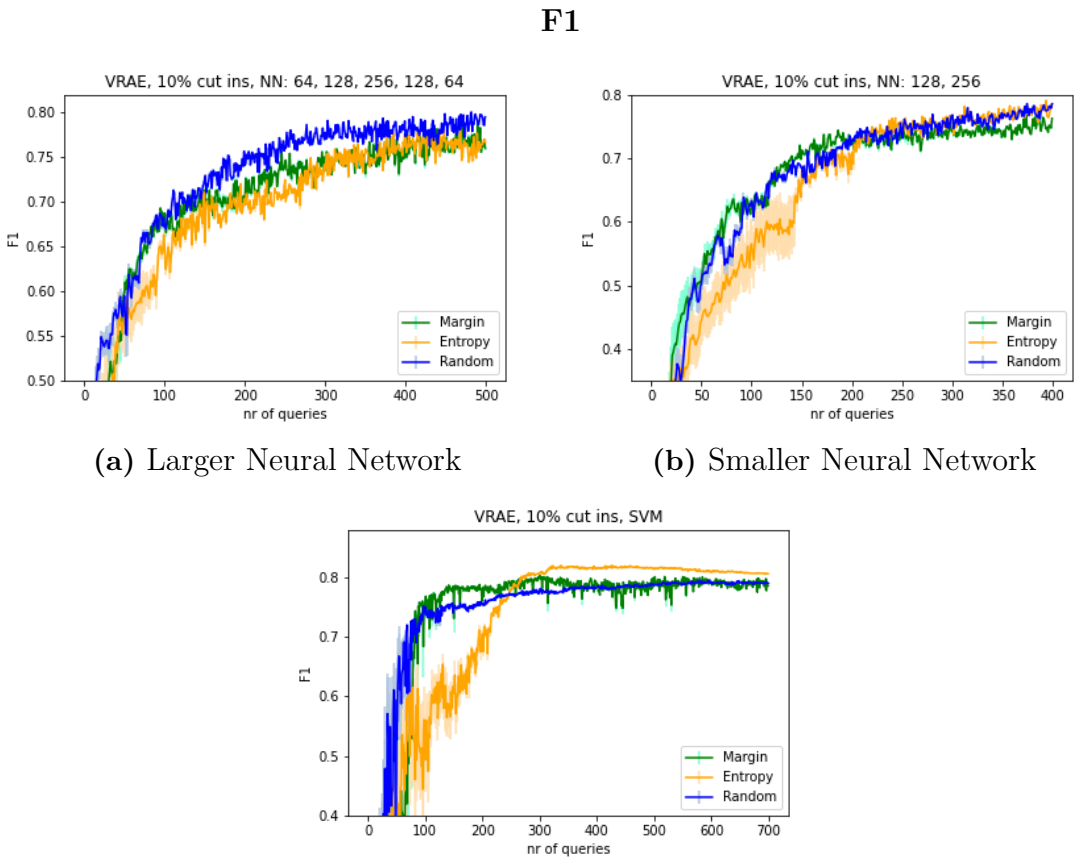


(a) Larger Neural Network

(b) Smaller Neural Network

(c) SVM

Figure 4.9: AUC for different classifiers with data embedding from VRAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Larger Neural Network

(b) Smaller Neural Network

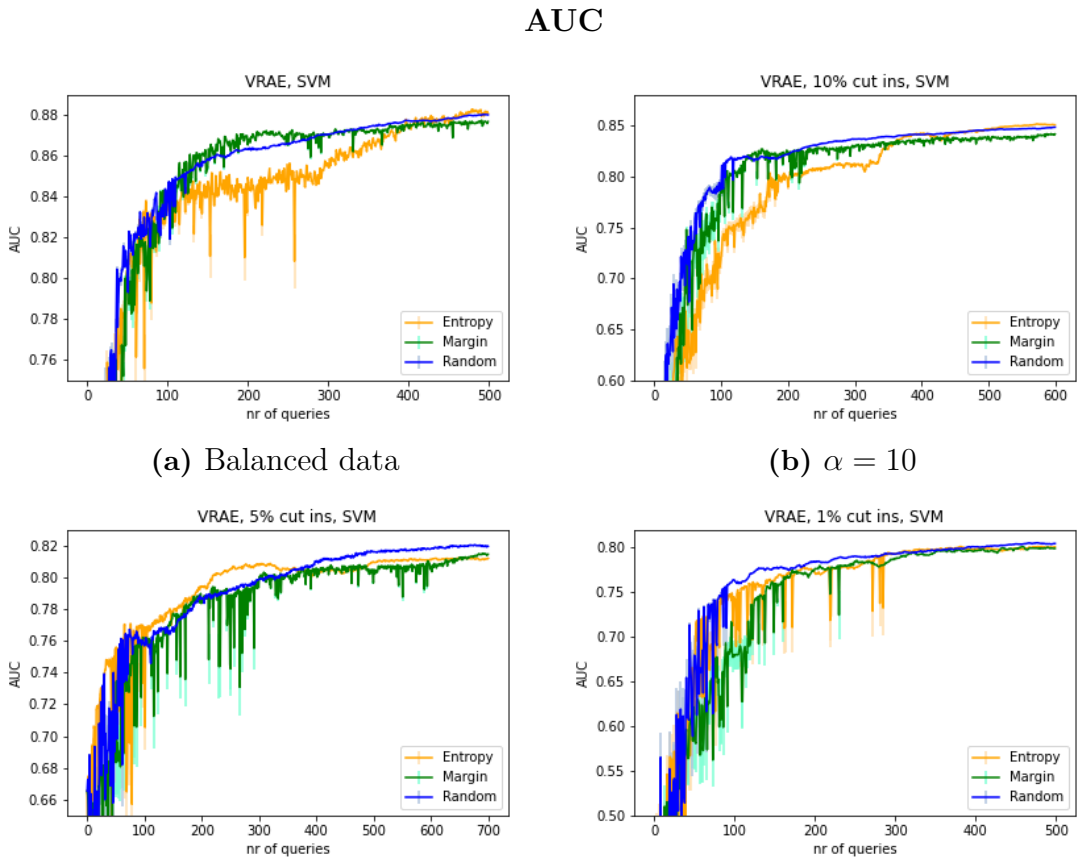
(c) SVM

Figure 4.10: F1 score for cut ins for different classifiers with data embedding from VRAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.

Figures 4.9 and 4.10 show that all classifiers obtain an AUC and F1 score of around 0.8. The only cases giving non-random query methods an advantage is using the two lager NN classifier shown in Figure 4.9b, where margin initially gives a higher AUC score.

4.3.1 Class distribution

The impact of the class distribution was investigated by performing active learning on data with varying α, where α is the percent of cut ins in the data. $\alpha = 33, 10, 5$ and 1 was used. AUC for classification with SVM on the different distributions can be found in Figure 4.11. F1 score for cut ins for the different distributions can be found in Figure 4.12. The data plotted is an average over 10 runs, and the variance is represented by the lighter colored lines. The results for the smaller network can be found in Appendix B in Figures B.5 and B.6.



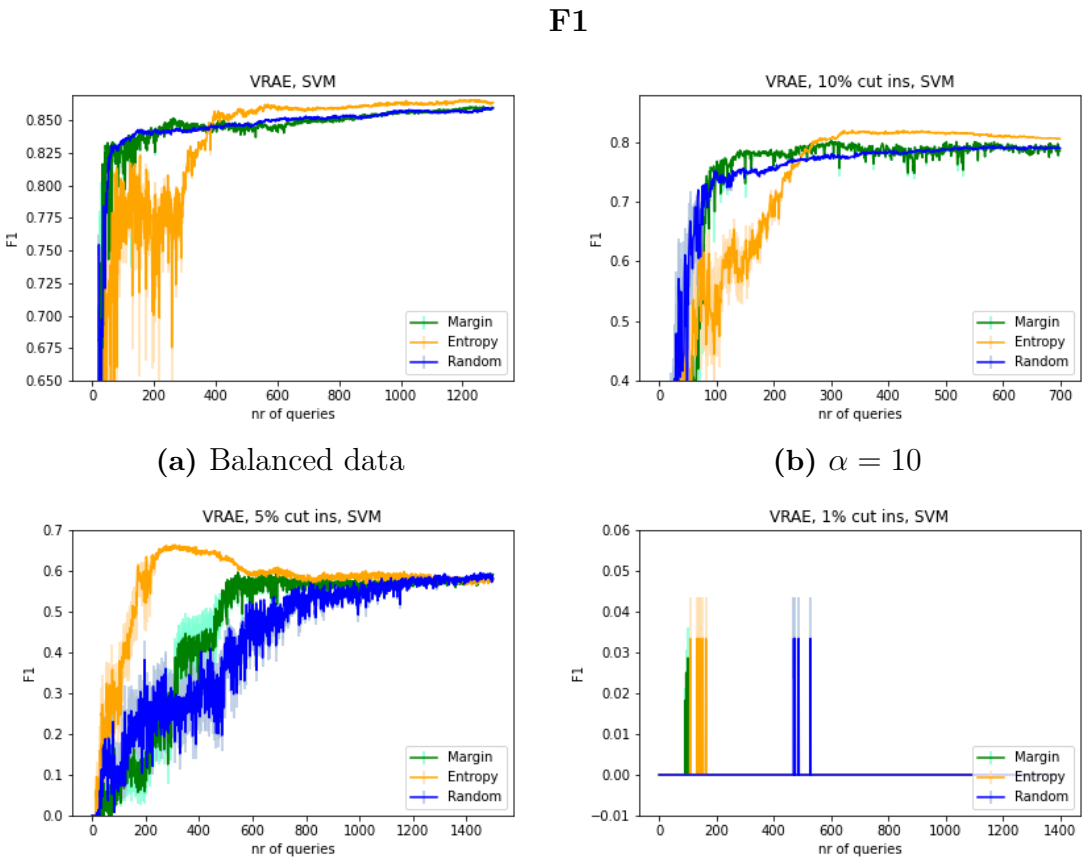
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure 4.11: AUC for different class distributions with data embedding from VRAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

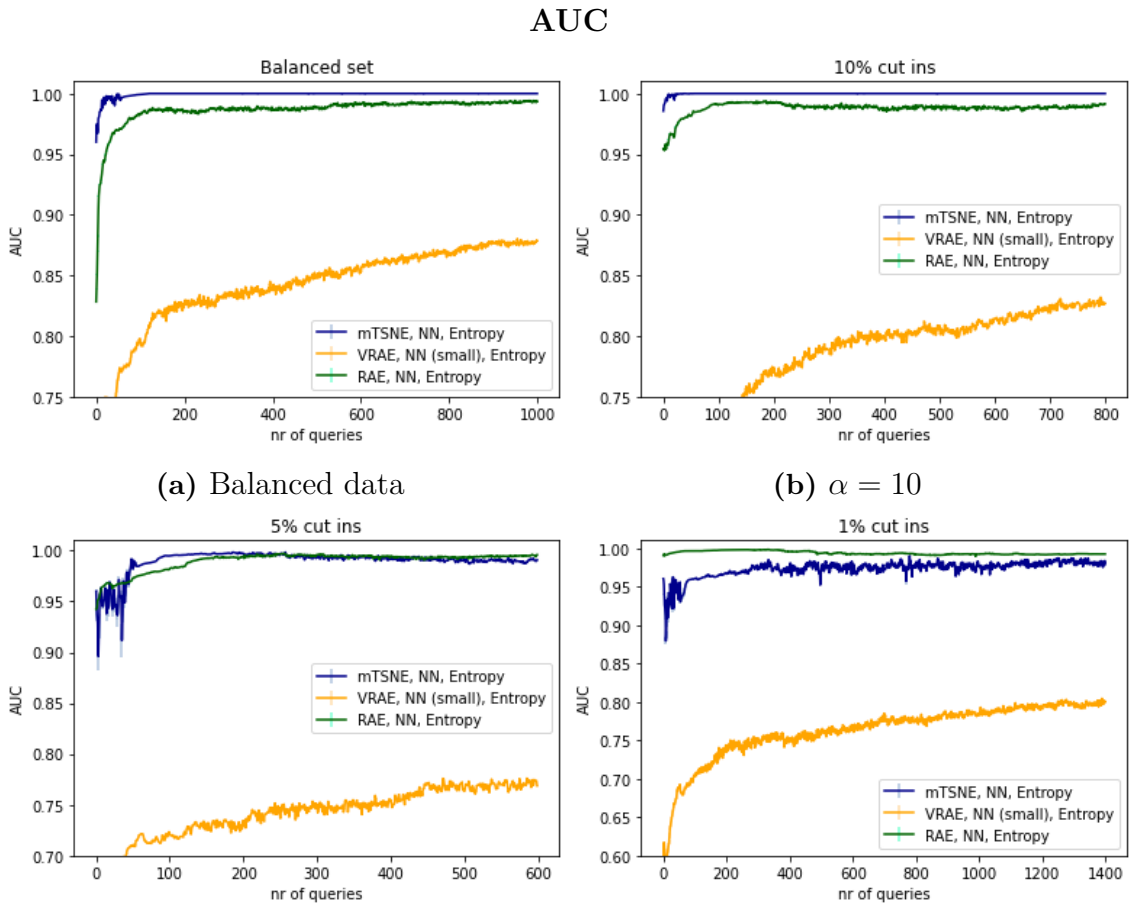
(d) $\alpha = 1$

Figure 4.12: F1 score for cut in for different class distributions with data embedding from VRAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.

As shown in Figures 4.11 and 4.12, only the balanced data set and the set with $\alpha = 10$ obtain an AUC and F1 score of at least 0.8. The two cases when a non-random query strategy shows a significant better performance is the F1 score for $\alpha = 10$ and $\alpha = 5$ using shown in Figures 4.12b respectively 4.12c, where margin respectively entropy saturates faster than the other query strategies.

4.4 Embedding

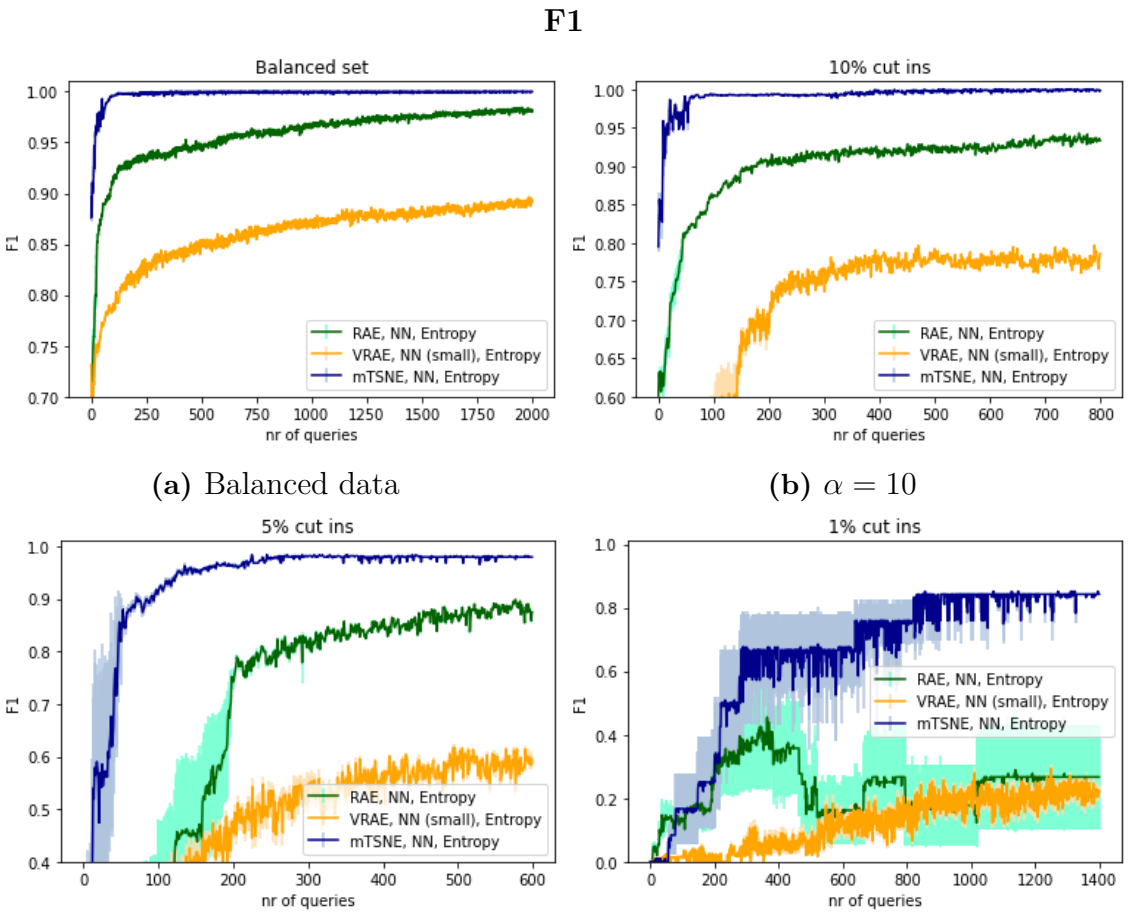
In this section results will be presented comparing performance of embeddings, using NN classifier and entropy query. The AUC and F1 score can be found in Figures 4.13 and 4.14.



(a) Balanced data **(b)** $\alpha = 10$

(c) $\alpha = 5$ **(d)** $\alpha = 1$

Figure 4.13: AUC for different class distributions comparing embeddings. Note the different scales. The data is an average of 10 runs and the lighter colored lines represent the variance in the data.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

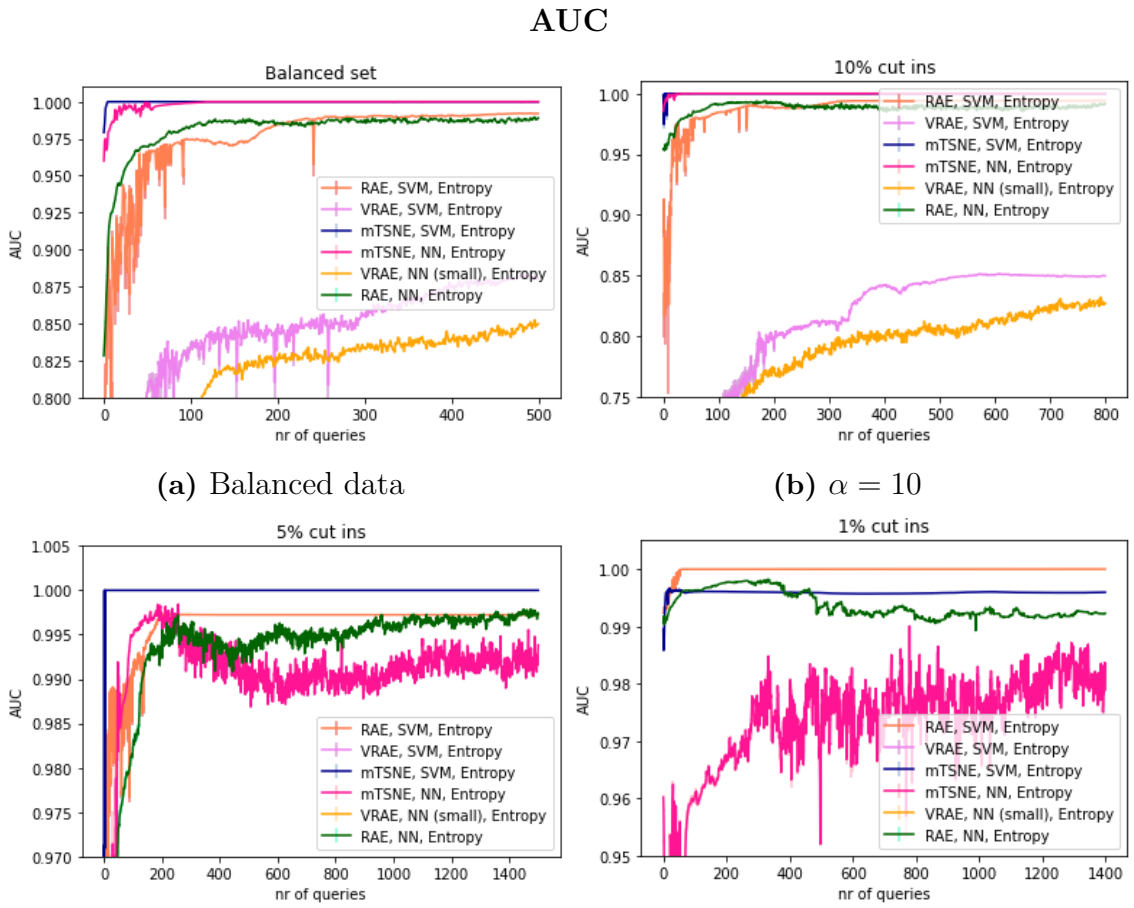
(d) $\alpha = 1$

Figure 4.14: F1 score for cut ins for different class distributions comparing embeddings. Note the different scales. The data is an average of 10 runs and the lighter colored lines represent the variance in the data.

Figures 4.13 and 4.14 show that mTSNE achieves an AUC and F1 score close to 1 on relatively few queries for balanced data, 10% and 5% cut ins. RAE also gives good results for balanced data and 10% cut ins, however for 5% it takes a long time before it reaches a high F1 score. mTSNE gives the overall best performance.

4.5 Classifier

In this section results will be presented comparing the performance of classifiers for each embedding using entropy query. The AUC and F1 score can be found in Figures 4.15 and 4.16.



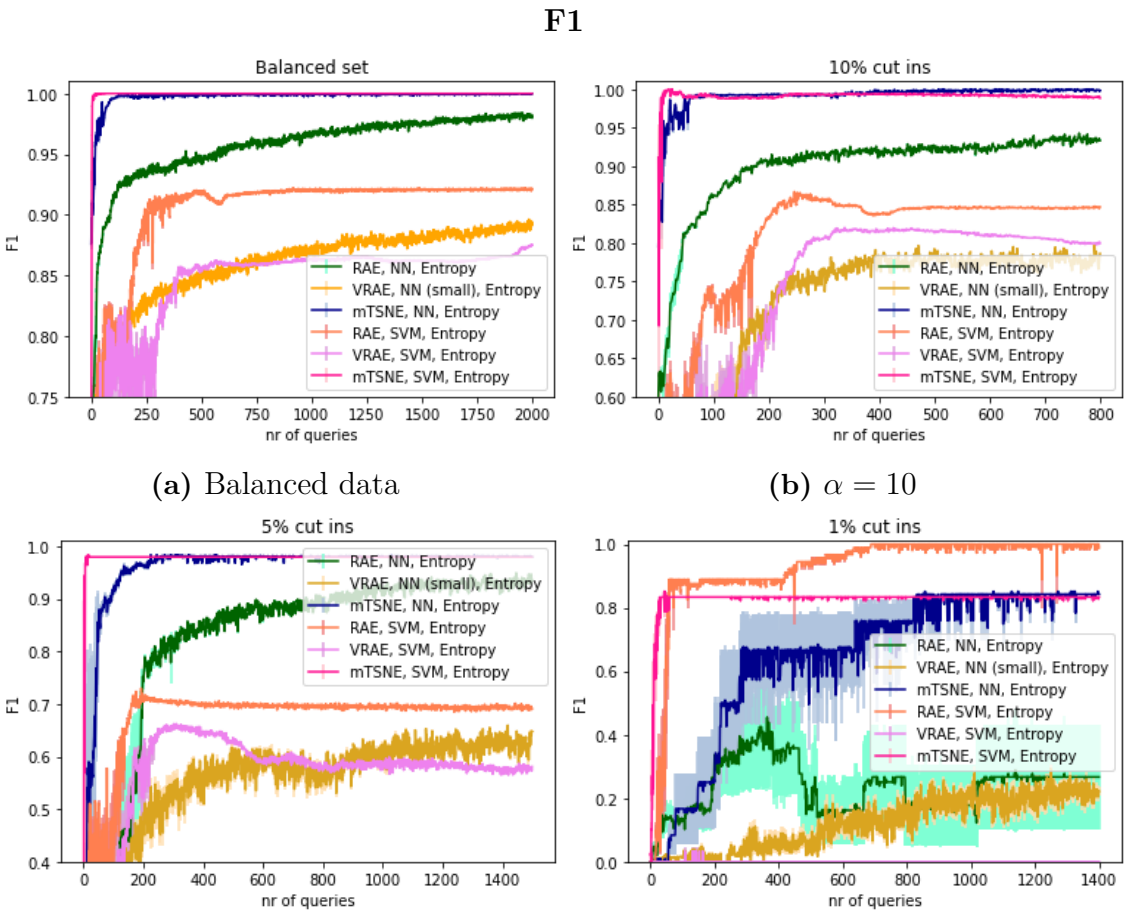
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure 4.15: AUC for different class distributions comparing classifiers. The data is an average of 10 runs and the lighter colored lines represent the variance in the data. Note the scales of the graphs, that in the top graphs, both implementations of mTSNE are not clearly visible due to the lines lying on top of each other. Also, in the bottom graphs, no implementations of VRAE are visible due to the scale of the vertical axis.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

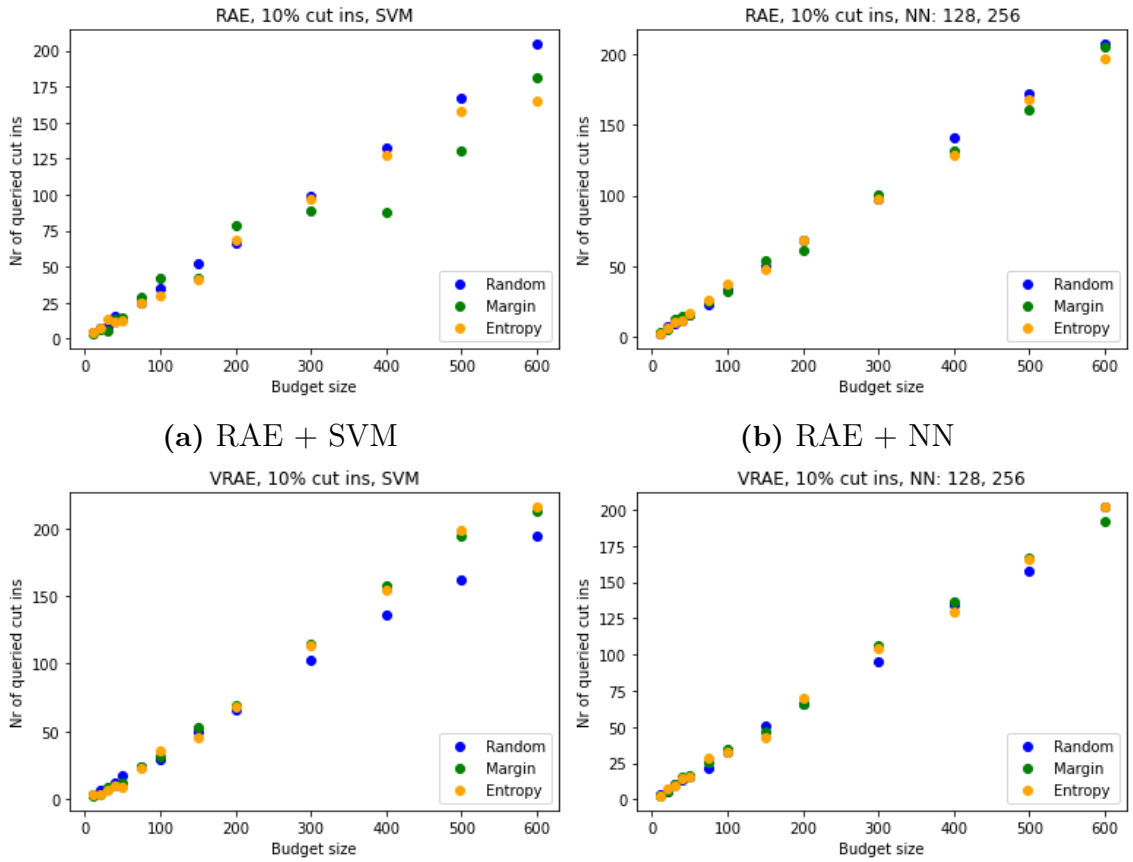
Figure 4.16: F1 score for cut ins for different class distributions comparing classifiers. Note the different scales. The data is an average of 10 runs and the lighter colored lines represent the variance in the data. Note the scales of the graphs.

In Figure 4.16, it is possible to see that SVM gives a better performance than NN for mSTNE, for example see Figure 4.16c. For RAE the NN consistently yields a higher F1 score than SVM, except for the case 1% cut ins.

4.6 Unknown class detection

Unknown class detection is carried out using RAE and VRAE embedded data, where cut in is considered as an unknown class. The data set containing 10% cut ins is used, applying an NN or SVM classifier. Figure 4.17 and Figure 4.18 show the number of queried cut ins for RAE and VRAE embedded data, using SVM or an NN classifier, where the latter is a zoomed in to show the 200 initial queries. For RAE combined with SVM in Figure 4.18a, before 200 queries either margin or entropy querying give best performance. However, after 200 queries both margin and entropy querying drop below random. Figure 4.18b illustrates VRAE with SVM classifier, all three query strategies show a linear behaviour up to 200 queries. After 200 queries margin and entropy querying query significantly more cut ins, see Figure

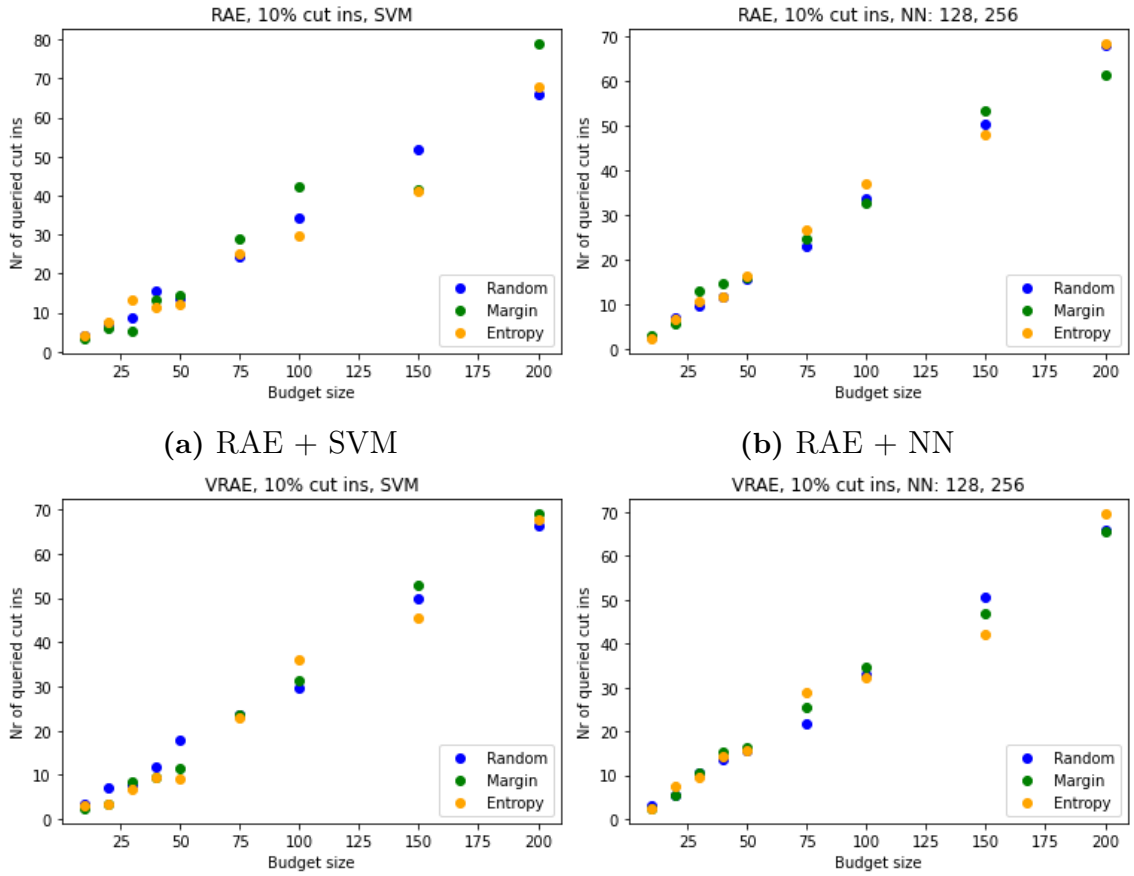
4.17a. RAE embedding with NN classifier is shown in Figure 4.17c. All three query strategies follow a linear trend. VRAE with NN showed in Figure 4.17d, margin and entropy query slightly more cut ins than random does after 200 cut ins.



(a) RAE + SVM **(b)** RAE + NN

(c) VRAE + SVM **(d)** VRAE + NN

Figure 4.17: Number of cut ins queried plotted against budget size.



(a) RAE + SVM

(b) RAE + NN

(c) VRAE + SVM

(d) VRAE + NN

Figure 4.18: Number of cut ins queried plotted against budget size.

5
Conclusion

Presented below are conclusions and a discussion regarding the study, as well as potential sources of error and directions for future work.

5.1 Discussion

Active learning for labelling trajectories has been studied using two different classifiers: a Neural Network and SVM. Three query strategies have been compared, entropy, margin and random as a reference. Unknown class detection has also been performed assuming cut in as the unknown class. For this purpose RAE and VRAE embeddings were used, and the different query strategies were compared in terms of number of queried cut ins. The data set used primarily in this study is a data set consisting 10% cut ins. AUC and F1 score have been used as evaluation metrics. Since the impact of different class distributions have been studied, AUC can be lacking in usefulness. The more unbalanced the classes are, the easier it is for the AUC to be high without learning to predict the smaller class at all. For this reason, F1 score is more useful, since it focuses on the smaller class and how well it is predicted.

5.1.1 Query strategy

From Figures 4.1 and 4.2 we can conclude that with SVM for $\alpha = 10$, both margin and entropy querying on mTSNE embedded data yield a higher AUC and F1 score faster than random. The effect is especially visible in the F1 score. For the NN, margin and entropy querying has a slight advantage on random in AUC. The jumps seen in Figure **??**, could be due to suboptimal SVM configurations, such as parameter values of c and γ. In Figure 4.4, it becomes obvious that both entropy and margin querying yield higher F1 scored than random querying for all α:s. The same trend can be seen in the AUC in Figure 4.3 as well, however, less distinct.

For RAE embedded data, we observe in Figures 4.5 and 4.6 that all query strategies are roughly equal in performance for the NN for $\alpha = 10$. For SVM, a positive effect of the active learning can be seen, as margin and entropy querying yield higher results than random querying. It is, however, worth noting that the baseline performance of margin querying is higher than that of random querying. As opposed to the quite stable margin and random querying, entropy querying is rather unstable, and the lows of the fluctuations frequently make it worse than random querying.

This is more noticeable in the AUC than in the F1 score. In Figure 4.8, no apparent difference can be seen in the query methods across class distributions, with a small exception for $\alpha = 5$. Random querying is slightly worse than the other two methods.

In Figure 4.9 and 4.10 only slight differences between query strategies in the VRAE embedded data with $\alpha = 10$ can be seen. For most cases, they are roughly equal, though entropy querying performs slightly worse across the board. This can be seen in Figures 4.11a and 4.12a for balanced data as well. In Figure 4.12c, for $\alpha = 5$, entropy querying shows a large positive deviation in F1 score from margin and random querying, however stabilises at the same score. No query method works for SVM with $\alpha = 1$, as seen in Figure 4.12d.

A general trend observed for mTSNE and RAE embedded data is that SVM in combination with margin query tend to show larger fluctuations, but with a stable baseline. This could be explained by that margin, as the name indicates, queries the point with the smallest margin. This means that SVM is more sensitive for new points queried by margin, as the separating hyper plane can change direction rapidly.

5.1.2 Embedding

As evident in Figures 4.13 and 4.14, mTSNE outperforms RAE and VRAE in both AUC and F1 score for all values of α. The exception is for $\alpha = 1$, where RAE performs better in AUC. Note that AUC is not a very reliable metric for unbalanced data. Note also that RAE is also performing well, while VRAE is significantly worse. This result is found when observing the top performing implementation (query strategy combined with classifier) of each embedding.

General for the mTSNE and the RAE embedding is that they both achieve a high AUC and F1 score using either classifiers. They yield an AUC close to 1 for $\alpha = 10$, see Figures 4.1 and 4.5. However as seen in Figures 4.2 and 4.6, the F1 score is slightly worse for the RAE embedding compared to mTSNE, and RAE requires more queries to achieve a satisfactory F1 score.

The VRAE embedding gives a significantly worse performance than both mTSNE and RAE embedded data do. The AUC and F1 score for all classifiers with $\alpha = 10$ saturates around 0.8, see Figures 4.9 and 4.10. The difference between the query strategies is small, although it is observable that entropy performs worse in all cases, especially for SVM. A possible explanation for the poorer performance of the VRAE embedding compared to the others, lies in the nature of VRAE. The purpose of the VRAE is to create a cohesive latent space that have regions with different classes, and gradients of the classes in between. This means that the areas between the class clusters are a combination of the classes in different ways. This is by design difficult to classify. Margin querying queries points that have a high likelihood to belong to two different classes, and entropy querying queries points that have the highest uncertainty of belonging to it's most certain class. This would indicate that both

margin and entropy querying query the points that fall between the clusters. These points are inherently difficult to classify, and might therefore not contribute more than a randomly queried point to the model.

Worth noting is the computational power required for the embeddings. Producing an mTSNE embedding takes a long time, due to the demanding computations performed. It is most suitable for a smaller data set, or a larger set if one has access to a cluster and can parallelize the computation. Due to the stochastic nature of the algorithm, the latent space will be different for each transformation, so it is not possible to add points to an already transformed data set. In the case of the autoencoders, the training time is not excessively high, and the transformation is fast. On top of that, the latent space is the same each time, so points can easily be added afterwards.

5.1.3 Classifier

Observe that in Figure 4.15 that for mTSNE embedded data, both classifiers perform very well. However, in Figure 4.16 it becomes apparent that SVM consistently performs better than the NN. Due to the separability in the mTSNE embedding, it is reasonable that a lower capacity model is sufficient. Since SVM is faster and simpler, it is preferred to using an NN classifier where possible.

The opposite trend can be observed in RAE, where the NN is achieving higher and less fluctuating F1 scores for all cases, with $\alpha = 1$ as the exception. Here SVM outperforms the NN by a wide margin. The AUC follows a similar pattern, but with a smaller discrepancy between SVM and the NN for $\alpha = 1$. The conclusion is that SVM is more prone to fluctuations in the performance, probably due to high sensitivity to the new queried point.

For VRAE, SVM yields higher AUC than the NN for all α:s. We can note, however, that for F1 score for $\alpha = 10$ and 5, SVM increased faster than the NN, but decreases after a while. For the other class distributions, the NN performs better. It seems SVM finds the trends in the data quicker, but the NN can learn a higher complexity over time. Since VRAE embedded data yielded an overall worse results than the other embeddings, a larger Neural Network consisting of five hidden layers with 64, 128, 256, 128 and 64 neurons was tested. Despite increased capacity the performance does not improve, see Figure 4.9. Although the larger NN was expected to improve performance, there are several factors that could contribute to stagnation at 0.8, such as the nature of VRAE and training configurations. Again, SVM is more noisy compared to the NN for the VRAE embedding.

It is notable that the behaviour of the two classifiers are often distinct and different. SVM quickly finds the large trend in the data, and after that does not continue learning. The NN on the other hand, learns a bit slower and therefore has a less defined elbow, but continues learning with more data. This can probably be explained

by the low capacity of the SVM and high capacity of the NN. SVM is a more general model that fits to the large trends of the data, while the NN is more flexible and can adapt more to difficult data. For the mTSNE embedding the SVM is highly suitable due to the simplicity and separability of the data. It is plausible that SVM performs better than the NN for the VRAE embedding due to the cohesive latent space that would allow SVM to capture large trends. For the RAE embedding, however, the latent space is likely too complex to be fully captured by a lower complexity model.

5.1.4 Budget size

The budget size is a highly relevant factor in assessing performance in the case of active learning. With a very low budget, mTSNE embedded data can achieve a very high performance. For $\alpha = 10$ with SVM, F1 score for entropy and margin querying elbow at roughly 15 queries, though margin querying needs about 20 queries to stabilize, as seen in Figure 4.4. The same elbow effect can be seen in balanced data, though without the fluctuations, and even fewer queries. For $\alpha = 5$ and 1, roughly 25 and 60 queries respectively is needed to find stability in F1 score. The NN is slightly slower, though plateaus at about 60 queries, as seen in Figure 4.2.

In Figure 4.6 for the RAE embedding, note that all query strategies elbow at approximately 125 queries for the NN. For SVM, nearly 300 queries are required to reach stability. In Figure 4.8, an elbow can be found in the data at about 125 and 200 queries for $\alpha = 33$ and 5 respectively.

In Figure 4.10, there is an elbow at F1 = 0.8 in the graphs at 200, 150 and 100 queries for the large network, the small network and SVM respectively. The same trend can be observed for balanced data in Figure 4.12. For $\alpha \leq 5$, the performance is poor independent of the number of queries.

5.1.5 Class distribution

According to Figures 4.3 and 4.7, mTSNE and RAE embedding on all the data sets achieve an AUC score close to 1. The F1 score is consistent with the AUC for all data sets with the mTSNE embedding, except for the one with $\alpha = 1$. This set achieves an F1 score of about 0.8. Due to the high class imbalance, there are only very few cut ins in the data set, and a small number of misclassified points can lead to a drop in F1. It would be interesting to investigate whether or not these points are correctly classified in the ground truth. Although the RAE embedding yields similar AUC scores to mTSNE embedded data, the F1 scores are lower for all class distributions. It still achieves a rather high score, except for the case of $\alpha = 1$, where no query strategy exceeds F1 = 0.5. From Figure B.4, it is evident that the RAE embedding performs quite well with SVM for high imbalance in the data, with both high AUC and F1 score. Regarding VRAE, only the balanced data set obtains an AUC and F1 score above 0.8.

The conclusion is that only some implementations yield high results for the data with $\alpha = 1$. mTSNE is highly apt for the task independent of classifier, and RAE can do it when combined with SVM. Similarily, the aforementioned combinations perform well for $\alpha = 5$ as well. The RAE embedding combined with the NN does not perform as well. Neither does any classifier with the VRAE embedding. Another conclusion is that the performance differs very little between balanced data and data with $\alpha = 10$.

5.1.6 Unknown class detection

Comparing RAE and VRAE embedded data in Figure 4.17, it is noticeable that the patterns are reversed. For RAE after 200 queries, margin and entropy querying fall below random for both classifiers, while for VRAE margin and entropy querying query significantly more cut ins than random querying does after 200 queries. Looking at the number of queried cut ins during the 200 initial cut ins in Figure 4.18, margin and entropy querying query more cut ins for RAE, while for VRAE the opposite. Similar to discussed earlier, since VRAE produces a more coherent latent space, it becomes easier for SVM to find a suitable separating hyper plane and thus VRAE with SVM performs better than using an NN, see Figure 4.17a and 4.17d. Due to that RAE could have a more complex latent space in combination with the lower capacity of SVM, this could explain why SVM only works well for less than 200 queries, and then for learning more complex features an NN is needed, even though it takes more queries to learn.

The number of queried cut ins tends to fluctuate as seen in Figure 4.18, a possible cause is that not enough runs has been done in order to obtain a stable average.

5.1.7 Sources of error

A possible source of error is the accuracy of the annotation tool used for the ground truth labels. In case of faulty ground truth labels, the false positive and false negative rates will increase and a lower accuracy is obtained. Another source of error could be the quality of data. Some of the trajectories when plotted do not look like like any of the three classes cut in, left drive by or right drive by. This might have an effect on fitting the model.

In order to handle different class distribution, the removal of cut ins led to that the 10 points used for fitting the model varies between the different distributions. This might as well impact how the model learns.

To verify the results an average were taken over several runs, however no study on how many runs that is necessary has been conducted.

5.2 Future work

This project has dealt with the basics of active learning and unknown detection. possible future work is to further extend the model to fit more complex critera. For instance the generative-discriminative model developed by [23], can optimize with two different criteria during the learning process. In the case of combining gaussian mixture model (GMM) and SVM, it is possible to use the likelihood and uncertainty criteria respectively. This approach could possibly be beneficial to explore rare or even unknown classes. Another suggestion of further work is to investigate different parameters of the NN, such as optimizer, loss function, activation functions and weights. Studies [24] [25] have shown that generalization can be improved by density weighting, which ensures the queried points to be both informative and representative.

Due to limited resources, unknown class detection was only briefly investigated and trajectory generation was not part of the scope. To further develop the active learning algorithm for unknown detection could be to add a new class along the way, without reinitializing the model. Looking at the reconstruction error from either RAE or VRAE, unknown or rare classes could be found. The work by [4] uses RC-GAN and RAE-GAN structures for scenario trajectory generation. Trajectories could also be generated using the VRAE developed in this project.

5.3 Conclusion

In this study, we have investigated the performance of active learning as an effective tool for reliable and cost-efficient labelling of time series trajectory data. We have also explored its uses as an unknown class detector.

The conclusion is that for many implementations, active learning is an effective tool. The effect is particularly defined when using SVM as a classifier, both for mTSNE and RAE. For mTSNE embedded data, entropy querying with SVM can be seen as the best option, due to high performance on few queries, and a higher stability than margin querying. The high performance of AUC and F1 ≈ 1 can be reached within 25 queries for most higher values of α, and within 60 for $\alpha = 1$. Unfortunately, mTSNE has drawbacks regarding the details of the embedding process, making RAE an attractive option. Sacrificing some accuracy compared to mTSNE, RAE is still high performing and more practical to use.

For the RAE embedding, the NN performed better than SVM, and for the NN, no significant difference in performance between the query strategies can be observed. For balanced data and $\alpha = 10$ the graph elbows at F1 ≥ 0.9 after approximately 125 queries. An F1 score of 0.8 is achieved after 150 queries for $\alpha = 5$. Combining the RAE embedding with SVM yields an F1 score of 0.9 after < 100 queries for $\alpha = 1$.

The VRAE embedding did not achieve a particularly high performance on either

classifier. The query strategies performed roughly equally, with the exception of entropy querying, which consistently under-performed. The performance of this embedding plateaus at about 0.8 after around 150 queries.

Regarding anomaly detection, it can be concluded that VRAE embedding using SVM classifier with either entropy or margin, queries the most number of cut ins for a larger budget size. For a budget size up to 200, RAE combined with SVM queries the most cut ins using margin.

Bibliography

[1] NounProject, "car top view." [Online]. Available: https://thenounproject.com/term/car-top-view/561980/

[2] A. Ajanki, "File:svm margins.png," June 2006. [Online]. Available: https://en.wikipedia.org/w/index.php?title=File:SVM_margins.png

[3] J. Bedford, "File:python roc curve.png," October 2014. [Online]. Available: https://commons.wikimedia.org/wiki/File:Python_roc_curve.png

[4] A. Demetriou, H. Alfsvåg, S. Rahrovani, and M. H. Chehreghani, "A deep learning framework for generation and analysis of driving scenario trajectories," *arXiv preprint arXiv:2007.14524*, 2020.

[5] G. Chevalier, "LARNN: linear attention recurrent neural network," *CoRR*, vol. abs/1808.05578, 2018. [Online]. Available: http://arxiv.org/abs/1808.05578

[6] Trasportstyrelsen, "Omkomna januari–november 2010–2020," December 2020.

[7] WHO, "Global status report on road safety 2018," December 2018.

[8] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[9] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[10] J. D. Bossér, E. Sörstadius, and M. H. Chehreghani, "Model-centric and data-centric aspects of active learning for neural network models," *arXiv preprint arXiv:2009.10835*, 2020.

[11] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," 2004.

[12] S. R. Karur and P. Ramachandran, "Radial basis function approximation in the dual reciprocity method," *Mathematical and Computer Modelling*, vol. 20, no. 7, pp. 59–70, 1994.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[14] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[15] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *arXiv preprint arXiv:1805.07836*, 2018.

[16] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *CoRR*, vol. abs/2010.16061, 2020. [Online]. Available: https://arxiv.org/abs/2010.16061

[17] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[19] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014. [Online]. Available: http://arxiv.org/abs/1402.1128

[20] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," 2016.

[21] O. Fabius and J. R. van Amersfoort, "Variational recurrent auto-encoders," 2015.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] T. M. Hospedales, S. Gong, and T. Xiang, "Finding rare classes: Active learning with generative and discriminative models," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 2, pp. 374–386, 2011.

[24] M. Wang, X.-S. Hua, Y. Song, J. Tang, and L.-R. Dai, "Multi-concept multimodality active learning for interactive video annotation," in *International Conference on Semantic Computing (ICSC 2007)*. IEEE, 2007, pp. 321–328.

[25] P. Donmez, J. G. Carbonell, and P. N. Bennett, "Dual strategy active learning," in *European Conference on Machine Learning*. Springer, 2007, pp. 116–127.

A

mTSNE embedding

The perplexity used to obtain the latent space representation of the data was set to 37.5. Figures A.1, A.2 and A.3 show the latent space representation generated using mTSNE on data sets with $\alpha = 33, 5$ and 1. For all three representations it is possible to separate the three classes. The red, white and blue dots correspond to cut in, right and left drive by.

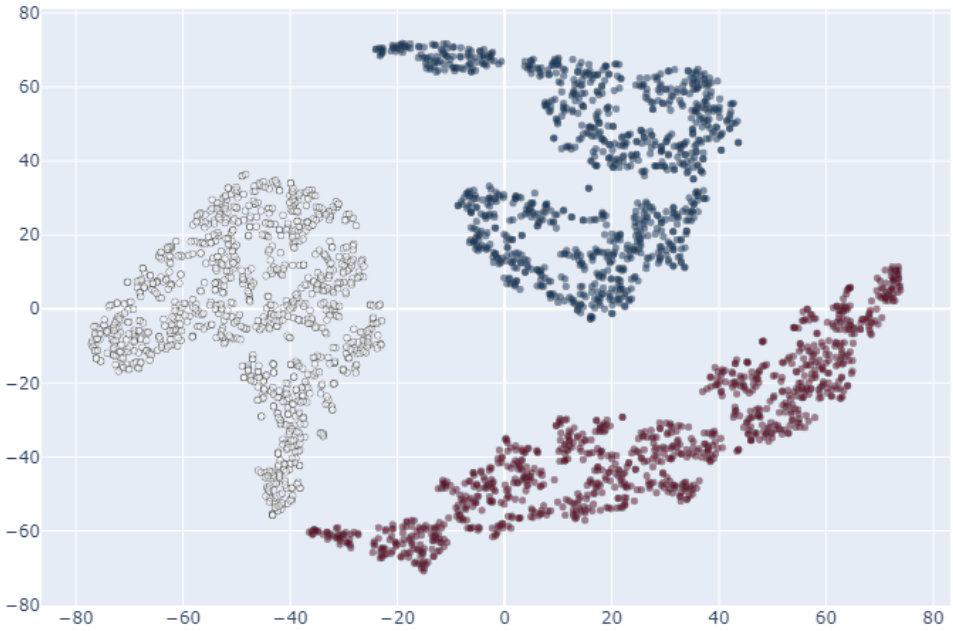


Figure A.1: mTSNE embedding of a data set having equal number of points in all three classes.

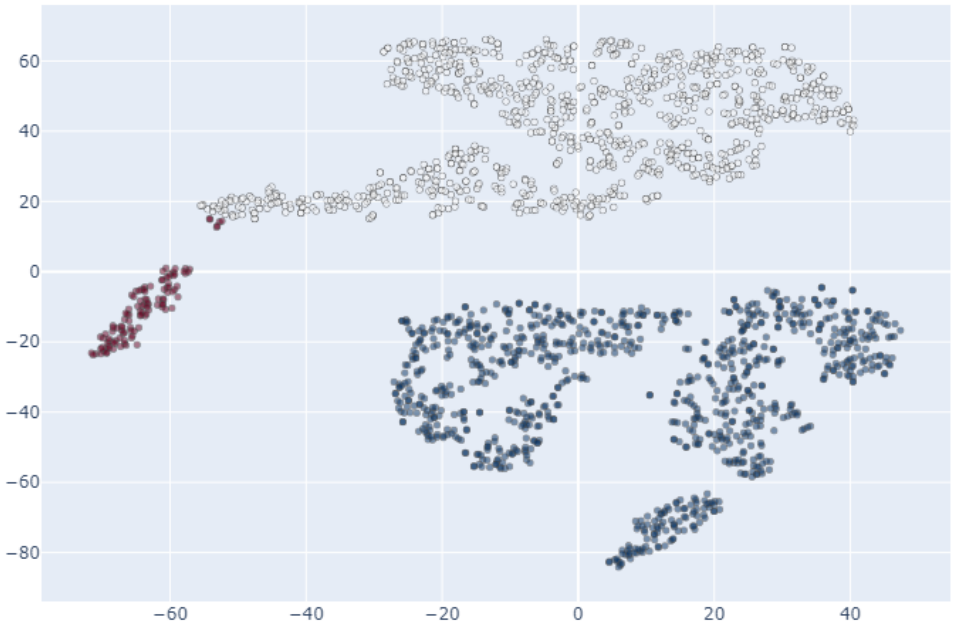


Figure A.2: The clusters formed in the latent space having 5% cut ins. Its possible to linearly separate the three classes.

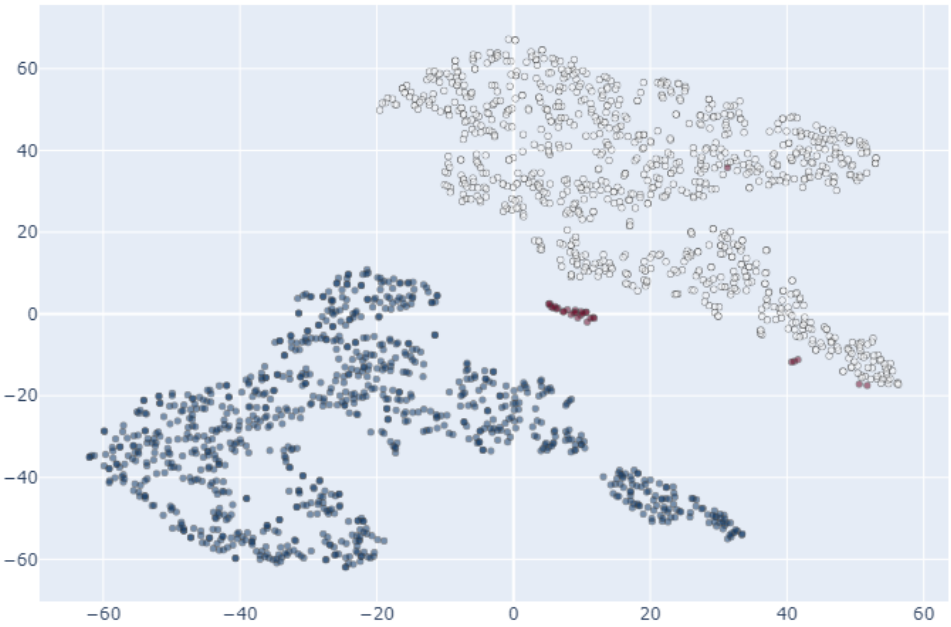
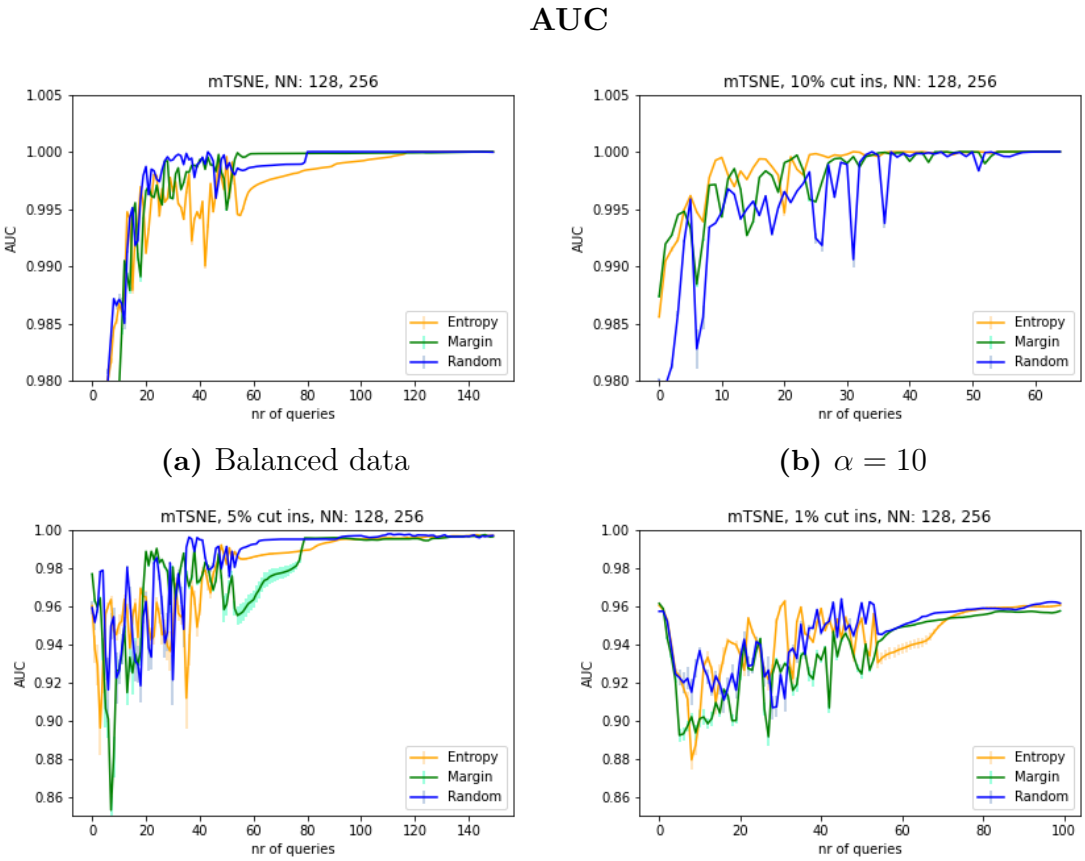


Figure A.3: The clusters formed in the latent space having 1% cut ins. Note the relative separability of the three classes.

B

Further results on class distribution

In this section some further results of the three embeddings are presented. AUC and F1 score for mTSNE can be found in Figures B.1 and B.2. AUC and F1 score for RAE can be found in Figures B.3 and B.4. AUC and F1 score for VRAE can be found in Figures B.5 and B.6.



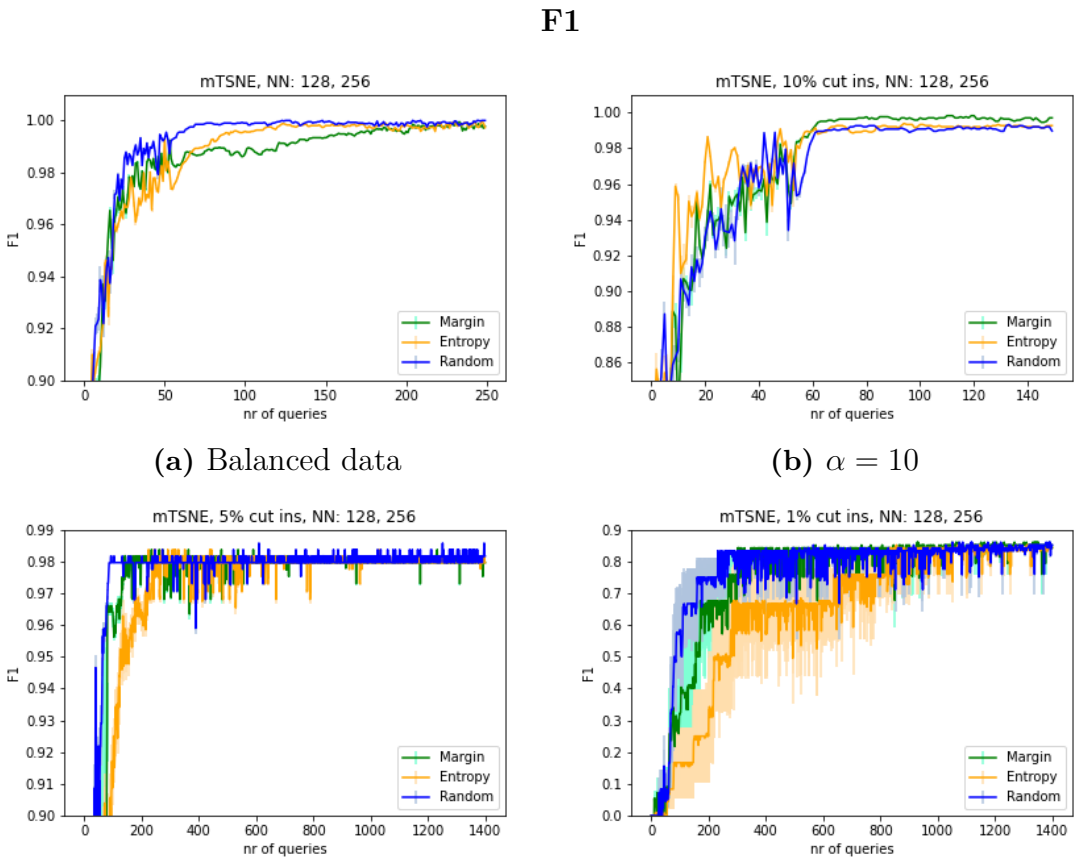
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.1: AUC for different class distributions with data embedding from mtsne. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



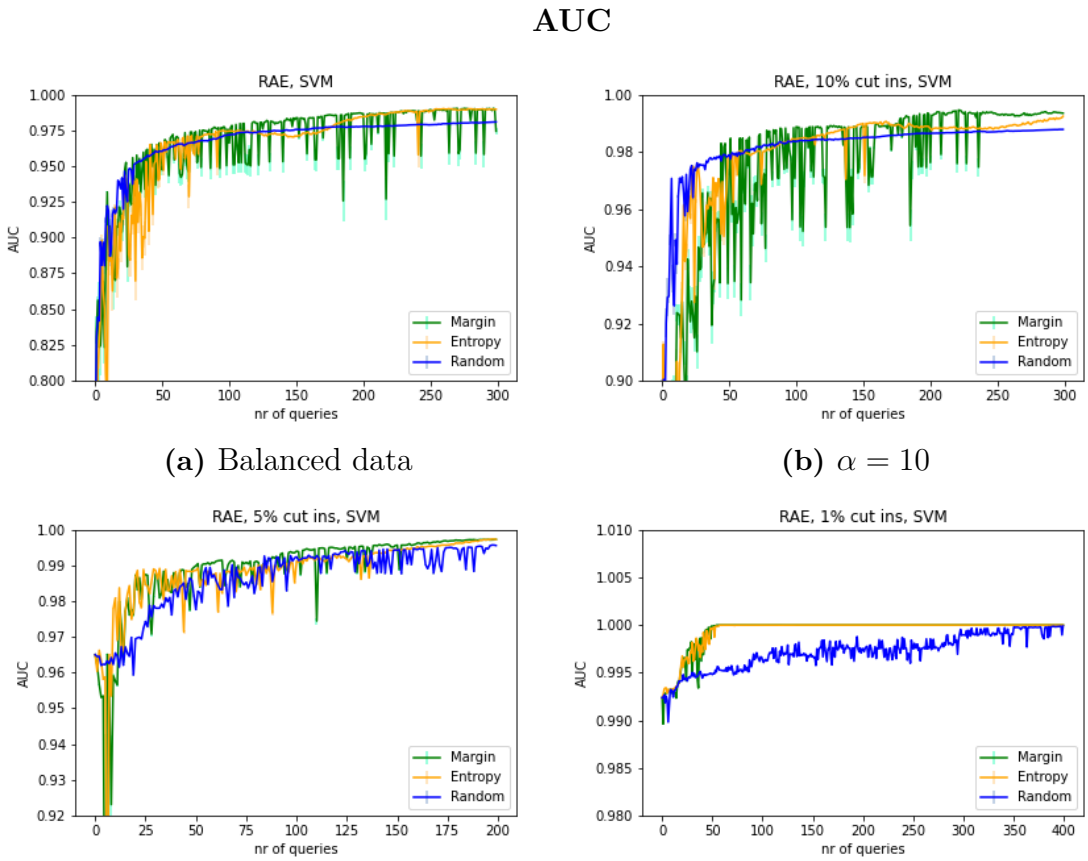
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.2: F1 score for cut in for different class distributions with data embedding from mTSNE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



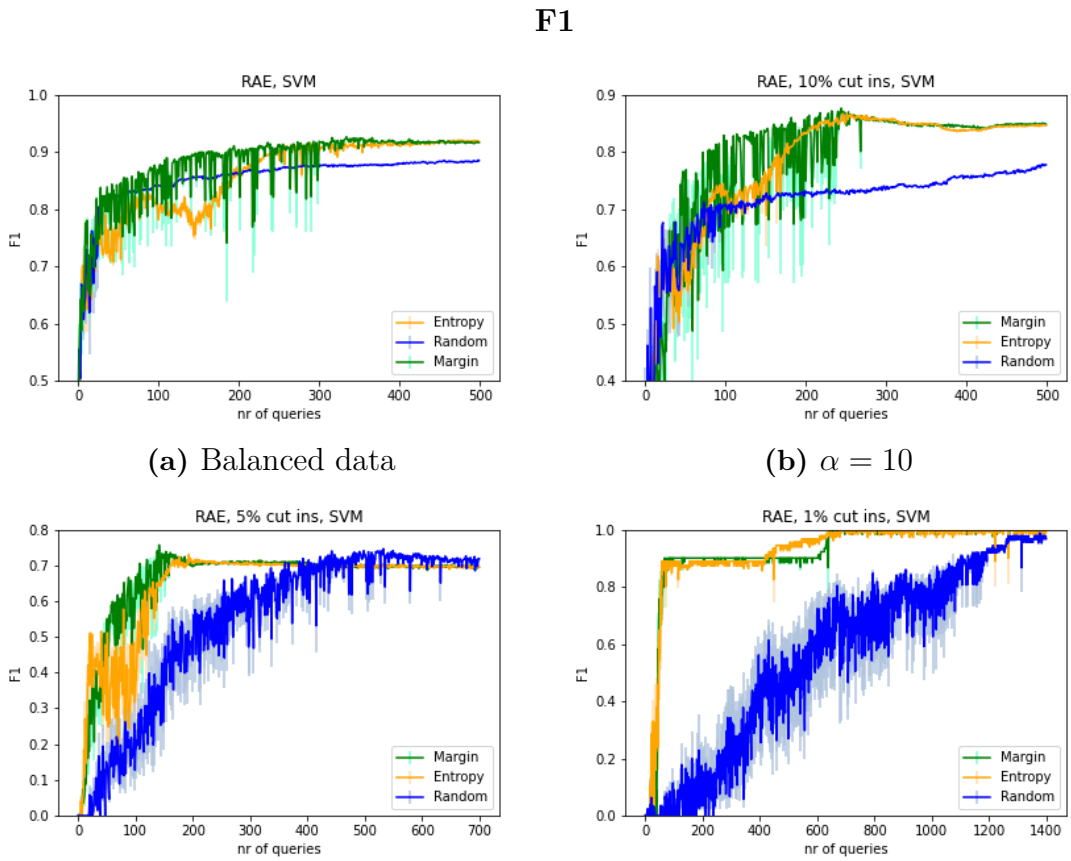
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.3: AUC for different class distributions with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



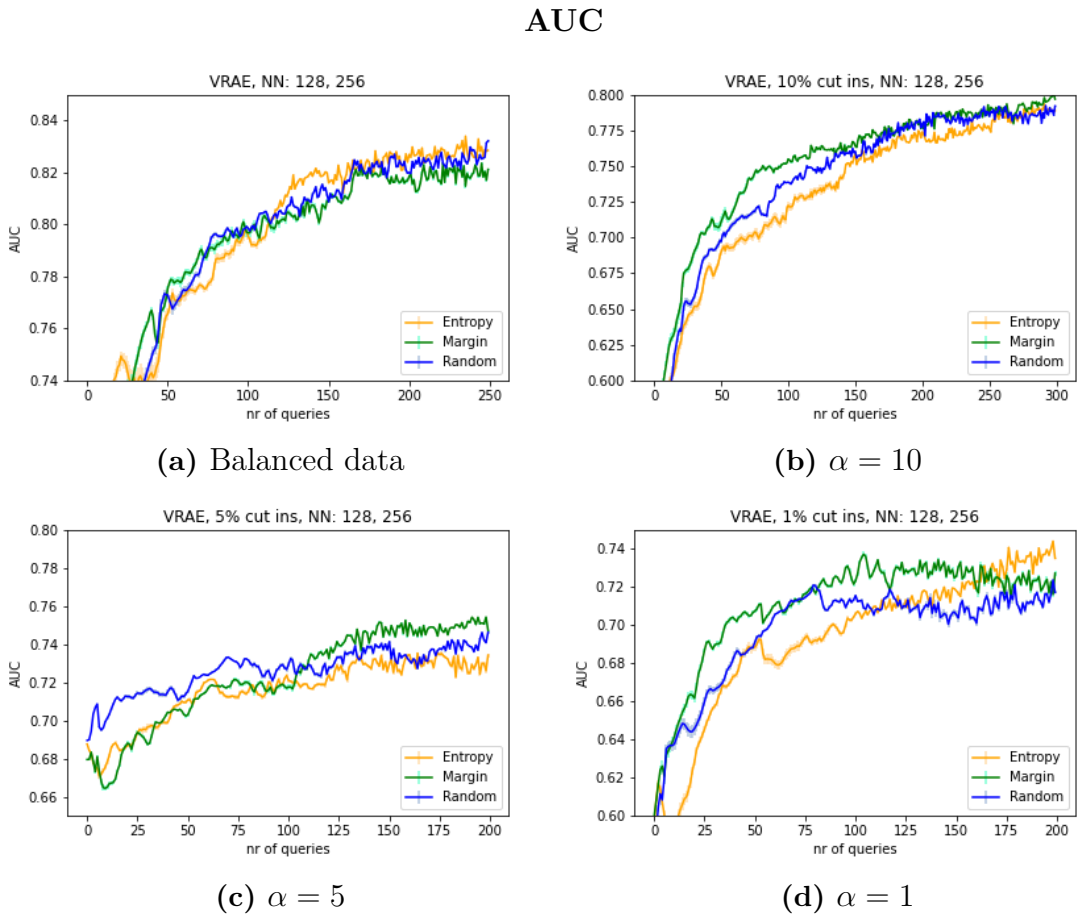
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.4: F1 score for cut in for different class distributions with data embedding from RAE. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



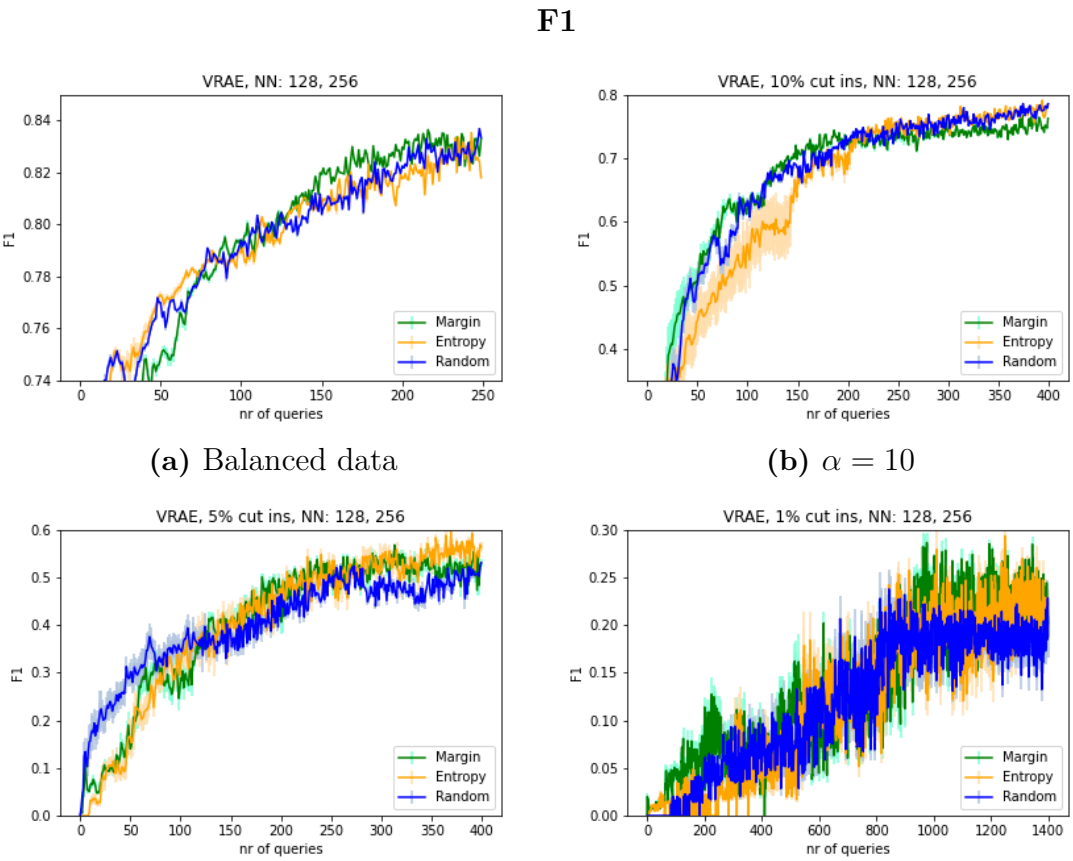
(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.5: AUC for different class distributions with data embedding from VRAE, using the smaller network as classifier. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data.



(a) Balanced data

(b) $\alpha = 10$

(c) $\alpha = 5$

(d) $\alpha = 1$

Figure B.6: F1 score for cut in for different class distributions with data embedding from VRAE, using the smaller network as classifier. The different colors represent the query methods. The data plotted is the mean over 10 runs, and the lighter colored lines represent the variance in the data. Note the scale of the axes.