# Formal Verification of Deep Neural Networks for Sentiment Classification

Paulius Skaisgiris
*Department of Data Science and Knowledge Engineering*
*Maastricht University*
Maastricht, The Netherlands

***Abstract*—Deep neural networks are powerful methods for modelling data and are increasingly deployed as part of safety-critical systems. Unfortunately, common neural networks evaluation methods draw conclusions which are often too optimistic for real-life scenarios. In this paper, we investigate verification, a method that ensures certain properties of neural networks. Specifically, natural language processing context is considered as research on verification of networks in this domain is scarce. We examine the performance of existing verification framework applications on networks performing sentiment classification as well as investigate piecewise linear activation function trade-offs. Moreover, latent space properties of many text representation techniques are investigated. Our empirical results show that the latent space induced by an autoencoder trained with a denoising adversarial objective is useful for verifying robustness of networks performing sentiment classification as well as interpeting results of verification tool outcomes.**

***Index Terms*—Formal Neural Network Verification, Robustness Verification, Sentiment Classification, Latent Space Analysis, Piece-Wise Linear Activation Functions**

## I. Introduction

Deep neural networks (DNNs) are reported to achieve superhuman performance across a wide variety of tasks. A common technique for neural network evaluation is computing a model's predictions on a large set of points from the input space and deciding whether the outcome is as desired. However, the input space set has essentially infinite cardinality, meaning it is impossible to check whether the model correctly predicts the outcome for all possible inputs [1]. Thus, the performance of the model in the real world, where the encountered input data is likely vastly different, may decrease significantly.

Indeed, DNNs have been shown to be surprisingly fragile. [2] demonstrated that neural networks for computer vision are susceptible to small, semantically invariant input perturbations that lead the model to misclassify. Similar effects have been observed in the natural language processing (NLP) domain. Adversarial examples have been found by appending distracting text [3], paraphrasing [4], inserting typos [5], or changing words in the text with synonyms [6–9].

Since humans are often not fooled by these so-called adversarial examples, these results bring to light a significant problem in the models' understanding. Furthermore, it raises serious concerns in situations where incorrect outputs lead to costly consequences, such as in safety-critical systems or when these models interact with people.

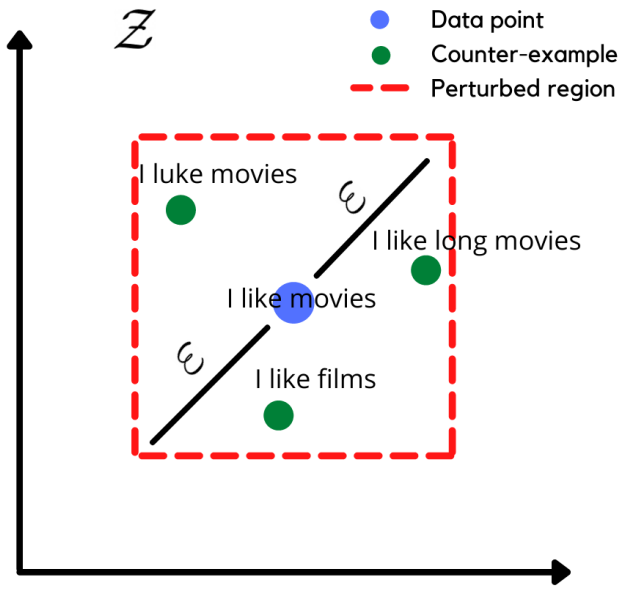



Fig. 1: Hyper-rectangle in the latent space enclosing semantically similar sentences. For such a latent space we could meaningfully verify robustness for NLP models.

Previous works turned to data augmentation [10, 11] and adversarial training [5, 12–14] in an attempt to tackle this problem. While they have improved performance on adversarial examples, the problem is still not solved. For instance, these efforts offer no guarantee that stronger attacks cannot break the system [7]. Formal verification methods provide a guarantee that the neural network in question possesses specific desirable properties. The adversarial robustness property could be formulated as follows: Points nearby an input example belong to the same output class as the input.

It has been proven that verifying neural networks with Rectified Linear Unit (ReLU) activations is NP-complete due to their large non-linear structure [15]. Verification of NLP neural networks is even more difficult. The input space is discrete and points in this space are converted to real-valued high-

This thesis was prepared in partial fulfilment of the requirements for the Degree of Bachelor of Science in Data Science and Artificial Intelligence, Maastricht University. Supervisor: Dr. Pieter Collins

dimensional vectors in order to benefit from common gradient-based optimization techniques. The main problem of this is that representing the text as a vector usually involves some black-box algorithm which makes it impossible to directly map back to discrete texts. Previous works have tried avoiding this issue by applying semantically-invariant perturbations to the discrete inputs [6, 7]. This is a limited approach because there is a combinatorial explosion of perturbations possible for text, and thus the authors were limited to verifying robustness of specific classes of perturbations like synonym or character substitutions.

This thesis studies how and to what extent NLP neural networks can be verified using existing formal verification frameworks. Since the robustness property closely relies on the latent space, we investigate verification performance on a plethora of text encoding techniques. Furthermore, denoising adversarial autoencoders (DAAE) are leveraged to solve a few key issues with NLP verification. DAAE encodes text to a space with a well-behaved geometry, thus it is a favourable approach for proving robustness for text against a broad class of adversarial perturbations in the input space. The key idea is pictured in Figure 1. Verification tools mainly support neural networks with piece-wise linear activation (PWL) functions because these functions are more amenable for formal verification [16]. It is thus worthwhile to investigate performance and verification trade-offs for neural networks with PWL activation functions. This work is restricted to verification of the robustness property as literature for other properties is very limited. Moreover, the IMDB movie review dataset [17] was used for training neural networks to perform binary sentiment classification. This work also only considers feed-forward neural networks as existing verification frameworks mainly support only these kinds of networks.

Concretely, this work attempts to answer the following research questions:

1) How can the robustness property for sentiment classification be formulated in a form which is amenable for formal verification analysis?
2) What complexity of neural networks performing sentiment classification can be verified by existing verification tools?
3) What are the trade-offs of piecewise-linearly approximated activation functions in neural networks compared to their smooth variants?

## II. Related Works

The first computational neural network verification approaches used satisfiability modulo theory (SMT) solvers [18, 19] to perform verification of neural networks. The approach was highly limited and could only verify networks with a single hidden layer and 10-20 hidden nodes. The first efficient DNN verification tool capable of verifying feed-forward neural networks (FFNNs) with 8 hidden layers and 300 hidden nodes per layer was developed by [15]. Authors extended the simplex algorithm for linear programming to handle non-convex ReLU activation functions. This work drew the interest of many researchers and led to an increase in research in this area.

Most current verification tools are limited to verifying FFNNs with ReLU functions. In general, the existing NN verification methods can be organized into three categories: Reachability, optimization, and search [1]. Reachability methods perform layer-by-layer reachability analysis of the network [20–23]. Optimization methods create linear constraints for nonlinear activation functions and attempt to find a configuration that does not satisfy the constraints using optimization. The optimization methods can be further sub-divided to primal formulation methods [24–26], dual formulation methods [27, 28], and semidefinite programming [29, 30]. Search methods are combined with reachability [31–33] or optimization methods [15, 16, 34] as these approaches provide search directions.

In general, while there has been a vast increase in amount of neural network verification approaches and tools developed in recent years, the properties that are chosen to be verified have not changed. The properties prevalent in literature include safety and robustness. Verification of a safety - also sometimes called reachability - property is the process of examining whether inputs in a specified range are always predicted to be in some desired range. Robustness seeks to ensure that there does not exist any points near an input point by some distance that change the output of the network. [35] investigated richer properties such as the conservation of energy in a physical system, the downstream task of handwritten digit addition, and the semantic change of the predicted label. The property of equivalence of two networks was studied by [36] but was restricted to binarized neural networks. [37] also studied the relationship between two networks and investigated whether the output logits of the two networks are within some small distance. Literature on more sophisticated properties than reachability or robustness exists yet remains scarce and is not yet widely adopted by the verification community. For this reason, the present paper will exclusively focus on the verification of the property of robustness.

[38] compared the types of networks and properties verified by several existing verification tools. They also stressed the need of a unified framework and common benchmarks for DNN verification. Following their work, a few DNN verification frameworks, such as `NeuralVerification.jl` [39] and `DNNV` [40], emerged which implemented a plethora of existing verification approaches. `DNNV` supports feed-forward and convolutional neural networks (CNNs) with piece-wise linear activation functions. While `NeuralVerification.jl` does not support CNNs, it provides methods for verifying networks with smooth activation functions. This thesis uses `NeuralVerification.jl` for NN verification because of its implementation of a wide range of verification methods, ease of use and extensive documentation. A thorough comparison study of existing verification tools using common benchmarks and the well-known MNIST dataset was carried out by [1]. A workshop and competition [41] was held with a similar goal in mind aiming to compare existing neural

network verification tools and standardize benchmarks.

One of the first works to point out that NLP networks are susceptible to adversarial examples was written by [3]. In particular, their work showed that appending distracting sentences to paragraphs of text fooled question-answering NLP models. Recently, several works have been presented on building defenses for neural networks against various types of adversarial attacks. [6] investigated one exponential class of text transformations: Synonym substitutions. During training, they employed Interval Bound Propagation (IBP) [27] to minimize an upper-bound on worst-case loss that any combination of word replacements can produce. [7] also modified the loss with IBP and trained networks with the auxiliary objective of being efficiently verifiable. They further extended this approach to symbol substitutions, that is, synonym and misspelling perturbations. Both of the aforementioned approaches were able to provide certified robustness guarantees for CNN models, the former method also investigated RNNs. IBP is also used by [42] to prove robustness against text deletion - models should not be more confident if valuable words from a sentence are deleted. Authors also investigated models with decomposable attention layers.

It is widely accepted that perturbed sentence vectors do not map back to discrete inputs [12]. However, possible approaches of doing so have not been investigated explicitly. This work attempts to bridge this gap.

## III. Methods

### A. Numerical text representation

Optimization algorithms are at the core of deep learning. These algorithms search for the best possible parameters for minimizing some defined loss function. Popular approaches benefit from the gradient information calculated at some input training example $p$. It is not possible to compute the gradient of this example $p$ if that point is a piece of text, a string. Thus, text data must be converted to numbers. Representing single words or whole paragraphs of text as real-valued vectors has been a very popular research topic in recent years. Each of these approaches usually involve a trade-off between computational resources and semantic information preserved in the representation space.

In this thesis, we consider a variety of existing text representation algorithms for two reasons. First, it is unclear which text representations are best performing for which natural language processing tasks in general, but especially for verification. Second, it is unclear which representation algorithms and their corresponding latent space geometry best suit the robustness verification problem. For instance, [7] have observed that the bounded region to-be-verified can reduce in volume if a different text representation algorithm, which forces synonyms to be closer in the latent space, is used. This approach increased verified accuracy and the amount of verifiable samples.

Concretely, this thesis investigates DNN training using the following text encoding approaches:

- **GloVe** [43]: this approach is similar to the original word embedding approach, Word2Vec [44], in that it is an unsupervised technique to generate word vectors. However, GloVe is a count-based model, whereas Word2Vec performs predictive modelling.
- **FastText** [45]: extension of Word2Vec by treating a word as a composition of character n-grams. FastText has two advantages over GloVe: it is better at representing rare words as well as out-of-vocabulary words.
- **Doc2Vec** [46]: by adding a separate document-unique feature vector, Doc2Vec is able to provide a single vector for a paragraph of text.
- **InferSent** [47]: first approach that learned sentence embeddings in a supervised manner. This work lead to an increase in research on sentence encoders, and asked the question "which supervised task is best suited for sentence represenation learning".
- **Universal Sentence Encoder (USE)** [48]: this encoding approach tried addressing the abovementioned question by using multiple data sources and tasks to train a transformer-based network which learns language nuances in different contexts.
- **DistilRoBERTa**: recently, transformer [49] based models have been dominating the field of NLP. In this thesis, we consider a simplified RoBERTa [50] model trained on paraphrases. The motivation is that perhaps this model should generate sentence encodings in latent space which has very similar sentences close together as it was trained on paraphrasing data.

### B. Robustness verification

The goal of neural network verification is to analyze whether the model in question satisfies a given specification. Here, we only consider the specification of robustness: For a given data point, all nearby points within some distance $\varepsilon$ are predicted by the model to have the same label.

Let $x, x' \in X$, where $X$ denotes the input space set for a network, let $f(x)$ denote a predicted label by neural network $f$ for a point $x$. Then, formally, the global robustness property is defined as follows:

$$\forall x \forall x' : d(x, x') \leq \varepsilon, f(x) = f(x'),$$

where $d$ is the Chebyshev distance. Chebyshev distance, also called $L_\infty$ metric, is defined as follows: $max(|x_i - y_i|)$, where $x_i$ and $y_i$ are the standard coordinate values of vectors $x$ and $y$.

Using generic verification algorithms, it is essentially intractable to computationally assess whether this property holds for all points in the input space. This is due to the fact that the input space can have very high dimension. Hence, it is useful to define a weaker, local version of the robustness property. A network is said to be $\varepsilon$-locally-robust if it satisfies

$$\forall z \forall x' : d(z, x') \leq \varepsilon, f(z) = f(x'),$$

where $z \in Z$ and $Z$ is a compact subset of $X$. Ideally, $Z$ is large and representative enough to provide some meaning

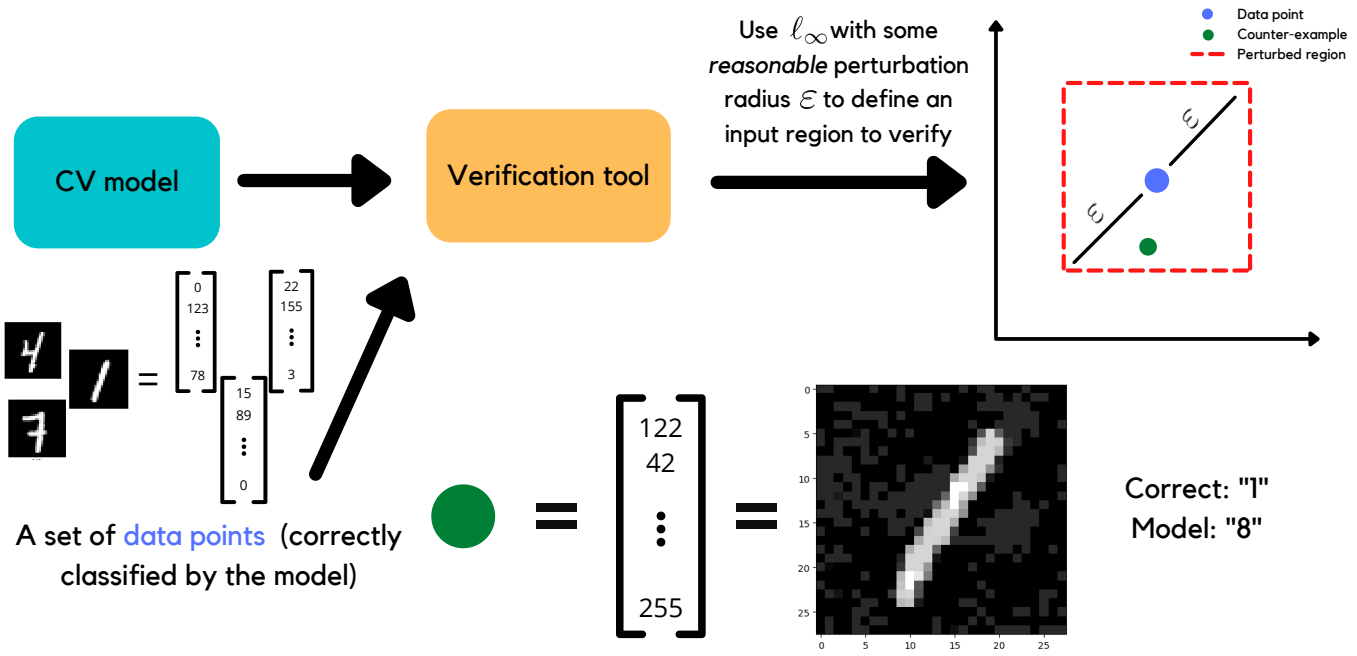



Fig. 2: Robustness verification process for networks performing classification on image data. Green circle is an adversarial example found within the $\varepsilon$-radius of some input data point (in blue). It is also shown that for such data an adversarial example can be easily mapped back to a concrete image. Handwritten digit images from [51].

about the network if the network is proven to be $\varepsilon$-locally-robust.

Robustness verification for computer vision networks is said to be more interpretable, as the perturbations have a clear meaning (for example, changing the brightness of a pixel) [13]. Hence, we will introduce the practical robustness verification procedure for computer vision first and discuss the problems that arise for NLP following it. Figure 2 shows the practical robustness verification procedure for NNs performing digit classification on image data. A neural network to-be-verified, and a set of data points, i.e. $Z$, from the test set is taken. These data points must be correctly classified by the network. Then, for each point $z \in Z$, a hyper-rectangle with radius $\varepsilon$ is crafted. In this context, this hyper-rectangle holds all points with slightly increased or decreased pixel values by at most $\varepsilon$ from the original data point. A verification solver is used to compute whether there exists a point in this hyper-rectangle that has a different predicted label than the original point. If the property holds, the result is `UNSAT` (solver has not found a counter-example in the input region). If the property is violated, the result it `SAT` and the solver usually is able to retrieve a counter-example for which the network misclassifies. This adversarial example can then be inspected by a human to determine whether the network should have known the label of this data point and, for example, it can be placed into the training set, and the model retrained with an augmented dataset. Alternatively, if the counter-example is perturbed too much even for a human to correctly identify its label, the perturbation radius could be decreased. Following this procedure, we can develop networks that are $\varepsilon$-locally-robust to adversarial perturbations.

If we follow the same procedure for networks performing NLP tasks, there are 3 problems:

1) Text encoding methods like the ones mentioned in Section III-A do not provide a way to map back to discrete input space for a given vector, let alone for a perturbed vector. If a counter-example is found by the verification tools, we cannot directly inspect what sentence it is and analyze the outcome.
2) Since text vectorization algorithms are opaque black-boxes, the encoded text vectors are impossible to interpret. Thus, perturbing the sentence vectors holds little to no meaning, and it is unclear what it means for a NLP network to be $\varepsilon$-locally-robust.
3) It is uncertain whether semantically similar sentences are nearby to one another by Chebyshev distance.

Recent works avoided these problems by introducing perturbations like synonym replacement and character typos in the discrete input space [6, 7] rather than in the latent space. However, their approach was limited to some specific class of perturbations and to the amount of perturbations applied as there is an enormous amount of possible perturbations for text.

This thesis attempts to interpret perturbations in the latent space with hope that a network can be proven to be robust for broader classes of perturbations. Thus, ideally, a bounded a hyper-rectangle around a sentence vector would incorporate

other sentences with similar semantic meaning as seen in Figure 1.

### C. Denoising Adversarial Autoencoder

As mentioned in Section III-B, there are a few problems with NLP system verification when applying perturbations to latent vectors. The following Section presents a method which can be used to address these issues.

Naively, Problem 1 could be tackled by a k-nearest neighbour approach. However, the performance of this approach highly depends on the size and variety of the dataset at hand. An alternative and more fine-grained approach to retrieve text from counter-examples in vector form is using an autoencoder [52]. The task of an autoencoder is to reconstruct the input. As a by-product, this neural network learns efficient data representations in an unsupervised manner. Its architecture is composed of two parts: Encoder and decoder. An encoder, over a sequence of hidden layers, compresses the data into a hidden representation and a decoder tries to reconstruct the input as closely as possible from this compressed encoding. This model is useful for NLP robustness verification - the encoder can be used to represent text data in a numerical form and the decoder part can be used to map back from counter-examples found by verification tools to discrete texts. Note that the decoding procedure is not perfect and is not guaranteed to give fully correct inputs, but it is a step forward in interpreting NLP verification.

[53] provide a solution to problem 3. In their paper, the researchers observe that neural encoders do not necessarily map similar sentences (both on a character-level and similar semantically) to nearby vectors in the latent space. The authors introduced a denoising objective to an adversarial autoencoder, thus creating a denoising adversarial autoencoder. This denoising procedure involves slightly perturbing the discrete input and asking the autoencoder to recreate the input to the original. DAAE is capable of mapping similar sentences in the discrete input space to points in the latent space that are nearby each other. DAAE latent space possesses interesting properties that enable meaningful sentence interpolation by traversing the latent space as well as to perform style transfer via latent vector arithmetic. The well-behaved latent space created by DAAE is intuitively useful for NLP robustness verification as similar sentences are close together in the latent space. Our expectation is that such well-behaved latent space allows us to bound a hyper-rectangle that encompasses semantically similar sentences as shown in Figure 1.

This thesis thus attempts to address problem 2 in Section IV-D by empirically analyzing latent spaces created by different text encoding methods.

### D. Networks with linearly approximated activations

Most neural network verification tools support verification of neural networks with PWL activation functions only. It is claimed that this decision allows verification algorithms to be *complete*, meaning they are guaranteed to always provide a definitive answer (whether the property holds or not) to a verification query. In fact, for simplicity, most tools support $ReLU = max(x, 0)$ function exclusively. However, state-of-the-art DNNs use various smooth non-linear activation functions like sigmoid or hyperbolic tangent. It is thus useful to look into how linear approximations of smooth functions affect the network performance and examine what the verification benefits are.

To the best of our knowledge, [54] produced the only work that investigated the impact of linearly approximated activation functions on neural networks. Their paper focused on an image classification task. For this work, we consider the automatic function linearization approach presented in [54]. Let $f : \mathbb{R} \to \mathbb{R}$ be a function over an interval $[l, u]$, where $l, u \in \mathbb{R}$, and let $n$ be the number of segments to approximate the function with. The function is split into $n$ equally sized sub-intervals $I_i$. For each $I_i$, compute the slope $m_i$ and intercept $b_i$. This approach thus returns lists $m$ and $b$ for all $n$ sub-intervals. The first value of these lists corresponds to a linear segment for $x < l$, similarly the last value of the lists corresponds to a segment for $x > u$. Combining these lists yields a piece-wise linear function:

$$\begin{cases} m_0 \cdot x + b_0 & \text{if } x < l \\ m_{i+1} \cdot x + b_{i+1}, \text{where } i = \lfloor \frac{x-l}{step} \rfloor & \text{if } l \leq x \leq u \\ m_{n+2} \cdot x + b_{n+2} & \text{if } x > u \end{cases}$$

This linearization method was utilized to make a PyTorch [55] activation module which takes in an arbitrary function and approximates it linearly. The module can then be used for training DNNs with PWL activations and the implementation is provided in the supplementary code. Function linearization applied to sigmoid and hyperbolic tangent functions can be viewed in the Appendix, in Figure 10. Throughout this thesis we refer to the function $f(x) = \frac{1}{1+e^{-x}}$ as the sigmoid function.

## IV. Experiments

### A. Text data

For the experiments, neural networks were trained on the IMDB movie review dataset [17]. Each review is labelled as negative or positive. The dataset is comprised of 50k reviews. The dataset is split by the authors of the dataset: 25k reviews in the training set and 25k in the testing set, with no class imbalance. In order to prevent information loss due to using text encoding and to make training of DAAE more effective, each review was split by sentence and the original review label was assigned to each sentence. We note that this decision may introduce some sentences in the dataset that do not have the correct label. The test set was split in half to make a validation set. The final datasets had 268k, 131k, 131k sentences for training, validation, and testing sets respectively. The final datasets contained roughly the same proportion of classes. No further data pre-processing was applied other than tokenization and making text lowercase.

### B. Text vectorization

Before training DNNs, the data was vectorized as the training time would have increased drastically and the data in vector form was used for different experiments as well. The sentences in training, validation, and test sets were encoded into vectors using methods described in IV-A.

- **GloVe**: model trained on 2 billion tweets with 27 billion tokens, producing 200-dimensional vectors was used. The pre-trained model is provided by [43]. The encoding interface by [56] was utilized for convenient text-vector conversion. GloVe provides embeddings for singular words, and sentence representations are made by averaging word embeddings appearing in a sentence. When creating sentence representations, words that do not appear in the vocabulary were skipped.
- **FastText**: model trained on wikipedia, IMBC, and statmt.org news datasets, totalling 16 billion tokens. Model produces 300-dimensional vectors and is provided by [57]. The encoding interface by [56] was utilized for convenient text-vector conversion.
- **Doc2Vec**: implementation provided by [56] was used and the Doc2Vec model was trained on the IMDB dataset described in Section IV-A. The model was trained to provide 100-dimensional vectors.
- **InferSent**: the version 2 pre-trained model utilizing FastText vectors internally was used and is provided by [47]. InferSent model converts sentences into 4096-dimensional vectors.
- **USE**: the version 4 pre-trained sentence encoder provided by [48] through TensorFlow Hub was employed. The model outputs 512-dimensional vectors.
- **DistilRoBERTa**: a pre-trained `paraphrase-distilroberta-base-v1` model from https://www.sbert.net/index.html [58] was used. The model provides 768-dimensional vectors and was trained on a broad categories of data ranging from wikipedia, AllNLI to question-answer data.
- **DAAE**: model implementation was accessed at https://github.com/shentianxiao/text-autoencoders [53]. Then, it was trained using an Nvidia GeForce GTX 2080Ti with 11GB VRAM on the IMDB data mentioned in Section IV-A. The training parameters are as follows: trained for 25 epochs, 10 set as the weight for adversarial loss, 0.3 as the probability for word drop, 30000 as the vocabulary size, output latent variables are 128-dimensional.

It is worth noting that GloVe and FastText produce variable-sized outputs as a vector is produced for each word in a sentence. Furthermore, some information is lost due to averaging of word embeddings to create fixed-size sentence representations.

### C. Training Deep Neural Networks

Training procedure was implemented in PyTorch. All trained networks had a feed-forward architecture with fully-connected linear hidden layers after which a non-linear function was applied to the data. Networks were trained on pre-vectorized data as described in Section IV-B. The same hyper-parameters were used across all training experiments: total number of epochs was set to 20, batch size of 1024 was used, loss function was set to be cross-entropy, Adam [59] was used as an optimizer with a constant learning rate of 0.1. If the network stopped learning, that is, the validation loss did not decrease by 0.001 or more for three consecutive epochs, then the training for the specific network configuration was stopped early. Metrics such as training time, training and validation loss, training and validation accuracy were saved for each epoch. After a network was trained, its performance was assessed on a test set and the confusion matrix was written to file. DNNs were trained on a laptop with 8 GB RAM, Intel Core i7-4720HQ CPU @ 2.60GHz and Nvidia GeForce GTX 950M GPU with 2 GB of VRAM, 640 CUDA cores.

A plethora of different neural network configurations were trained in an attempt to answer research questions two and three. Namely, the networks were trained on data encoded using the seven methods described in Section III-A. To investigate the effect of network sizes for verification, 5 different network sizes were chosen: 2 hidden layers with 10 hidden nodes each, 5 hidden layers with 25 hidden nodes, 5 hidden layers with 300 hidden nodes, 20 hidden layers with 25 hidden nodes, and 20 hidden layers with 300 hidden nodes. Lastly, five separate non-linear activation functions were used: ReLU, hyperbolic tangent, PWL approximation of hyperbolic tangent, sigmoid, PWL approximation of sigmoid. Hence, there were $7 \times 5 \times 5 = 175$ networks trained in total. Various performance metrics of these networks are reported in Tables XI-XVII in the Appendix.

### D. Latent spaces for NLP robustness verification

To start investigating verification, a subset $Z$ of the test set must be picked. The points in $Z$ will be used to perform local robustness verification. 50 examples from the test set were chosen by the author of this thesis that were thought to have a relatively unambiguous label.

*1) Measuring the efficacy of robustness verification:* Robustness verification deals with investigating whether points within a hyper-rectangle with radius $\varepsilon$ around an input point are predicted to have a different label than the input point. This chosen radius must not be too small as it might else be the case that there are no points in the whole dataset that fall within this hyper-rectangle. In that case, robustness could be considered trivial and would bear no meaning for the network.

To test whether there exist points in the dataset that are closer than $\varepsilon$ radius, k-nearest neighbours (KNN) models were trained. Seven separate KNN models for each text encoding method were fitted using 100k (InferSent used 20k because of memory issues) vectorized data points from the training set. The proportion of verification sentences for which KNN predicts there to exist at least one other sentence within distance $\varepsilon$ was computed. Two hundred equally spaced points between 0.0001 and 3.0 were investigated. Figure 4 shows

the results of this experiment. Additional way to assess the size and density of a vector space is by computing nearest neighbour distances. The KNN models described above were utilized for this task. Table III shows the results of the average distances and distribution of distances can be seen in Figure 14 in the Appendix.

Reasonable perturbation radius search can further be guided by applying perturbations in the discrete input space and computing the distance to the original vector. Three sentences were picked with increasing difficulty:

- $\sigma_1$ = "i like movies"
- $\sigma_2$ = "watching the jaws was a quite terrifying experience"
- $\sigma_3$ = "john had to write to the right people to keep his rights during his rites"

Text perturbations were applied using `TextAttack` [60]. Sentences were augmented by single character typos or replacing single words with WordNet synonyms. Each of the approaches generated 20 sentences. These augmented sentences were converted into vectors and Chebyshev distance was computed to the original vector. Minimum and maximum values of the distances are reported in Table I for character-level perturbations and in Table II for synonym replacements.

*2) Neighborhood preservation:* [53] introduced a measure of recall in order to quantify how well autoencoders preserve sentence space neighborhood structure in the latent space. In other words, whether sentences similar in the discrete input space are nearby each other in the latent space. The authors computed nearest neighbours of sentences in the text space by utilizing normalized edit distance which is Levenshtein distance of two sentences divided by the max length of two sentences. They also computed nearest neighbours of the sentences based on Euclidean distance in the latent space. Thus, the recall rate is defined as

$$\frac{|\text{NN}_x \cap \text{NN}_z|}{|\text{NN}_x|},$$

where, for some sentence, $\text{NN}_x$ is the set of nearest neighbours in the text space and $\text{NN}_z$ is the set of nearest neighbours in the latent space. This can be evaluated for different values of $k$ neighbours. [53] fixed $|\text{NN}_x| = 10$ and authors varied $k$ to allow for neighbours that are further away in the latent space. In this thesis, recall rate is computed for different text encoding methods including DAAE and is shown in Figure 3. Nearest neighbours in the latent space are computed using Chebyshev distance instead of Euclidean.

Autoencoders can help interpret the output of verification tools for NLP networks by mapping arbitrary vectors to text. However, they are not guaranteed to reconstruct the input perfectly. Tables VIII and VII in the Appendix supports this notion by showing performance capabilities of DAAE. The network on which the solver ran with perturbation radius 1.31 has 5 hidden layers, 25 hidden nodes per layer and ReLU activation functions.

### *E. Robustness verification for sentiment classification*

The experiments outlined below were carried out using `NeuralVerification.jl` verification framework written in Julia. The library is very accessible: It has extensive documentation and tutorials online. It further implements ways to verify safety and robustness of feed-forward neural networks. A plethora of verification approaches are implemented in `NeuralVerification.jl` including complete methods; support for smooth activation functions is also present. The experiments were carried out on the same laptop described in Section IV-C.

The first set of experiments for verifying robustness concerned verifying FFNNs with ReLU activation functions. Before verification, the model classified 50 sentences of set $Z$. The ones that have been correctly classified were used for verification. A property-to-be-verified in this context means picking some sentence vector, crafting a hyper-rectangle by adding and subtracting $\varepsilon$ to each value in the vector and checking if all points within this hyper-rectangle have the same output label as the original sentence vector. DLV [32] was chosen as a solver as some preliminary experiments have proven it to be rather fast. A timeout of 10 seconds was imposed for each property. Figure 6 shows the proportion of verification outcomes for varying $\varepsilon$ perturbation radii. Since from this plot it is unclear why some properties do not hold, we investigated five specific epsilon radii with a larger timeout of 200 seconds. Results are reported in figures 8 and 9 for DistilRoBERTa and DAAE encodings respectively as well as for three activation functions: ReLU, Sigmoid, and linearized Sigmoid.

Subsequent experiments investigated the performance of verification for ReLU networks for all five NN sizes. Perturbation radius was fixed to 0.0002, and timeout of 10 seconds was imposed per property. Results are depicted in Figure 5 for DAAE encoding. For results using other encodings, please refer to the Appendix.

## V. Results and Discussion

### *A. Analysis of latent spaces for NLP verification*

| Encoding | $\sigma_1$ | | $\sigma_2$ | | $\sigma_3$ | |
|---|---|---|---|---|---|---|
| | min | max | min | max | min | max |
| GloVe | 0.286 | 2.179 | 0.138 | 0.648 | 0.048 | 0.390 |
| FastText | 0.059 | 0.222 | 0.013 | 0.035 | 0.009 | 0.022 |
| Doc2Vec | 0.038 | 0.109 | 0.033 | 0.068 | 0.041 | 0.125 |
| USE | 0.086 | 0.168 | 0.045 | 0.136 | 0.040 | 0.114 |
| InferSent | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.147 |
| DistilRoBERTa | 0.522 | 1.627 | 0.326 | 0.903 | 0.241 | 0.924 |
| DAAE | 0.510 | 1.347 | 0.143 | 0.764 | 0.089 | 1.080 |

TABLE I: Chebyshev distance of 20 vectors with a single character perturbation in the discrete input space from original sentence vector. Results reported for different encoding methods and three sentences with increasing difficulty.

Figure 3 shows how well different text representation methods map similar sentences to similar latent representations by

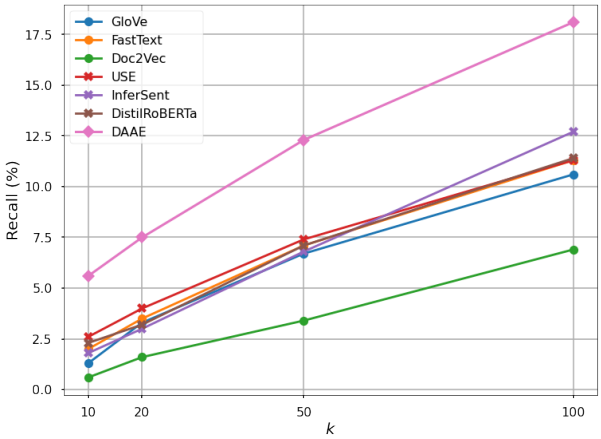



Fig. 3: Recall rate of different text encoding methods on the IMDB dataset. This measure quantifies how well text similarity based on Levenshtein distance is preserved in the latent space based on Chebyshev distance.

| Encoding | $\sigma_1$ | | $\sigma_2$ | | $\sigma_3$ | |
|---|---|---|---|---|---|---|
| | min | max | min | max | min | max |
| GloVe | 0.272 | 2.054 | 0.166 | 0.714 | 0.060 | 0.410 |
| FastText | 0.044 | 0.084 | 0.011 | 0.037 | 0.008 | 0.042 |
| Doc2Vec | 0.042 | 0.094 | 0.040 | 0.010 | 0.044 | 0.092 |
| USE | 0.101 | 0.193 | 0.055 | 0.147 | 0.029 | 0.095 |
| InferSent | 0.000 | 0.000 | 0.000 | 0.159 | 0.000 | 0.181 |
| DistilRoBERTa | 0.376 | 1.286 | 0.352 | 1.546 | 0.129 | 0.832 |
| DAAE | 0.348 | 0.973 | 0.111 | 0.840 | 0.037 | 0.277 |

TABLE II: Chebyshev distance of 20 vectors with a single WordNet synonym perturbation in the discrete input space from original sentence vector. Results reported for different encoding methods and three sentences with increasing difficulty.

| Encoding | Avg. nearest neighbour distance |
|---|---|
| GloVe | 0.21442 |
| FastText | 0.03604 |
| Doc2Vec | 0.07285 |
| USE | 0.11428 |
| InferSent | 4.322156e-08 |
| DistilRoBERTa | 0.751579 |
| DAAE | 0.62981 |

TABLE III: Average Chebyshev distance of 10k points from the training set to their nearest neighbour.

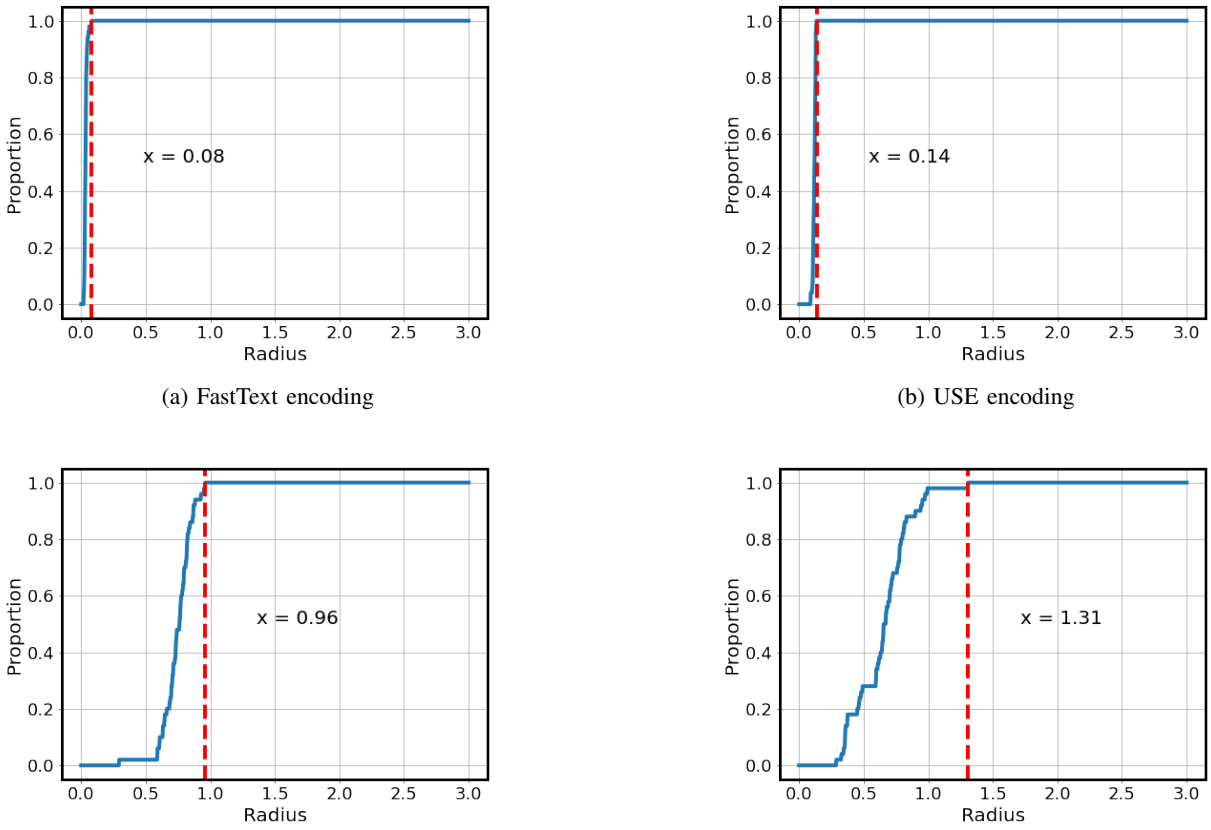
Chebyshev distance. It is expected that Doc2Vec performs poorly - the model was trained by authors of this thesis from scratch without extensive hyper-parameter search. As hypothesized, DAAE maintains the best recall rate amongst all other text encoding methods. These findings provide hope that DAAE, at the very least, is better at mapping similar texts on a character-level nearby each other in the latent space than other methods.

In order to start verification procedure, we needed to choose a suitable size for the hyper-rectangle. First, it is important to take a radius that incorporates at least one other point in the dataset as otherwise we cannot say that the network is robust against any other data points. Second, it is well-known that vector spaces induced by separate text representation techniques will have distinct properties as the training procedure is different and tries to capture unique linguistic nuances. Figure 4 shows what is the minimum distance needed to incorporate at least one data point for all 50 verification sentences. Both FastText and USE seem to incorporate at least one other data point for a relatively small radius. Sentences encoded by DistilRoBERTa and DAAE are more dispersed in the latent space - a larger radius is needed to reach at least one other data point.

Tables I and II show how slight perturbations in the discrete input space affect the resulting vectors in the latent spaces. The distance for each perturbed sentence to the original vector was computed. The reported minimum distance means that one should verify a network using that encoding with *at least* that large of a bound, as smaller distances would not hold any meaning. Similarly, the maximum value hints to a radius of a hyper-rectangle which could bound sentences perturbed in this manner. Interestingly, there seems to be no consensus over the values: Shorter, ”easier” sentences have larger distance values and longer, ”more difficult” sentences have smaller values. Perhaps this is of no surprise as the longer the sentence, the less its meaning is impacted by one-symbol substitutions. From the tables it is also evident that the distance depends on the number of dimensions of the vector space and how well-spaced the points are. For instance, DistilRoBERTa has 768 dimensions yet has values similar to DAAE which has 128 dimensions. USE also has relatively high number of dimensions but the vector space seems much more dense, the points are closer together. An extreme case of the latter is InferSent which has extremely small distances between points but also has 4096 dimensions - more than any other method by far. Possibly, for verification, the radius of the hyper-rectangle should be adjusted on the length of the sentence and number of dimensions. In addition, the verification input region does not need to be a hyper-rectangle. It could be a hyper-polytope and could bound an input point more tightly and more precisely for some class of perturbations in the discrete space. In this thesis, however, we do not investigate these possibilities and rely mainly on the values discovered in Figures 4 all the more so that the values discovered there incorporate most of the values seen in Tables I and II. Table III offers further insight into the scale of the vector space and portrays results similar to Tables I and II.

### B. *Robustness verification for NLP deep neural networks*

Figure 5 portrays the verification outcomes and the cumulative time it took to verify them for ReLU networks with different sizes. The Figure points to an obvious fact: Network sizes affect the effectiveness of verification. The biggest network constantly takes order of magnitude longer than the smaller networks. In particular, it seems that the number nodes in the network seem to affect the speed of




(a) FastText encoding

(b) USE encoding

(c) DistilRoBERTa encoding

(d) DAAE encoding

Fig. 4: Plots showing the proportion of 50 cherry-picked verification data points that have at least one data point from the training set with at most specified Chebyshev distance away from them. The red dotted line shows the maximum of minimum distances needed to reach at least one other data point appearing in the training set for each of the verification sentences.

verification. This is deduced because a shallower, 5 layer network takes longer to verify all its properties than a deeper 20-layer network. The first DNN has 1500 nodes, and the second one only 500 nodes.

Figure 6 shows the proportion of properties and their outcomes found by DLV solver for increasing radii of input hyper-rectangle. The results are presented for networks of the same size. The hope is that the network properties would hold for (nearly) all values. Based on the latent space analysis made earlier, one could then say that the network is robust to adversarial attacks at least for character and synonym replacements. Unfortunately, all networks other than DAAE fail to verify properties quite early on. To compare these results, we can compute the percentage of radii covered until the solver can no longer prove any properties to hold, in other words, when the proportion of properties which hold hits 0. Results are as follows: FastText: 2.5%, USE: 5.0%, DistilRoBERTa: 8.54%, DAAE: 83.9%. DAAE networks seem to possess favourable properties for robustness. It is both possible to verify robustness for larger radii and the networks seem to be more robust for larger radii than e.g. DistilRoBERTa networks. The latter can be deduced from the fact that properties are being violated for larger radii. One could say that it is possible to efficiently verify DAAE because of its small dimension size - 128. However, FastText has 300-dimensional vectors and starts timing out quicker than USE and DistilRoBERTA which have 512 and 768 dimensions respectively. It is unclear how truly robust FastText and USE networks are, but, in general, we report that it takes much longer to arrive at a conclusion using the DLV solver.

Table IV demonstrates the nearest neighbours for a specific sentence as well as for its counter-example found by Reluplex. According to qualitative inspection, KNN is a good approach to interpreting NLP verification results. One notable downside is that the nearest neighbours of the counter-examples could exceed the perturbation radius as is the case for USE and DistilRoBERTa encodings in Table IV. KNN performance highly depends on the data it has been trained on. Memory resources available to us only permitted fitting KNNs on 100k data points even though over 500k were available in total for IMDB dataset. Naturally, this approach does not create new samples for which the DNN in-question misclassifies. The

| **Input: the plot is ridiculous and the characters are horrible people . .** | |
|---|---|
| Text | $\delta_o$ |
| **GloVe** | |
| when something happens , the reactions of the characters are vague and dry.best not to look this one up . | 0.231 |
| **USE** | |
| the characters are awful , as is the story . | 0.089 |
| **DistilRoBERTa** | |
| the plot is ridiculous and the whole " little man " crap is just so stupid . | 0.590 |
| **DAAE** | |
| the acting is first class and the characters are represented well . | 0.629 |

(a) Nearest neighbour for a specific example.

| **Input: the plot is ridiculous and the characters are horrible people . .** | | |
|---|---|---|
| Text | $\delta_c$ | $\delta_o$ |
| **GloVe, radius = 0.05** | | |
| the script is bad , the zombies are awful , there is no tension , lines are bad , actors are bad .. the list just goes on.you will probably want to see this movie just because of its reputation of being awful . | 0.252 | 0.296 |
| **USE, radius = 0.014** | | |
| the plot and characters are ridiculous and barely qualify as " plot " and " character " . | 0.100 | 0.103 |
| **DistilRoBERTa, radius = 0.164** | | |
| the characters are shallow and trite as are the dialog and plot line . | 0.703 | 0.867 |
| **DAAE, radius = 1.31** | | |
| the acting is first class and the characters are represented well . | 1.939 | 0.629 |

(b) Nearest neighbour of a counter-example found by Reluplex for a specific sentence. $\delta_c$ is the Chebyshev distance to the counter-example.

TABLE IV: Nearest neighbours for different encodings. $\delta_o$ is the Chebyshev distance to the original point.

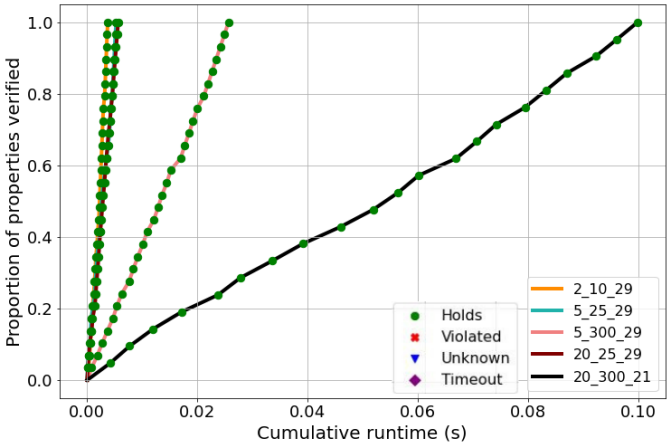



Fig. 5: Verification results of DLV solver for different sizes of ReLU networks for DAAE encoding. Input hyper-rectangle radius set to 0.0002. Timeout of 10 seconds per property imposed. The format of $x\_y\_z$ in the legend corresponds to the following properties of NNs: $x$ - number of layers, $y$ - number of nodes per layer, $z$ - number of examples correctly classified out of the 50 cherry-picked sentences.

nearest neighbours for GloVe, USE, and DistilRoBERTa seem to show legitimate counter-examples - the network should classify these sentences as negative but it classified a close point as positive. For DAAE, however, the counter-example is unreasonable as the sentiment is clearly positive.

An autoencoder is a fine-grained approach of interpreting

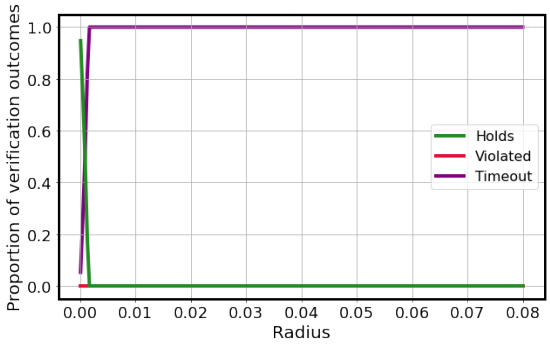
| | |
|---|---|
| Original | the plot is ridiculous and the characters are horrible people . |
| Original reconstruction | the plot is not very good |
| Counter-example reconstruction | the film is a very good film . |

TABLE V: Reconstruction performed by DAAE of a sentence and its counter-example found by Reluplex for radius 1.31.

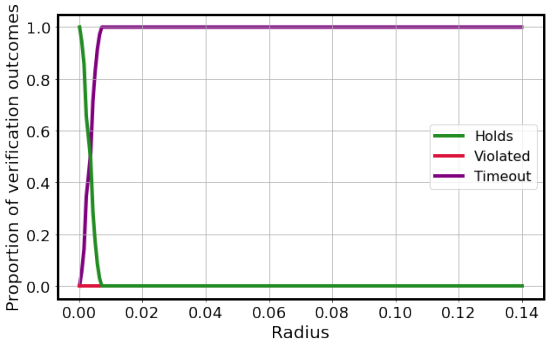
verification results and one which generates new samples. Table V shows the reconstructions using DAAE for the same sentence as in Table IV and its counter-example. The reconstruction of the original is quite similar to the first sentence albeit shorter. The reconstruction of the counter-example seems to point to the same conclusion arrived for DAAE using nearest neighbours, meaning that the perturbation radius of 1.31 is too wide for DAAE networks. This conclusion is to be taken with caution as other decision in the pipeline could have influenced this such as the reconstructive abilities of DAAE or the performance of the DAAE network. In general, further research is needed on network performance and optimal parameter configuration for DAAE.

### *C. Piece-wise linear activation function trade-offs for neural networks*

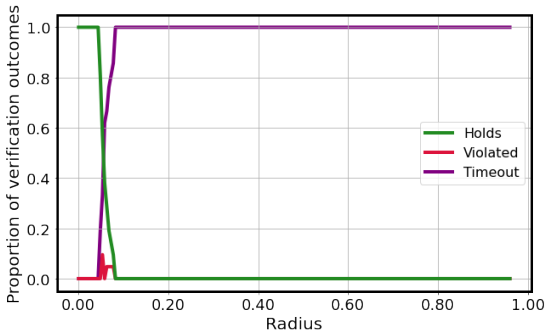
Table VI shows some performance measures of networks trained on various encodings and a fixed network size. For three out of four shown encoding methods, sigmoid activation function seems to provide strong results. Networks trained on DAAE encodings did not seem to perform well regardless of the activation. Increase in average training time per epoch for PWL activations is consistent, but minor: No more than one



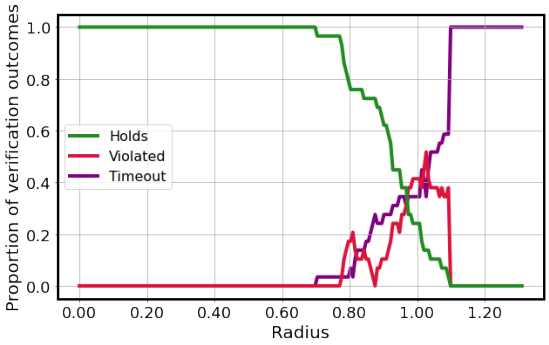

(a) FastText encoding




(b) USE encoding




(c) DistilRoBERTa encoding




(d) DAAE encoding

Fig. 6: Proportion of properties that are proven to hold by DLV [32] as the radius of the input hyper-rectangle is increased. Proportion calculated for 200 points between 0.0001 and minimum distance that incorporates at least one other point in the bounded hyper-rectangle as shown in Figure 4. Time out of 10 seconds imposed per property.

| Activation | Train accuracy | Test accuracy | Avg. train time per epoch (s) |
|---|---|---|---|
| Tanh | 50.13 | 51.17 | 38.54 |
| PWL Tanh | 50.29 | 51.17 | 38.98 |
| Sigmoid | **68.68** | **67.36** | 38.93 |
| PWL Sigmoid | 50.72 | 51.17 | 40.05 |
| ReLU | 67.37 | 67.07 | 39.15 |

(a) FastText encoding

| Activation | Train accuracy | Test accuracy | Avg. train time per epoch (s) |
|---|---|---|---|
| Tanh | 66.96 | 67.44 | 61.75 |
| PWL Tanh | 68.01 | 67.91 | 62.42 |
| Sigmoid | **70.52** | 68.86 | 62.44 |
| PWL Sigmoid | 50.71 | 48.83 | 62.98 |
| ReLU | 70.30 | **69.13** | 62.4 |

(b) USE encoding

| Activation | Train accuracy | Test accuracy | Avg. train time per epoch (s) |
|---|---|---|---|
| Tanh | 67.64 | 61.37 | 92.21 |
| PWL Tanh | 66.62 | 67.09 | 92.74 |
| Sigmoid | **69.76** | **68.80** | 92.34 |
| PWL Sigmoid | 50.62 | 48.83 | 93.22 |
| ReLU | 67.60 | 65.32 | 92.19 |

(c) DistilRoBERTa encoding

| Activation | Train accuracy | Test accuracy | Avg. train time per epoch (s) |
|---|---|---|---|
| Tanh | 50.16 | 48.83 | 17.70 |
| PWL Tanh | 50.16 | 48.83 | 18.71 |
| Sigmoid | 50.68 | 51.17 | 18.64 |
| PWL Sigmoid | 50.67 | 51.17 | 18.72 |
| ReLU | 50.69 | 51.17 | 17.94 |

(d) DAAE encoding

TABLE VI: Metrics for networks trained using different encoding methods and different activation functions. Network is feedforward with 5 hidden layers and 25 hidden nodes per layer.

second increase compared to their smooth versions. Previously, [54] have shown that the training time of DNNs trained using linearized activation functions approximated with three line segments was increased 2-3 times compared to the smooth variants. Researchers used MNIST [51] as data and the Keras training framework. Our findings bring to light the fact that FFNNs with PWL functions can be used for training networks using PyTorch with only a minor increase in training time.

Boxplots of test accuracies grouped by five activation functions can be seen Figure 7. It is evident that, on average, networks trained on PWL sigmoid are performing worse than with other activations. Linearzed tanh, however, performs better than its continuous counterpart. Another PWL function, ReLU, demonstrates good results consistently while continu-

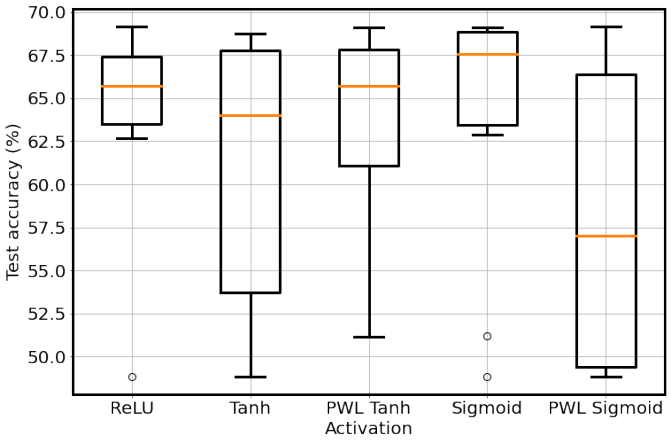



Fig. 7: Boxplots of test accuracies, grouped by activation functions. Only encodings of GloVe, FastText, Doc2Vec, USE, and DistilRoBERTa, and sizes of 2 hidden layers with 10 nodes-per-layer, 5 layers with 25 nodes-per-layer were considered. This is because other network configurations performed very poorly regardless of the activation. Box and whiskers plot computed using definition found in [61].

ous sigmoid shows the best results overall.

While the results presented seem to show that networks with linearized activation functions achieve comparable performance to their smooth versions, it is unclear how consistent such results are, especially for different DNN architectures. Tables in the Appendix show performance of networks for both deeper and wider networks. Further experiments employing a thorough hyper-parameter search could uncover more concrete trade-off between linearized and smooth activations still.

Figures 8 and 9 exhibit the effect of different activation functions for robustness verification. There seems to be no effect for DAAE networks, all properties are proven to hold by MaxSens [20] solver until perturbation of radius 1, when it times out. However, both sigmoid and linearized sigmoid seem to affect the outcome of verification for radius of 0.01. The proportion of properties verified to hold is around 50% for both sigmoid and PWL sigmoid DNNs while it is 100% for ReLU network. Thus, the choice of activation functions for DNNs can affect how robust it is.

## VI. Conclusion

This thesis examined existing verification approaches of neural networks for natural language processing. The performance of verification tools was shown to be impacted by input size, network size, and especially by input hyper-rectangle radius. PWL DNNs seem to have miniscule downsides in training time but are less likely to perform as well as their smooth counterparts. Networks trained using popular text encoding methods like word embeddings or sentence encoders in an ad-hoc manner were demonstrated to be inappropriate for providing good robustness guarantees. Prior work introduced an autoencoder with a modified loss function which has a well-behaved latent space geometry. The finding was leveraged to show, in turn, that networks trained on such encodings are better suitable for robustness verification. Lastly, the latent spaces, reasonable perturbation radius options, and verification result interpretation approaches were investigated. We conclude that nearest neighbour approach is good for interpreting and guiding perturbation radius search but autoencoders can be leveraged for generating semantically similar samples for which the network is not robust against.

## VII. Future work

Considering this thesis stressed the importance of latent space properties and how it helps with robustness verification, future work could further investigate interpretation of verification results and come up with latent spaces that better fit NLP verification but still keep the models rather strong.

Additional work on convolutional neural networks as well as recurrent neural networks could be carried out as verification tools that are capable of handling these types of networks exist (even if limited). Different properties can still be investigated involving downstream tasks or relationship between two networks, for example.

Furthermore, it would be interesting to see the impact of network simplification as well as adversarial training methods on NLP DNN verification.

## Acknowledgments

I would like to thank my supervisor Dr. Pieter Collins for showing me the broad and interesting field of neural network verification and guiding me through it. Also thanks to Dr. Jan Niehues for advising me on NLP-related matters. Furthermore, I would like to thank my friends and family who contributed to both my physical and mental well-being over the course of writing this thesis. Special thanks to Krzysztof Cybulski for valuable discussions and advice on PyTorch. Lastly, I would like to thank Drexciya Research Lab for inspiring me to dive deep into the realm of research.

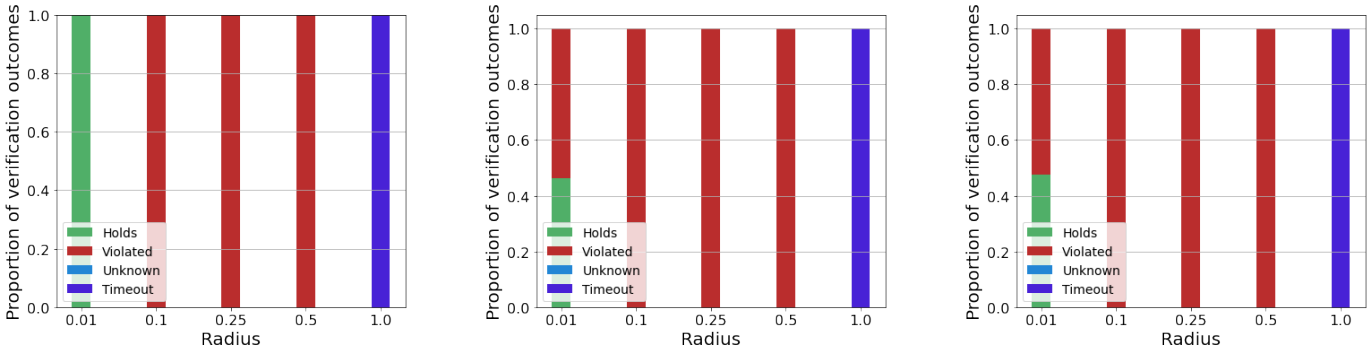
## References

[1] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *arXiv:1903.06758 [cs, stat]*, 2020. arXiv: 1903.06758. [Online]. Available: http://arxiv.org/abs/1903.06758 (visited on 11/05/2020).

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV].

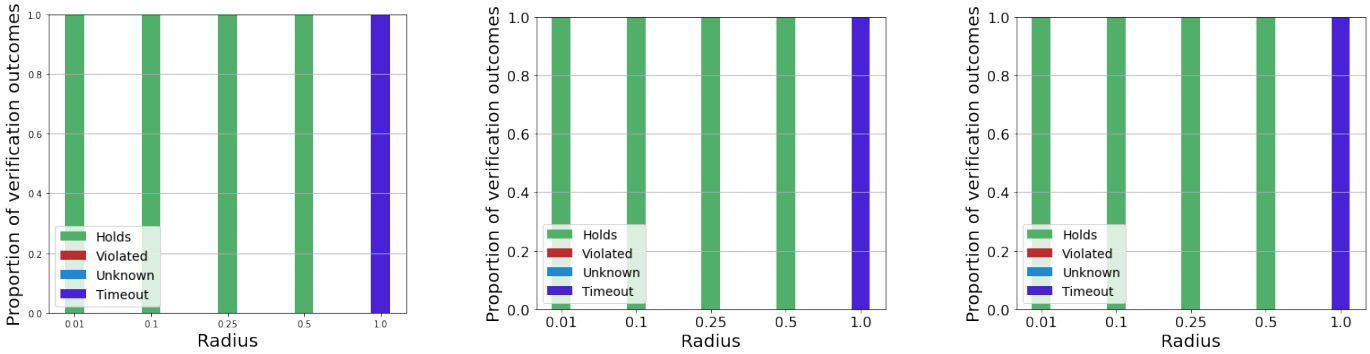
[3] R. Jia and P. Liang, *Adversarial examples for evaluating reading comprehension systems*, 2017. arXiv: 1707.07328 [cs.CL].

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging NLP models," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 856–865. DOI: 10.18653/v1/P18-1079. [Online]. Available: https://www.aclweb.org/anthology/P18-1079.




(a) ReLU (b) Sigmoid (c) PWL Sigmoid

Fig. 8: Proportion of verification outcomes found by MaxSens [20] solver for network with 2 hidden layers and 10 nodes per layer trained using DistilRoBERTa encodings. Results reported for different perturbation radii. Timeout of 200 seconds per property imposed.




(a) ReLU (b) Sigmoid (c) PWL Sigmoid

Fig. 9: Proportion of verification outcomes found by MaxSens [20] solver for network with 2 hidden layers and 10 nodes per layer trained using DAAE encodings. Results reported for different perturbation radii. Timeout of 200 seconds per property imposed.


[5] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, *Hotflip: White-box adversarial examples for text classification*, 2018. arXiv: 1712.06751 [cs.CL].

[6] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified Robustness to Adversarial Word Substitutions," *arXiv:1909.00986 [cs]*, 2019, arXiv: 1909.00986. [Online]. Available: http://arxiv.org/abs/1909.00986 (visited on 01/17/2021).

[7] P.-S. Huang, R. Stanforth, J. Welbl, C. Dyer, D. Yogatama, S. Gowal, K. Dvijotham, and P. Kohli, "Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation," *arXiv:1909.01492 [cs, stat]*, 2019, arXiv: 1909.01492. [Online]. Available: http://arxiv.org/abs/1909.01492 (visited on 03/16/2021).

[8] M. Ye, C. Gong, and Q. Liu, *Safer: A structure-free approach for certified robustness to adversarial word substitutions*, 2020. arXiv: 2005.14424 [cs.LG].

[9] X. Dong, A. T. Luu, R. Ji, and H. Liu, "TOWARDS ROBUSTNESS AGAINST NATURAL LANGUAGE WORD SUBSTITUTIONS," en, p. 14, 2021.

[10] Y. Li, T. Cohn, and T. Baldwin, "Robust training under linguistic adversity," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain: Association for Computational Linguistics, Apr. 2017.

[11] Y. Belinkov and Y. Bisk, *Synthetic and natural noise both break neural machine translation*, 2018. arXiv: 1711.02173 [cs.CL].

[12] T. Miyato, A. M. Dai, and I. Goodfellow, *Adversarial training methods for semi-supervised text classification*, 2017. arXiv: 1605.07725 [stat.ML].

[13] S. Barham and S. Feizi, "Interpretable adversarial training for text," *CoRR*, vol. abs/1905.12864, 2019. arXiv: 1905.12864. [Online]. Available: http://arxiv.org/abs/1905.12864.

[14] T. Wang, X. Wang, Y. Qin, B. Packer, K. Li, J. Chen, A. Beutel, and E. Chi, "CAT-Gen: Improving Robustness in NLP Models via Controlled Adversarial Text Generation," en, in *Proceedings of the 2020 Conference*

*on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, 2020, pp. 5141–5146. DOI: 10.18653/v1/2020.emnlp-main.417. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.417 (visited on 02/24/2021).

[15] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, *Reluplex: An efficient smt solver for verifying deep neural networks*, 2017. arXiv: 1702.01135 [cs.AI].

[16] R. Ehlers, "Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks," en, *arXiv:1705.01320 [cs]*, Aug. 2017, arXiv: 1705.01320. [Online]. Available: http://arxiv.org/abs/1705.01320 (visited on 10/24/2020).

[17] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: https://www.aclweb.org/anthology/P11-1015.

[18] L. Pulina and A. Tacchella, "An Abstraction-Refinement Approach to Verification of Artificial Neural Networks," vol. 616, 2010, pp. 243–257, ISBN: 978-3-642-14294-9. DOI: 10.1007/978-3-642-14295-6_24.

[19] ——, "Challenging SMT solvers to verify neural networks," *AI COMMUNICATIONS*, vol. 25, pp. 117–135, 2012. DOI: 10.3233/AIC-2012-0525.

[20] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multi-layer neural networks," 2017. arXiv: 1708.03322. [Online]. Available: http://arxiv.org/abs/1708.03322.

[21] ——, "Reachable set computation and safety verification for neural networks with relu activations," 2017. arXiv: 1712.08163. [Online]. Available: http://arxiv.org/abs/1712.08163.

[22] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation," en, in *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA: IEEE, 2018, pp. 3–18, ISBN: 978-1-5386-4353-2. DOI: 10.1109/SP.2018.00058. [Online]. Available: https://ieeexplore.ieee.org/document/8418593/ (visited on 10/27/2020).

[23] G. Singh, R. Ganvir, M. Püschel, and M. Vechev, "Beyond the single neuron convex barrier for neural network certification," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/0a9fdbb17feb6ccb7ec405cfb85222c4-Paper.pdf.

[24] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. V. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," *CoRR*, vol. abs/1605.07262, 2016. arXiv: 1605.07262. [Online]. Available: http://arxiv.org/abs/1605.07262.

[25] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *CoRR*, vol. abs/1706.07351, 2017. arXiv: 1706.07351. [Online]. Available: http://arxiv.org/abs/1706.07351.

[26] V. Tjeng and R. Tedrake, "Verifying neural networks with mixed integer programming," *CoRR*, vol. abs/1711.07356, 2017. arXiv: 1711.07356. [Online]. Available: http://arxiv.org/abs/1711.07356.

[27] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks," *CoRR*, vol. abs/1803.06567, 2018. arXiv: 1803.06567. [Online]. Available: http://arxiv.org/abs/1803.06567.

[28] J. Z. Kolter and E. Wong, "Provable defenses against adversarial examples via the convex outer adversarial polytope," *CoRR*, vol. abs/1711.00851, 2017. arXiv: 1711.00851. [Online]. Available: http://arxiv.org/abs/1711.00851.

[29] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *CoRR*, vol. abs/1801.09344, 2018. arXiv: 1801.09344. [Online]. Available: http://arxiv.org/abs/1801.09344.

[30] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *IEEE Transactions on Automatic Control*, 2020, ISSN: 1558-2523. DOI: 10.1109/TAC.2020.3046193.

[31] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," *CoRR*, vol. abs/1804.10829, 2018. arXiv: 1804.10829. [Online]. Available: http://arxiv.org/abs/1804.10829.

[32] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," *CoRR*, vol. abs/1610.06940, 2016. arXiv: 1610.06940. [Online]. Available: http://arxiv.org/abs/1610.06940.

[33] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *CoRR*, vol. abs/1811.00866, 2018. arXiv: 1811.00866. [Online]. Available: http://arxiv.org/abs/1811.00866.

[34] G. Katz, D. A. Huang, D. Ibeling, K. D. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. Dill, M. J. Kochenderfer, and C. Barrett, "The marabou framework for verification and analysis of deep neural networks," in *CAV*, 2019.

[35] C. Qin, Krishnamurthy, Dvijotham, B. O'Donoghue, R. Bunel, R. Stanforth, S. Gowal, J. Uesato, G. Swirszcz, and P. Kohli, *Verification of non-linear specifications for neural networks*, 2019. arXiv: 1902.09592 [cs.LG].

[36] N. Narodytska, S. P. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh, *Verifying properties of bina-*

*rized deep neural networks*, 2018. arXiv: 1709.06662 [stat.ML].

[37] B. Paulsen, J. Wang, J. Wang, and C. Wang, "NeuroDiff: Scalable differential verification of neural networks using fine-grained approximation," en, in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, Virtual Event Australia: ACM, 2020, pp. 784–796, ISBN: 978-1-4503-6768-4. DOI: 10.1145/3324884.3416560. [Online]. Available: https://dl.acm.org/doi/10.1145/3324884.3416560 (visited on 04/13/2021).

[38] F. Leofante, N. Narodytska, L. Pulina, and A. Tacchella, "Automated verification of neural networks: Advances, challenges and perspectives," *arXiv:1805.09938 [cs]*, 2018. arXiv: 1805.09938. [Online]. Available: http://arxiv.org/abs/1805.09938 (visited on 01/16/2021).

[39] C. Liu, T. Arnon, C. Lazarus, and M. J. Kochenderfer, "Neuralverification.jl: Algorithms for verifying deep neural networks," 2019.

[40] D. Shriver, *Dnnv: A framework for deep neural network verification*, 2019. [Online]. Available: https://github.com/dlshriver/DNNV.

[41] *3rd International Workshop on Verification of Neural Networks (VNN20)*, 2020. [Online]. Available: https://sites.google.com/view/vnn20/vnncomp.

[42] J. Welbl, P.-S. Huang, R. Stanforth, S. Gowal, K. ( Dvijotham, M. Szummer, and P. Kohli, "Towards verified robustness under text deletion interventions," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SyxhVkrYvr.

[43] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://www.aclweb.org/anthology/D14-1162.

[44] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. arXiv: 1310.4546. [Online]. Available: http://arxiv.org/abs/1310.4546.

[45] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Association for Computational Linguistics, Apr. 2017, pp. 427–431.

[46] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14, Beijing, China: JMLR.org, 2014, II–1188–II–1196.

[47] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *CoRR*, vol. abs/1705.02364, 2017. arXiv: 1705.02364. [Online]. Available: http://arxiv.org/abs/1705.02364.

[48] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," *CoRR*, vol. abs/1803.11175, 2018. arXiv: 1803.11175. [Online]. Available: http://arxiv.org/abs/1803.11175.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

[50] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[51] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[52] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *NIPS*, 1993.

[53] T. Shen, J. Mueller, R. Barzilay, and T. Jaakkola, "Educating Text Autoencoders: Latent Representation Guidance via Denoising," en, *arXiv:1905.12777 [cs, stat]*, Jul. 2020, arXiv: 1905.12777. [Online]. Available: http://arxiv.org/abs/1905.12777 (visited on 05/02/2021).

[54] W. Kokke, E. Komendantskaya, D. Kienitz, R. Atkey, and D. Aspinall, "Neural networks, secure by construction," in *Asian Symposium on Programming Languages and Systems*, Springer, 2020, pp. 67–85.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[56] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[57] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word

representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[58] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1908.10084.

[59] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

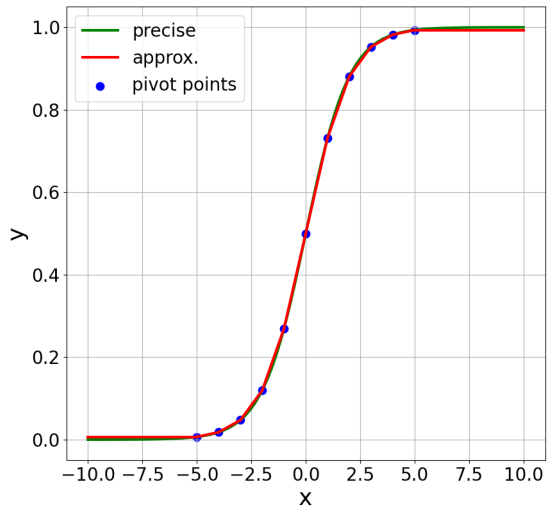
[60] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 119–126.

[61] J. W. Tukey *et al.*, *Exploratory data analysis*. Reading, Mass., 1977, vol. 2.

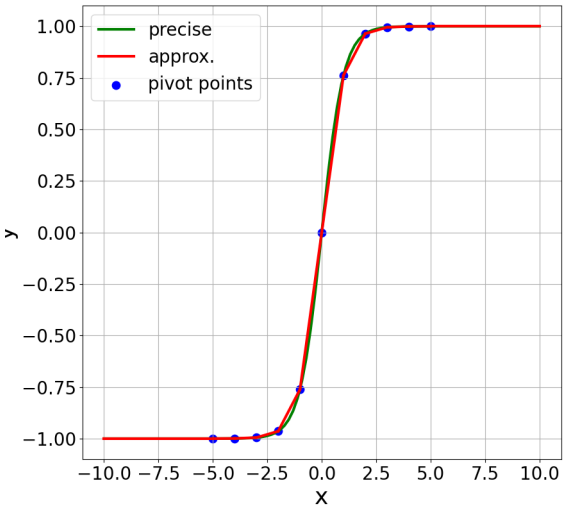
## APPENDIX

| **Input: this is a very good show .** | |
|---|---|
| Vector value | Reconstructed text |
| $v$ | this movie is a very good movie . |
| $v - 1.31$ | i was really really really really good movie . |
| $v - 1.17$ | i was not sure , i was not that . |
| $v - 1.03$ | i was not sure , i was a fan of this movie . |
| $v - 0.9$ | i was not a fan of this movie . |
| $v - 0.76$ | i was not a fan of this movie . |
| $v - 0.62$ | this movie is a very good movie . |
| $v - 0.48$ | this movie is a very good movie . |
| $v - 0.34$ | this movie is a very good movie . |
| $v - 0.21$ | this movie is a very good movie . |
| $v - 0.07$ | this movie is a very good movie . |
| $v + 0.07$ | this movie is a very good movie . |
| $v + 0.21$ | this movie is a very good movie . |
| $v + 0.34$ | this movie is a very good movie . |
| $v + 0.48$ | this is a very good movie . |
| $v + 0.62$ | this is a very good movie . |
| $v + 0.76$ | this is a very good movie . |
| $v + 0.9$ | this is a very good movie . |
| $v + 1.03$ | this is a movie . |
| $v + 1.17$ | this is a movie . |
| $v + 1.31$ | this is a movie . |

TABLE VII: Reconstructed text by DAAE for different vector values. Original vector value is $v$ and 20 points were created by element-wise addition of a constant. The constants were chosen based on Figure 4.




(a) Sigmoid

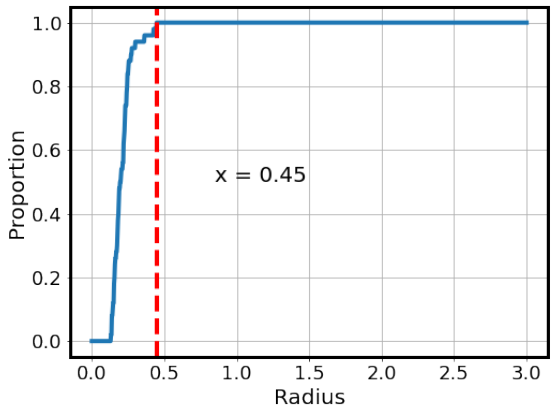



(b) Hyperbolic tangent

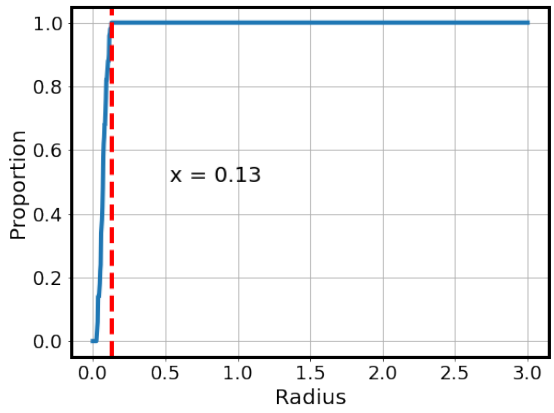
Fig. 10: PWL approximation of functions with 10 line segments over the interval $[-5, 5]$. Values of $x$ outside the specified interval are equal to the last encountered $f(x) = y$ value.

| Original | Original reconstruction | Counter-example reconstruction |
|---|---|---|
| rented the video with a lot of expectations , but it was a disappointment . | the movie was a good movie . | the movie was a good movie . |
| this was essentially made for tv and it shows . | this movie is not a waste of time . | this is a very good film with a lot of the film . |
| not sure if this is just a lousy movie or if it was intended to be a mockery of a “ b ” western . | i was not a fan of this film . | the movie is a very good movie , but it ’s a lot of the <unk> . |
| i cant believe how bad it was.i give this movie a 2 out of 10 . | i was not sure that i was a fan of this movie . | the movie was n’t a lot of the movie i ’ve seen it . |
| the movie has some points , but , if you want to make it a worthwhile movie , i suggest that you become what the main characters are called ... stoners.2/10! | the movie is a very good film | the movie was a very good movie , but it ’s not a lot of the worst movie ever made . |
| god , i never felt so insulted in my whole life than with this crap . | i ’m not sure that i was not a fan of this film . | the only thing that the movie was a bit of the <unk> , but the story is a very good movie . |
| i rather watch carrot top do a george bush impression than watch this no-skill hack . | i am not sure that the movie was a good movie . | the movie was a bit of the <unk> and the <unk> of the film . |
| it commits the mortal sin of being boring and not fun in the slightest . | it ’s not even more like the film . | the story is a good film , but the story is a good film . |
| the plot is ridiculous and the characters are horrible people . | the plot is not very good . | the film is a very good film . |
| throughout the film i ca n’t seem to find a connection or for that matter , sympathy with the characters , perhaps thats because they do n’t develop one throughout the film , character that is . | the movie is n’t that bad . | the movie is a very good film , but it ’s not a lot of the <unk> . |

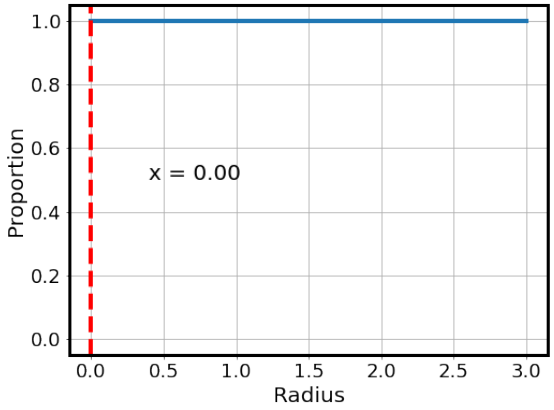
TABLE VIII: Reconstructions performed by DAAE of 10 sentences and their counter-examples found by Reluplex.




(a) GloVe encoding

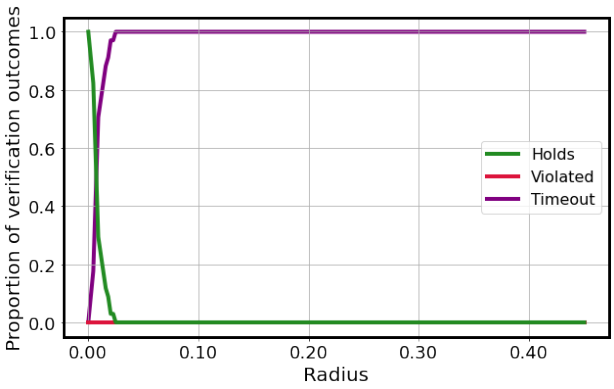



(b) Doc2Vec encoding

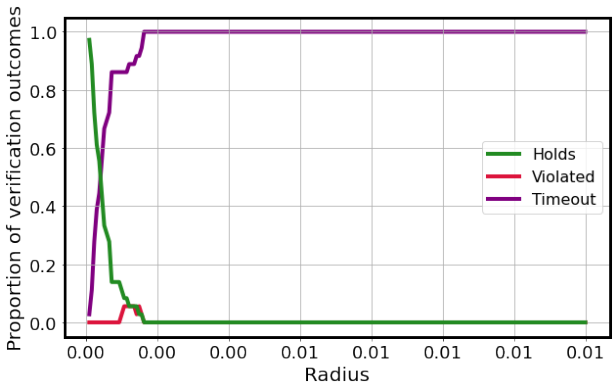



(c) InferSent encoding

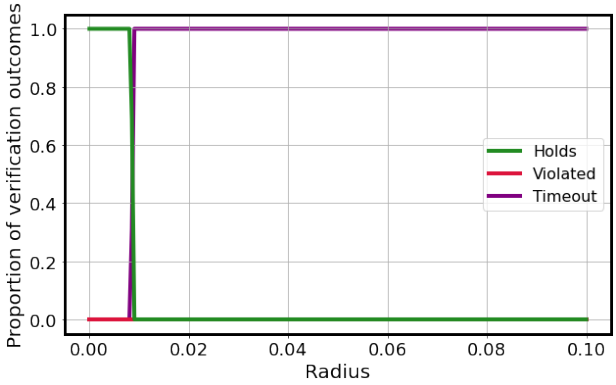
Fig. 11: Plots showing the proportion of 50 cherry-picked verification data points that have at least one data point from the training set with at most specified Chebyshev distance away from them. The red dotted line shows the maximum of minimum distances needed to reach at least one other data point appearing in the training set for each of the verification sentences.




(a) GloVe encoding

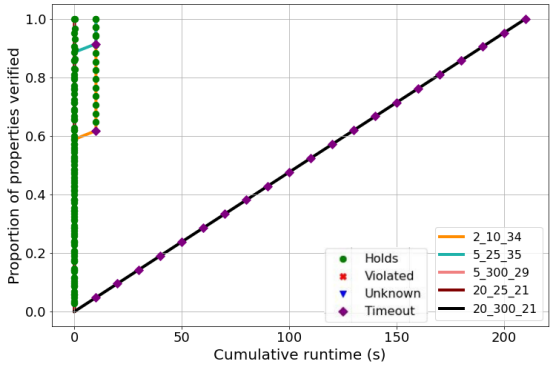



(b) Doc2Vec encoding

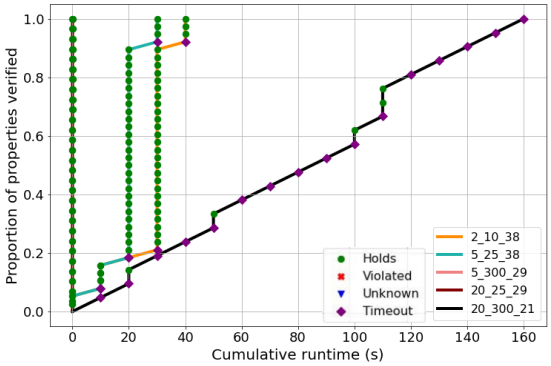



(c) InferSent encoding

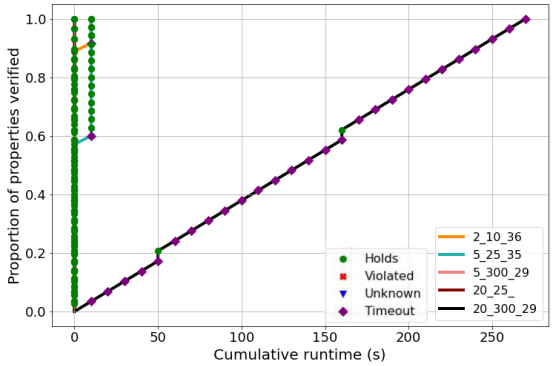
Fig. 12: Proportion of properties and their outcomes found by DLV [32] as the radius of the input hyper-rectangle is increased. Proportion calculated for 200 points between 0.0001 and minimum distance that incorporates at least one other point in the bounded hyper-rectangle as shown in Figure 4. Time out of 10 seconds imposed per property.



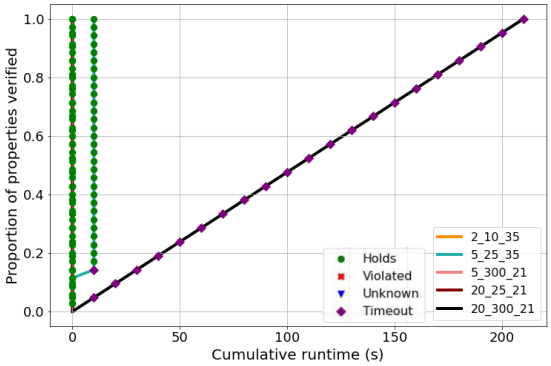

(a) GloVe encoding



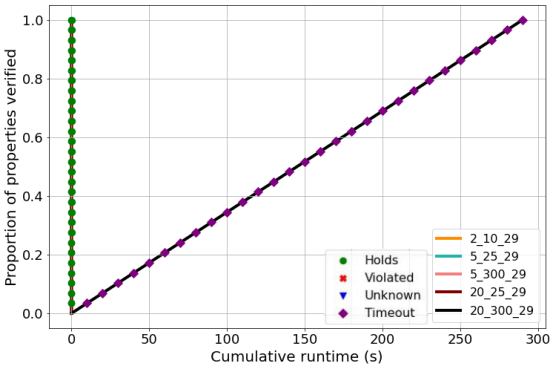

(b) FastText encoding



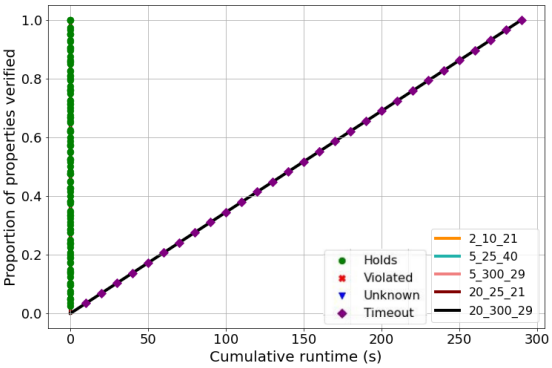

(c) Doc2Vec encoding




(d) USE encoding

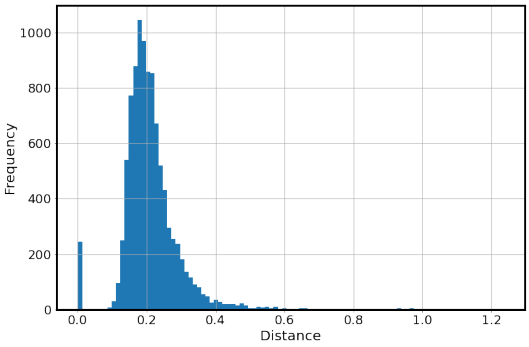



(e) InferSent encoding

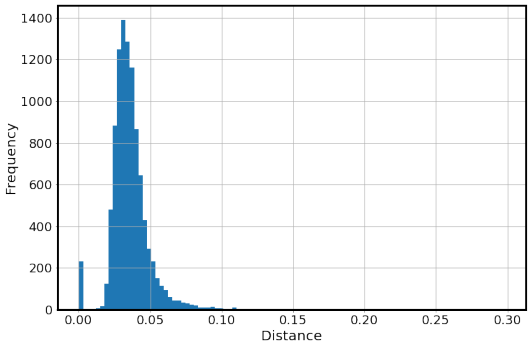



(f) DistilRoBERTa encoding

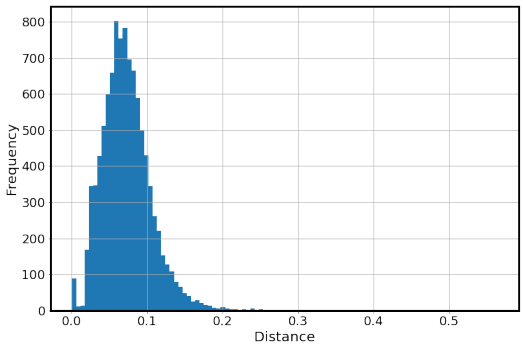
Fig. 13: Verification results of DLV solver for different sizes of ReLU networks. Input hyper-rectangle radius set to 0.0002. Timeout of 10 seconds per property imposed. The format of $x\_y\_z$ in the legend corresponds to the following properties of NNs: $x$ - number of layers, $y$ - number of nodes per layer, $z$ - number of examples correctly classified out of the 50 cherry-picked sentences.



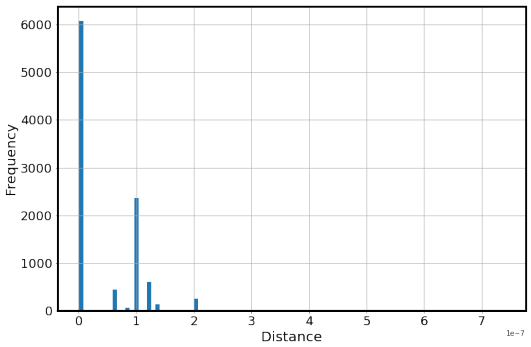

(a) GloVe encoding



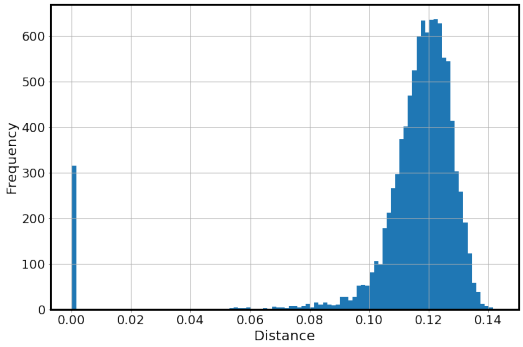

(b) FastText encoding



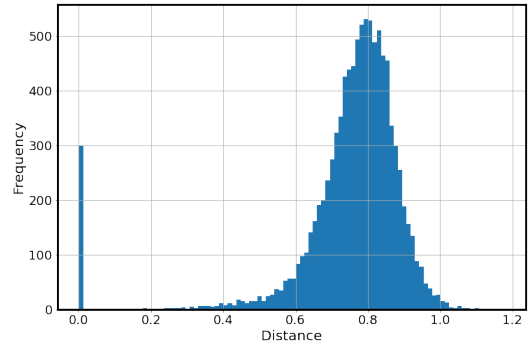

(c) Doc2Vec encoding



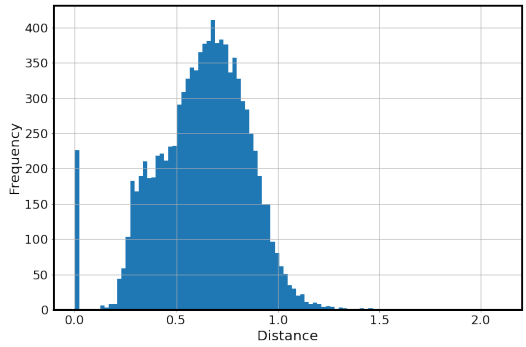

(d) InferSent encoding




(e) USE encoding




(f) DistilRoBERTa encoding




(g) DAAE encoding

Fig. 14: Histograms of nearest neighbour distances for 10k points from the training dataset. Distances were retrieved by training a k-NN algorithm for 100k points from the training set, except for InferSent, which was trained on 20k points.

| **Input: the plot is ridiculous and the characters are horrible people .** | |
|---|---|
| Text | $\delta_o$ |
| **GloVe** | |
| when something happens , the reactions of the characters are vague and dry.best not to look this one up . | 0.231 |
| not only is the plot involving and the characters fascinatingly drawn , but the setting is absolutely out of this world ! | 0.243 |
| the plot twists are top-notch , and one of the other great twists in this movie is that some of the supporting characters actually act as if they have brains . | 0.245 |
| what little there is , is atrocious to begin with , and made much worse by the terrible video and editing.the worst part of this atrocity , though , apart from the plot , would have to be the effects ... or rather the disturbing lack thereof . | 0.249 |
| the story is silly ( even for a fantasy ) , the kids are terrible actors and one of them ( charles ) is incredibly obnoxious . | 0.252 |
| **FastText** | |
| the movie is boring , the characters and scenarios are unrealistic , unbelievable , the action is hilarious . | 0.030 |
| the problem with the realism is that the characters are so patently unrealistic and atypical - contrary to the fetid imaginings of “ extreme ” filmmakers most teenagers are not drug addled rapists . | 0.033 |
| the rescue scenes are great - even if the computer generation is hokey and the scenarios are pretty unreal - but the backstory is lame and disappointing . | 0.033 |
| most of he supporting performances are hilariously amateurish , the cinematography is terrible and the locations and sceneries are beneath contempt . | 0.034 |
| the actors are reduced to macho posturing , the plot rings false , the action sequences are soulless and suspenseless , the dialogue is absurd even the violence becomes numbingly predictable . | 0.034 |
| **Doc2Vec** | |
| people do n’t act like this . | 0.036 |
| cats jumping on people . | 0.042 |
| mumbling and people wandering wistfully . . | 0.042 |
| stop ! | 0.044 |
| hardly . | 0.044 |
| **USE** | |
| the characters are awful , as is the story . . | 0.089 |
| the characters are uniformly unpleasant , and plot makes no sense . | 0.093 |
| the actors are not that bad , but their characters are rather dumb and the story is boring and downright stupid . | 0.096 |
| the acting is horrible , the plot ( what plot ) is stupid and degrading and insane . . | 0.099 |
| the plot and characters are ridiculous and barely qualify as “ plot ” and “ character ” . | 0.103 |
| **InferSent** | |
| two hundred years later , insomnia returns , dawn is reincarnated as enigma , and insomnia has returned to destroy zu . | 0.000 |
| the acting was so bad that i was hoping that one and all would be buried at the end . | 0.000 |
| there is no action , no suspense , not even a spark between the 2 leading actors . | 0.000 |
| this movie could have been titled “ beverly hills cop and the temple of doom ” since parts of this movie plays like a spielbergian adventure , kinda like an indiana jones comedy . | 0.000 |
| the characters were not developed well . | 0.000 |
| **DistilRoBERTa** | |
| the plot is ridiculous and the whole “ little man ” crap is just so stupid . | 0.590 |
| the plot is extremely ridiculous ; the characters are insufferably dumb , the gore-factor is negligible and the whole thing is just plain boring ! | 0.612 |
| the plot is ridiculous , the characters poorly developed , and the premise irritatingly stupid . | 0.650 |
| also , this was a shallow movie with weak acting , a predictable plot line and characters who are less than memorable. | 0.665 |
| the reason that this is so terrible is not because it deviated from the formula , but because the plot was just pathetic ? | 0.674 |
| **DAAE** | |
| the acting is first class and the characters are represented well . | 0.629 |
| the dancing is perfect , and so are the special effects . | 0.771 |
| the humor was weak and the characters fairly flat . | 0.807 |
| the other char-s are very nice also . | 0.875 |
| the characters are awful , as is the story . | 0.877 |

TABLE IX: 5 nearest neighbours of a specific sentence. $\delta_o$ is the Chebyshev distance to the original point.

| **Input: the plot is ridiculous and the characters are horrible people .** | | |
|---|---|---|
| Text | $\delta_c$ | $\delta_o$ |
| **GloVe, radius = 0.05** | | |
| the script is bad , the zombies are awful , there is no tension , lines are bad , actors are bad .. the list just goes on.you will probably want to see this movie just because of its reputation of being awful . | 0.252 | 0.296 |
| and the worst part is that all the above mentioned statements are true ! ! ! | 0.257 | 0.298 |
| the only thing that is good in this horrible mess are the excerpts of the jerry goldsmith score of bi1 . | 0.261 | 0.295 |
| what makes this idea fail is that right in the middle of some great 80 's duran duran songs , confusing and annoying cut scenes take place showing the fictional antagonist trying to stop the band at one of their concerts . | 0.261 | 0.311 |
| the jokes are lame as ... and the plot is ridiculous . | 0.262 | 0.291 |
| **FastText, radius = 0.004** | | |
| the movie is boring , the characters and scenarios are unrealistic , unbelievable , the action is hilarious . | 0.030 | 0.032 |
| the actors are reduced to macho posturing , the plot rings false , the action sequences are soulless and suspenseless , the dialogue is absurd even the violence becomes numbingly predictable . | 0.032 | 0.034 |
| the creature is ludicrous and its victims are simply despicable . | 0.033 | 0.037 |
| is it the acting , or the script that is bad , or both ? the protagonist is also highly unbelievable for social realism - ravenously consuming canonical english literature and the bible while high or hungover and able to produce such profoundly sophomoric soliloquies while intoxicated ? | 0.035 | 0.039 |
| yes , the storyline has potential but the dialogs are flat , the actors unconvincing . | 0.036 | 0.035 |
| **Doc2Vec, radius = 0.018** | | |
| people do n't act like this . | 0.050 | 0.036 |
| mumbling and people wandering wistfully . | 0.051 | 0.042 |
| the actors are incredible and the documentary style is superb . | 0.051 | 0.065 |
| and 3 ) people should n't like pokemon . | 0.052 | 0.059 |
| all the good guys are african americans . | 0.053 | 0.053 |
| **USE, radius = 0.014** | | |
| the plot and characters are ridiculous and barely qualify as " plot " and " character " . | 0.100 | 0.103 |
| the characters are awful , as is the story . | 0.100 | 0.089 |
| the acting is horrible , the plot ( what plot ) is stupid and degrading and insane . | 0.107 | 0.099 |
| the characters are uniformly unpleasant , and plot makes no sense . | 0.107 | 0.093 |
| the actors are not that bad , but their characters are rather dumb and the story is boring and downright stupid . | 0.108 | 0.096 |
| **DistilRoBERTa, radius = 0.164** | | |
| the characters are shallow and trite as are the dialog and plot line . | 0.703 | 0.867 |
| the plot is unbelievable . | 0.727 | 0.722 |
| the plot is totally cool , and the characters are excellently written . | 0.744 | 0.813 |
| the plot is ridiculous and the whole " little man " crap is just so stupid . | 0.744 | 0.590 |
| do you like when a plot contains unrealistic choices by the characters and is boring and lacks any kind of tension .. ? | 0.765 | 0.881 |
| **DAAE, radius = 1.31** | | |
| the acting is first class and the characters are represented well . | 1.939 | 0.629 |
| the actors were n't bad , but the plot needs more innovation . | 1.989 | 0.970 |
| the colors and backgrounds were just as bad as the effects . | 2.001 | 1.238 |
| the entire cast is wonderful and all the episopes have good plots . | 2.002 | 1.123 |
| the comedy is n't funny and the tragedy is n't very tragic . | 2.022 | 0.949 |

TABLE X: 5 nearest neighbours retrieved for the counter-example found by Reluplex. $\delta_c$ is the Chebyshev distance to the counter-example and $\delta_o$ is the Chebyshev distance to the original point for which the counter-example was found.

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 66.14 | 65.01 | 68.34 | 40.98 |
| Tanh | 2_10 | 66.67 | 65.32 | 68.08 | 41.06 |
| PWL Tanh | 2_10 | 66.01 | 64.35 | **68.98** | 41.05 |
| Sigmoid | 2_10 | 66.81 | 65.14 | 68.69 | 40.99 |
| PWL Sigmoid | 2_10 | **66.84** | 65.24 | 68.38 | 41.06 |
| ReLU | 5_25 | 66.46 | **66.12** | 67.5 | 41.22 |
| Tanh | 5_25 | 51.61 | 51.16 | 0.64 | 41.23 |
| PWL Tanh | 5_25 | 50.62 | 51.2 | 0.44 | 42.06 |
| Sigmoid | 5_25 | 50.93 | 48.83 | 65.62 | 41.22 |
| PWL Sigmoid | 5_25 | 50.98 | 48.83 | 65.62 | 41.91 |
| ReLU | 5_300 | 51.1 | 51.18 | 0.08 | 45.15 |
| Tanh | 5_300 | 50.72 | 51.34 | 1.5 | 44.03 |
| PWL Tanh | 5_300 | 49.98 | 51.15 | 0.14 | 47.33 |
| Sigmoid | 5_300 | 50.17 | 51.17 | nan | 47.28 |
| PWL Sigmoid | 5_300 | 49.94 | 51.17 | nan | 48.39 |
| ReLU | 20_25 | 50.9 | 48.83 | 65.62 | 42.56 |
| Tanh | 20_25 | 50.25 | 51.15 | 0.06 | 42.96 |
| PWL Tanh | 20_25 | 50.29 | 51.15 | 0.08 | 46.11 |
| Sigmoid | 20_25 | 50.92 | 48.83 | 65.62 | 42.83 |
| PWL Sigmoid | 20_25 | 50.99 | 48.83 | 65.62 | 46.05 |
| ReLU | 20_300 | 50.53 | 48.83 | 65.62 | 59.91 |
| Tanh | 20_300 | 50.16 | 51.17 | nan | 58.97 |
| PWL Tanh | 20_300 | 50.18 | 48.91 | 65.61 | 77.04 |
| Sigmoid | 20_300 | 50.12 | 51.17 | nan | 69.71 |
| PWL Sigmoid | 20_300 | 50.18 | 51.17 | nan | 82.75 |

TABLE XI: Metrics of deep neural networks trained on GloVe encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 64.14 | 62.68 | 63.9 | 21.8 |
| Tanh | 2_10 | 64.17 | 62.71 | 62.85 | 21.99 |
| PWL Tanh | 2_10 | 64.03 | 62.86 | 62.71 | 22.13 |
| Sigmoid | 2_10 | 64.49 | **62.89** | 63.57 | 21.8 |
| PWL Sigmoid | 2_10 | **64.59** | 62.83 | 63.74 | 21.94 |
| ReLU | 5_25 | 64.27 | 62.97 | 59.39 | 22.17 |
| Tanh | 5_25 | 50.14 | 48.84 | 65.6 | 22.21 |
| PWL Tanh | 5_25 | 59.17 | 60.47 | 57.81 | 22.87 |
| Sigmoid | 5_25 | 50.59 | 51.17 | nan | 22.14 |
| PWL Sigmoid | 5_25 | 50.62 | 51.17 | nan | 22.84 |
| ReLU | 5_300 | 54.25 | 51.17 | 0.01 | 25.34 |
| Tanh | 5_300 | 50.05 | 48.83 | **65.62** | 25.05 |
| PWL Tanh | 5_300 | 50.15 | 50.85 | 58.88 | 28.33 |
| Sigmoid | 5_300 | 49.87 | 48.83 | 65.62 | 35.44 |
| PWL Sigmoid | 5_300 | 49.98 | 48.83 | 65.62 | 28.77 |
| ReLU | 20_25 | 50.75 | 51.17 | nan | 23.53 |
| Tanh | 20_25 | 50.16 | 51.17 | nan | 23.86 |
| PWL Tanh | 20_25 | 50.17 | 51.19 | 0.4 | 26.98 |
| Sigmoid | 20_25 | 50.71 | 51.17 | nan | 23.73 |
| PWL Sigmoid | 20_25 | 50.62 | 51.17 | nan | 27.09 |
| ReLU | 20_300 | 50.42 | 51.17 | nan | 41.42 |
| Tanh | 20_300 | 50.11 | 50.93 | 10.28 | 39.05 |
| PWL Tanh | 20_300 | 49.99 | 48.83 | 65.62 | 58.41 |
| Sigmoid | 20_300 | 49.93 | 48.83 | 65.62 | 51.22 |
| PWL Sigmoid | 20_300 | 49.87 | 48.83 | 65.62 | 64.79 |

TABLE XIII: Metrics of deep neural networks trained on Doc2Vec encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 67.56 | 67.5 | 67.24 | 38.89 |
| Tanh | 2_10 | 68.66 | **67.89** | 69.01 | 39.03 |
| PWL Tanh | 2_10 | 68.0 | 67.56 | 69.04 | 38.12 |
| Sigmoid | 2_10 | **68.72** | 67.79 | 68.63 | 39.02 |
| PWL Sigmoid | 2_10 | 67.88 | 66.75 | 69.14 | 39.02 |
| ReLU | 5_25 | 67.37 | 67.07 | 67.66 | 39.15 |
| Tanh | 5_25 | 50.13 | 51.17 | nan | 38.54 |
| PWL Tanh | 5_25 | 50.29 | 51.17 | nan | 38.98 |
| Sigmoid | 5_25 | 68.68 | 67.36 | **69.26** | 38.93 |
| PWL Sigmoid | 5_25 | 50.72 | 51.17 | nan | 40.05 |
| ReLU | 5_300 | 51.06 | 51.17 | nan | 43.54 |
| Tanh | 5_300 | 49.99 | 51.17 | 0.01 | 42.23 |
| PWL Tanh | 5_300 | 50.02 | 51.17 | nan | 45.33 |
| Sigmoid | 5_300 | 50.09 | 51.17 | nan | 43.84 |
| PWL Sigmoid | 5_300 | 50.06 | 51.17 | nan | 46.87 |
| ReLU | 20_25 | 50.65 | 51.17 | nan | 40.15 |
| Tanh | 20_25 | 50.09 | 51.17 | nan | 40.96 |
| PWL Tanh | 20_25 | 50.15 | 51.17 | 0.0 | 44.53 |
| Sigmoid | 20_25 | 50.72 | 51.17 | nan | 40.57 |
| PWL Sigmoid | 20_25 | 50.72 | 51.17 | nan | 43.79 |
| ReLU | 20_300 | 50.38 | 48.83 | 65.62 | 60.69 |
| Tanh | 20_300 | 50.16 | 48.83 | 65.62 | 55.98 |
| PWL Tanh | 20_300 | 50.15 | 48.83 | 65.62 | 75.69 |
| Sigmoid | 20_300 | 50.0 | 51.17 | nan | 123.61 |
| PWL Sigmoid | 20_300 | 50.07 | 51.17 | nan | 83.89 |

TABLE XII: Metrics of deep neural networks trained on FastText encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 69.81 | 68.91 | **69.61** | 61.17 |
| Tanh | 2_10 | 69.78 | 68.7 | 68.76 | 62.26 |
| PWL Tanh | 2_10 | 69.9 | 68.96 | 68.34 | 62.23 |
| Sigmoid | 2_10 | 70.65 | 69.08 | 69.02 | 61.11 |
| PWL Sigmoid | 2_10 | **70.65** | **69.17** | 69.2 | 61.25 |
| ReLU | 5_25 | 70.3 | 69.13 | 69.07 | 62.4 |
| Tanh | 5_25 | 66.96 | 67.44 | 68.18 | 61.75 |
| PWL Tanh | 5_25 | 68.01 | 67.91 | 64.34 | 62.42 |
| Sigmoid | 5_25 | 70.52 | 68.86 | 66.1 | 62.44 |
| PWL Sigmoid | 5_25 | 50.71 | 48.83 | 65.62 | 62.98 |
| ReLU | 5_300 | 50.1 | 48.83 | 65.62 | 66.15 |
| Tanh | 5_300 | 62.46 | 60.97 | 63.23 | 66.6 |
| PWL Tanh | 5_300 | 58.24 | 58.36 | 38.63 | 70.99 |
| Sigmoid | 5_300 | 49.99 | 51.17 | nan | 98.22 |
| PWL Sigmoid | 5_300 | 49.94 | 51.17 | nan | 70.3 |
| ReLU | 20_25 | 50.49 | 48.83 | 65.62 | 64.58 |
| Tanh | 20_25 | 49.91 | 51.17 | nan | 63.63 |
| PWL Tanh | 20_25 | 50.39 | 49.0 | 52.29 | 66.9 |
| Sigmoid | 20_25 | 50.71 | 48.83 | 65.62 | 63.87 |
| PWL Sigmoid | 20_25 | 50.56 | 48.83 | 65.62 | 68.66 |
| ReLU | 20_300 | 50.51 | 48.83 | 65.62 | 84.78 |
| Tanh | 20_300 | 49.93 | 51.46 | 1.64 | 81.53 |
| PWL Tanh | 20_300 | 49.96 | 51.17 | nan | 101.12 |
| Sigmoid | 20_300 | 50.0 | 51.17 | nan | 91.42 |
| PWL Sigmoid | 20_300 | 50.03 | 51.17 | nan | 109.37 |

TABLE XIV: Metrics of deep neural networks trained on USE encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 50.8 | 51.17 | nan | 472.38 |
| Tanh | 2_10 | 50.38 | 51.17 | nan | 472.03 |
| PWL Tanh | 2_10 | 50.42 | 51.17 | nan | 471.73 |
| Sigmoid | 2_10 | 50.43 | 51.17 | nan | 474.49 |
| PWL Sigmoid | 2_10 | 50.43 | 51.17 | nan | 475.08 |
| ReLU | 5_25 | 50.8 | 51.17 | nan | 475.11 |
| Tanh | 5_25 | 50.21 | 48.83 | **65.61** | 475.4 |
| PWL Tanh | 5_25 | 50.05 | 48.83 | 65.61 | 476.02 |
| Sigmoid | 5_25 | 50.65 | 51.17 | nan | 479.17 |
| PWL Sigmoid | 5_25 | 50.65 | 51.17 | nan | 479.1 |
| ReLU | 5_300 | 50.95 | 51.17 | nan | 480.31 |
| Tanh | 5_300 | 50.17 | 48.83 | 65.61 | 484.9 |
| PWL Tanh | 5_300 | 50.0 | 51.17 | nan | 490.86 |
| Sigmoid | 5_300 | 50.14 | 48.83 | 65.61 | 486.5 |
| PWL Sigmoid | 5_300 | 50.07 | 48.83 | 65.61 | 489.41 |
| ReLU | 20_25 | 50.66 | 51.17 | nan | 477.97 |
| Tanh | 20_25 | 50.14 | 48.83 | 65.61 | 472.7 |
| PWL Tanh | 20_25 | 50.14 | 48.83 | 65.61 | 474.73 |
| Sigmoid | 20_25 | 50.64 | 51.17 | nan | 475.97 |
| PWL Sigmoid | 20_25 | 50.64 | 51.17 | nan | 482.97 |
| ReLU | 20_300 | **51.15** | **51.17** | nan | 505.8 |
| Tanh | 20_300 | 50.01 | 51.17 | nan | 503.3 |
| PWL Tanh | 20_300 | 50.18 | 48.83 | 65.61 | 522.08 |
| Sigmoid | 20_300 | 50.12 | 48.83 | 65.61 | 523.74 |
| PWL Sigmoid | 20_300 | 50.08 | 48.83 | 65.61 | 532.09 |

TABLE XV: Metrics of deep neural networks trained on InferSent encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 50.66 | 48.83 | 65.61 | 91.56 |
| Tanh | 2_10 | 69.05 | 68.73 | 66.25 | 93.22 |
| PWL Tanh | 2_10 | 69.81 | **69.12** | 67.6 | 91.95 |
| Sigmoid | 2_10 | 70.04 | 68.96 | 64.58 | 91.73 |
| PWL Sigmoid | 2_10 | **70.1** | 68.83 | 66.78 | 91.7 |
| ReLU | 5_25 | 67.6 | 65.32 | **70.85** | 92.19 |
| Tanh | 5_25 | 67.64 | 61.37 | 41.67 | 92.21 |
| PWL Tanh | 5_25 | 66.62 | 67.09 | 67.06 | 92.74 |
| Sigmoid | 5_25 | 69.76 | 68.8 | 64.66 | 92.34 |
| PWL Sigmoid | 5_25 | 50.62 | 48.83 | 65.62 | 93.22 |
| ReLU | 5_300 | 50.53 | 51.17 | 0.01 | 97.2 |
| Tanh | 5_300 | 50.06 | 51.17 | 0.0 | 95.88 |
| PWL Tanh | 5_300 | 50.16 | 51.17 | 0.03 | 101.01 |
| Sigmoid | 5_300 | 50.06 | 51.17 | nan | 97.56 |
| PWL Sigmoid | 5_300 | 50.07 | 51.17 | nan | 100.88 |
| ReLU | 20_25 | 50.59 | 48.83 | 65.62 | 94.77 |
| Tanh | 20_25 | 49.97 | 51.17 | nan | 95.41 |
| PWL Tanh | 20_25 | 49.97 | 51.17 | 1.64 | 98.67 |
| Sigmoid | 20_25 | 50.64 | 48.83 | 65.62 | 95.78 |
| PWL Sigmoid | 20_25 | 50.64 | 48.83 | 65.62 | 98.38 |
| ReLU | 20_300 | 50.54 | 51.17 | nan | 124.69 |
| Tanh | 20_300 | 50.02 | 51.15 | 0.09 | 113.96 |
| PWL Tanh | 20_300 | 49.96 | 51.17 | nan | 132.4 |
| Sigmoid | 20_300 | 50.0 | 51.17 | nan | 131.28 |
| PWL Sigmoid | 20_300 | 50.03 | 51.17 | nan | 141.88 |

TABLE XVI: Metrics of deep neural networks trained on DistilRoBERTa encodings

| Activation | Size | Train accuracy | Test accuracy | F1 score | Avg. train time per epoch (s) |
|---|---|---|---|---|---|
| ReLU | 2_10 | 50.67 | 51.17 | nan | 17.3 |
| Tanh | 2_10 | 50.29 | 48.83 | **65.62** | 17.64 |
| PWL Tanh | 2_10 | 50.3 | 48.83 | 65.62 | 17.32 |
| Sigmoid | 2_10 | 50.5 | 51.17 | nan | 17.34 |
| PWL Sigmoid | 2_10 | 50.3 | 48.83 | 65.62 | 17.42 |
| ReLU | 5_25 | 50.69 | 51.17 | nan | 17.94 |
| Tanh | 5_25 | 50.16 | 48.83 | 65.62 | 17.7 |
| PWL Tanh | 5_25 | 50.16 | 48.83 | 65.62 | 18.71 |
| Sigmoid | 5_25 | 50.68 | 51.17 | nan | 18.64 |
| PWL Sigmoid | 5_25 | 50.67 | 51.17 | nan | 18.72 |
| ReLU | 5_300 | **50.92** | **51.19** | 0.5 | 22.29 |
| Tanh | 5_300 | 50.05 | 51.17 | nan | 21.7 |
| PWL Tanh | 5_300 | 50.11 | 51.17 | nan | 25.93 |
| Sigmoid | 5_300 | 49.87 | 51.17 | nan | 30.79 |
| PWL Sigmoid | 5_300 | 49.89 | 51.17 | nan | 26.02 |
| ReLU | 20_25 | 50.69 | 51.17 | nan | 19.48 |
| Tanh | 20_25 | 50.09 | 48.83 | 65.62 | 19.99 |
| PWL Tanh | 20_25 | 50.16 | 48.83 | 65.62 | 23.3 |
| Sigmoid | 20_25 | 50.69 | 51.17 | nan | 19.61 |
| PWL Sigmoid | 20_25 | 50.67 | 51.17 | nan | 23.28 |
| ReLU | 20_300 | 50.26 | 48.83 | 65.62 | 39.82 |
| Tanh | 20_300 | 49.98 | 51.17 | nan | 37.09 |
| PWL Tanh | 20_300 | 49.99 | 48.83 | 65.62 | 56.46 |
| Sigmoid | 20_300 | 49.99 | 51.17 | nan | 47.81 |
| PWL Sigmoid | 20_300 | 49.88 | 51.17 | nan | 64.54 |

TABLE XVII: Metrics of deep neural networks trained on DAAE encodings