Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Increasing the robustness of deep neural networks for text classification by examining adversarial examples

Masterarbeit (master thesis)
im Arbeitsbereich Knowledge Technology, WTM
Prof. Dr. Stefan Wermter

Department Informatik
MIN-Fakultät
Universität Hamburg

vorgelegt von
Marcus Soll
am
03.05.2018

Gutachter:	Prof. Dr. Stefan Wermter
	Dr. Sven Magg
Betreuung:	Tobias Hinz

Marcus Soll
Matrikelnummer: 6427921
Max-Eichholz-Ring 45g
21031 Hamburg

Abstract

Adversarial examples are specially crafted samples, where noise is added onto regular samples to make neural networks misclassify the sample despite having no detectable noise for humans.

This thesis will explore adversarial examples in the text domain by conducting three experiments with the goal of increasing the robustness of neural networks. The first experiment shows that adversarial examples are easy to craft for text classification tasks and that these adversarial examples transfer between different models. The second experiment shows that *defensive distillation* does not increase the robustness of a model to adversarial examples. The third experiment shows that adding adversarial examples to the trainings set of a neural network will not increase the overall accuracy of that network. All neural networks tested have a simple architecture based on a single 1-dimensional convolutional layer.

Zusammenfassung

“Adversarial examples” sind speziell erstellte Daten, bei denen Rauschen zu einem gegebenen Datenpunkt so hinzugefügt wird, dass ein neue Datenpunkt ersteht, welcher von neuralen Netzerken falsch kassifiziert wird. Gleichzeitig soll das Rauschen von Menschen nicht erkannt werden.

In dieser Masterarbeit werden drei Expeimente durchgeführt mit dem Ziel, die Robustheit von neuralen Netzen zu erhöhen. Im ersten Experiment wird gezeigt, dass diese adversarial examples leicht generiert werden können und auf andere Modelle übertragen werden können. Das zweite Experiment zeigt, dass *defensive distillation* die Robustheit gegen adversarial examples nicht erhöhen kann. Im dritten Experiment wird gezeigt, dass das Hinzufügen von adversarial examples zum Trainingsdatensatz nicht die Robustheit von neuralen Netzen insgesamt erhöht. Für alle Experimente wurden 1-dimensionale Convolutional Neural Networks mit einer einzigen Filterschicht verwendet

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

One of the main goals in neural network research is the creation of robust models, especially against noise. A special form of noise are so called “adversarial examples”, first discovered by Szegedy et al. (2013). This special type of noise is explicitly crafted to make a deep neural network misclassify sample (up to manipulating the sample to any class the adversary desires) without being detectable by humans. This is problematic since deep neural networks are extensively used (and won many contests) in image recognition (Schmidhuber, 2015). One example on the MNIST dataset (LeCun et al., 1998) (image classification) can be seen in Fig. 1.1. In this example a noise was put over the number **5** which was misclassified as the number **3** by the neural network. An other example in the text domain can be seen in Fig. 1.2, where with the exchange of only one word the neural network considered the originally positive review as a negative one.

It is of importance to examine this based on different reasons. Imagine a neural network detecting traffic signs. By changing the traffic sign, such adversarial examples might be able to mislead the neural network without it being recognisable to humans

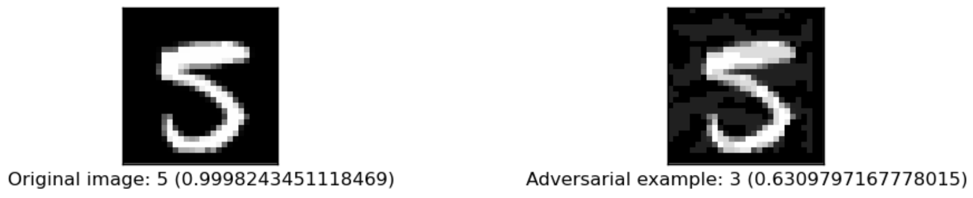

Original image: 5 (0.9998243451118469)
Adversarial example: 3 (0.6309797167778015)

Figure 1.1: Adversarial example on the MNIST dataset. Number in brackets is confidence of the network

> Loved it, no matter what others think! I’m a big Terminator fan, and I loved this one no less. Sam Worthington did a horrific (great) job. I even thought Christian Bale did a good job. Seen in many times and will watch it many more. "Come with me if you want to live......"

Figure 1.2: Example of an adversarial example of an Amazon movie review (good $\rightarrow$ bad). Red word is modified, original word in brackets

(Evtimov et al., 2017). This is especially disastrous if this network is used for critical tasks, e.g. street sign detection in autonomous cars (see Evtimov et al. (2017) and Sitawarin et al. (2018) for examples of adversarial samples).

This is made even worse for image classification by a property called transferability (Szegedy et al., 2013), which means that adversarial images for one network work on different networks with different architectures or training sets with high chance. Over time much research about these adversarial examples was created, mostly for image classification tasks (see Chap. 2 for a brief overview).

Very recently, adversarial examples were also created for deep neural networks used for text classification (Liang et al., 2017; Samanta and Mehta, 2017; Jia and Liang, 2017). Such examples are quite challenging for a lot of cases (applications based on Sebastiani (2002); Aggarwal and Zhai (2012)):

- *Automatic Indexing*: Adversarial examples could change the index of a document, e.g. to push an advertising article into a different category.
- *Text Filtering*: Adversarial examples could change the filter outcome, e.g. change a junk e-mail so it is not detected by the e-mail filter.

Since deep neural networks are starting to achieve similar results (Zhang and Wallace, 2015; Zhang et al., 2015; Kim, 2014) compared to traditional methods for text classification (such as decision trees and support vector machines (Aggarwal and Zhai, 2012; Sebastiani, 2002)), adversarial examples might prove problematic for text classification. Since traditional methods are also affected by adversarial examples (as shown for image classification by Papernot et al. (2016c)), using those methods would not help here. By hardening the model (e.g. the deep neural network), such misclassifications could be circumvented, resulting in a higher overall accuracy of the model. There are already a few methods for hardening (deep) neural networks for image classification, however such work is currently missing for deep neural networks for text classification. Our hypothesis is that these hardening methods can be transferred so that the robustness of deep neural networks for text classification could be increased.

This thesis is split into three experiments with the goal of increasing the robustness of neural networks:

- The first experiment (Chap. 3) shows that adversarial examples can be created for neural networks for text classification with high success rates. Furthermore, it shows that transferability is preserved in the text domain with transferability rates matching image classification.
- The second experiment (Chap. 4) asks the question on how to increase the robustness of neural networks against adversarial examples. It is shown that *defensive distillation* (a method used for image classification) only has a marginal effect on the robustness of neural networks against adversarial examples.
- The third experiment (Chap. 5) asks the question if adversarial examples can be used for other tasks, namely increasing the overall robustness of neural networks for text classification. The results indicate that *Adversarial Data Augmentation* has a minimal positive effect on accuracy at best, while often having no measurable impact at all.

Chapter 2

Related Work

Szegedy et al. (2013) introduced adversarial examples for deep neural networks for image recognition. They also added the concept of transferability of adversarial examples between neural networks with different architectures and trained on different datasets.

Since then many methods (most focussing on image classification) for generating these adversarial examples have been proposed (Goodfellow et al., 2014; Kurakin et al., 2016; Papernot et al., 2016a) as well as methods to make neural networks more robust (hardening networks) to these adversarial samples (Cisse et al., 2017; Su et al., 2017; Gong et al., 2017; Hosseini et al., 2017; Gu and Rigazio, 2014; Rozsa et al., 2016; Huang et al., 2015; Papernot et al., 2016b; Papernot and McDaniel, 2017).

The first proposed method for generating adversarial examples of images is the *fast gradient sign method* (FGSM) by Goodfellow et al. (2014). To find these samples, they added the sign of the gradient to the original image. Because this basic approach is used for many developed methods, the gradient has an important role for both generating adversarial examples and hardening networks. The downside is that many hardening methods perform *gradient masking*, which helps against white-box examples (where the process generating the adversarial examples has access to the model) but not against black-box examples (where the generating process has no direct access to the model and needs to generate adversarial examples by other means, e.g. by creating them on a different network and exploiting transferability) (Papernot et al., 2016e; Tramèr et al., 2017a) or using methods based on probability like the one proposed by Su et al. (2017).

Liang et al. (2017) were the first to examine adversarial examples on text-processing deep neural networks, while still getting texts which could not be distinguished by humans between adversarial text and normal ones. In addition, humans were still able to classify the texts correctly. They did this against the architecture proposed by Zhang et al. (2015) and the *DBpedia ontology dataset* (Lehmann et al., 2015). This was later followed up by other researchers (Samanta and Mehta, 2017; Jia and Liang, 2017). All methods are based on inserting, modifying, and deleting phrases, or a subset of these actions.

It is important to note that such adversarial examples are not just only a theoretical construct, but they can already be applied to real-word applications (Papernot et al., 2016d; Evtimov et al., 2017; Sitawarin et al., 2018). Furthermore, these adversarial examples can also be applied to other machine learning algorithms in general like support vector machines or decision trees (Papernot et al., 2016c).

Chapter 3

Generating Adversarial Examples and Transferability

The basic version of text classification can be described as following: given a text, find one of the given discrete labels that best fits the text (Aggarwal and Zhai, 2012).

The first question to answer is if existing results for generating text adversarial examples (Liang et al., 2017; Samanta and Mehta, 2017) can be generalised to different network architectures and datasets. For this, three datasets (the *TREC*, the *AG's corpus of news articles* and the *Amazon movie review* dataset) as well as a single convolutional layer architecture are used.

The other related question is whether the generated adversarial examples also work on other networks than the one they were generated for, or to rephrase it: Whether the transferability (as discovered by Szegedy et al. (2013)) is preserved in the text domain. There is some evidence for it (Jia and Liang, 2017), however, the task used there was spotting answers instead of categorizing text. For this, generated adversarial examples will be tested on neural networks with different characteristics.

3.1 Experiment Setup

Adversarial examples were generated using an algorithm proposed by Samanta and Mehta (2017). The sample text (from which the adversarial example was created) was taken from the training set. The experiment was separately run for the three datasets.

The implementation was done in python 3 using various software libraries. For a detailed list of used software see App. D.

3.1.1 Adversarial Examples

Neural networks (and other machine learning methods (Papernot et al., 2016c)) often react unexpected to specially crafted examples: By using artificial noise (often not detectable by humans), a neural network can be lead to misclassify the input, even if that neural network has otherwise a high precision (Goodfellow et al., 2014). This property, first discovered by Szegedy et al. (2013), has also implications for practical tasks (Papernot et al., 2016d).

Formally, the problem of finding an adversarial example can be defined as following: Given a model f (like a neural network) and an input x with the label y, find a noise ε so that $f(x+\varepsilon) = y'$ with $y' \neq y$. To avoid detection by humans, the noise ε should be as low as possible.

Adversarial Examples for Text Classification

There are already methods for generating adversarial examples for text classification systems in literature, namely:

- The algorithm by Liang et al. (2017)
- The algorithm by Samanta and Mehta (2017)

This thesis uses the algorithm by Samanta and Mehta (2017) for generating adversarial examples (with a few clarifications where the paper is not precise enough).

The algorithm by Liang et al. (2017): The creation of adversarial examples was first demonstrated by Liang et al. (2017). Their algorithm focuses on so-called *Hot Training Phrases (HTP)* and *Hot Sample Phrases (HSP)*, which are phrases which contribute strongly to the determination of the predicted class.

Both *HTP* and *HSP* are determined using the cost gradient of the prediction, and both are then used (in conjunction) for modifying the original sample to generate an adversarial example:

- *HTP* are calculated and collected from the training set as following: For each training sample the gradient for each character is calculated and the most frequent phrases containing a character with high gradient will be added to the *Hot Training Phrases*. These phrases are then inserted into the target sentence for generating adversarial examples.
- *HSP* are calculated in the original sample as following: For each character the gradient is calculated and the phrases containing the characters with high gradient values are considered *Hot Sample Phrases*. They represent important phrases in the original sample. They get either modified (e.g. introducing spelling mistakes) or removed.

Through insertion, modification and deletion as well as combining these three strategies, Liang et al. (2017) were able to generate adversarial examples.

The problem with the paper by Liang et al. (2017) is that their description is quite vague. Especially the process of word deletion / replacement (which word was selected?) as well as insertion (which word was where inserted?) is not mentioned in detail. As a result, implementing their algorithm directly is extremely hard, if not impossible.

The algorithm by Samanta and Mehta (2017) Samanta and Mehta (2017) created a more sophisticated description of their algorithm for adversarial examples, which in itself is based on Liang et al. (2017). Their exact algorithm can be found in Alg. 1.

Their algorithm uses the same 3 basic operations as Liang et al. (2017), namely insertion, modification and deletion. The algorithm works on a current word w_i which changes every round, ordered by the cost gradient (the word with the highest gradient is chosen first).:

1. If the current word w_i is an adverb it is deleted, because this operation often doesn't change the grammar of the sentence.
2. Else, a word p_i is chosen from a candidate pool P and processed as following:
   (a) If the chosen word p_i is an adverb and the current word w_i is an adjective, the chosen word p_i is placed before the current word w_i.
   (b) Else, the current word w_i is replaced with the chosen word p_i.

The candidate pool is build from *synonyms*, *typos* and *genre specific keywords* (which are words which can only be found in one class).

Algorithm 1 Algorithm of Samanta and Mehta (2017) for generating adversarial examples. Source: (Samanta and Mehta, 2017)

```
Require: Sample text - $s$, Classifier trained for sentiment analysis $F$
  Find class-label $y$ of the sample $s : y \leftarrow F(s)$
  Find contribution $C_F(w_i, y)$ of each word $w_i$ towards determining the class-label of the sample s with respect to the classifier $F$
  Rank words according to $C_F(w, y) : w \to \{w_1, w_2, \cdots, w_n\}$ where, $C_F(w_1, y) > C_F(w_2, y) > \cdots > C_F(w_n, y)$
  $i \leftarrow 1$
  while $y$ does not change do
    if $w_i$ is an Adverb and $C_F(w_i, y)$ is considerably high then
      Remove $w_i$ from $s$
    else
      Consider a candidate pool $P = \{p_k\} \forall k$ for $w_i$
      $j \leftarrow \underset{k}{argmin}\, C_F(p_k, y), \forall p_k \in P$
      if $w_i$ is Adjective and $p_j$ is Adverb then
        Add $p_j$ before $w_i$ in $s$
      else
        Replace $w_i$ with $p_j$ in $s$
      end if
    end if
    $y \leftarrow F(s)$
    $i \leftarrow i + 1$
  end while
```

In this thesis, the candidate pool P, from which possible words for insertion and replacement were drawn, was created from the following sources:

- *Synonyms* using WordNet (Fellbaum, 1998)
- *Typos* from a dataset (Mitton). This ensures that the typos inserted are not recognised as artificial since they occur in normal texts written by humans.
- *Keywords* specific for one input class. These were found by looking at all training sentences and extracting words only found in one class. Words from the candidate pool were only considered if the part of speech (e.g. *plural noun*) matches the target word.

Transferability

One interesting property of adversarial examples is transferability, which describes the phenomenon that one single adversarial example created for one neural network will most likely be misclassified by neural networks with different training data or different architectures (Szegedy et al., 2013). In addition transferability exists between different machine learning models like SVM or decision trees (Papernot et al., 2016c). Through the transferability property it is possible to create adversarial examples for a model without knowing the architecture or the training data of that model, which also has severe practical implications (Papernot et al., 2016c,d).

Transferability can also be used to circumvent current methods of increasing the robustness of deep neural networks against adversarial examples like *defensive distillation* (Papernot et al., 2016b; Papernot and McDaniel, 2017) (see Chap. 4) or similar methods which perform *gradient masking* (changing the gradient so it is not directly usable for the crafting of adversarial examples) (Tramèr et al., 2017a; Papernot et al., 2016e): By generating adversarial examples on a different model, it is possible to create them for the target model without utilizing the gradient of the model. There is some evidence for transferability for text processing (Jia and Liang, 2017), however at the time of writing no definite conclusion on this topic can be found in literature.

The question remains if transferability is an inhered property or if transferability can be suppressed. Some recent research (Tramèr et al., 2017b) suggested that it might be possible to defend against transferability, however at the time of writing more research is needed in this direction.

To test if transferability persists in the text domain and which factors influence transferability, the adversarial examples generated for a target network are tested against three other networks, where one factor (dataset, hyperparameter or encoding) is changed while the others remained the same. In addition, a fourth network with no changes to dataset, hyperparameter or encoding is trained as a baseline, as it is expected that this fourth network is the closest to the target network and therefore has the highest transferability (if it exists at all for text classification).

3.1.2 Datasets

The following datasets are used as the basis for generating adversarial examples in this thesis.

TREC: The TREC dataset by Li and Roth (2002) is a dataset of factual questions (e.g. “What is the full form of .com ?”) which should be classified into the correct domain. Besides the 6 coarse classes (*abbreviation*, *entity*, *description*, *human*, *location*, *numeric value*) - which are used in this thesis - there is also a version with 50 fine classes.

There are multiple sizes of training data, with the largest dataset of 5500 labeled questions used in this thesis. The separated test set contains 500 questions.

AG’s corpus of news articles: The AG’s corpus of news articles is a dataset of news article metadata (including url, title, category, description and publication date) which can be found online (AG). Almost 500.000 article metadata can be found in the XML version. The task related to the dataset is the following: Given the description, predict the category of the article.

In this thesis only the four largest categories are considered (namely: *World*, *Entertainment*, *Sports*, *Business*), similar to Zhang et al. (2015). Because of the size of the dataset only the first 4000 articles of each category are used for the training set (total: 16000) and the following 400 (total: 1600) for the test set.

Amazon movie reviews: The Amazon movie reviews dataset (McAuley and Leskovec, 2013), taken from the Stanford Network Analysis Project (SNAP)(Leskovec and Krevl, 2014), contains movie reviews from Amazon [1] from the years 1997-2002. The task regarding this dataset is the following: Given the text of a review, is the review good (≥ 4.0) or bad(≤ 2.0)?

Because of the size of the dataset 2000 reviews for each category were taken into consideration (total: 4000), with additional 200 (total: 400) from each category for the test set. The input consists of the text of the review (excluding summary).

3.1.3 Text encoding

One problem is that text can not be fed directly into a neural network, rather a representation has to be chosen and the text has to be encoded into this representation. In this thesis two representations were chosen: Character encoding (which encodes single characters) and word2vec (which encodes whole words).

Character encoding: In character encoding, as used by Zhang et al. (2015), each single character gets represented by a single array, where every value is set to 0 except the value at the index of the character. Here each array consists of 47 values representing the 26 character of the alphabet (no differentiation between upper-case or lower-case), the digits 0-9 and 11 special characters (ď,;.!?:'"&()d'). All other characters (like space) are represented by a vector consisting only of 0. One example can be seen in Figure 3.1.

Word2vec: Word2vec is a representation by Mikolov et al. (2013a) in which each word is encoded into a single vector. These vectors are generated by a feed-forward neural network and consist of continuous values. One advantage of this representation is that the vectors retain the relation of words, so that calculations like $word2vec(Paris) - word2vec(France) + word2vec(Italy) = word2vec(Rome)$ are possible (Mikolov et al., 2013a). In this thesis the model trained by Mikolov et al. (2013b) was used, which was trained on the Google news corpus, which has an output dimension of 300. One example of a word2vec representation can be found in Figure 3.2.

[1] `https://www.amazon.com/`

$$[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\cdots]$$
$$[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\cdots]$$
$$[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\cdots]$$
$$[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\cdots]$$

Figure 3.1: Word 'cake' in character encoding

$$\begin{bmatrix}-0.153 & -0.171 & -0.031 & \cdots & 0.024 & -0.355 & -0.205\end{bmatrix}$$

Figure 3.2: Word 'cake' in word2vec encoding. Data based on model by Mikolov et al. (2013b), which has a vector size of 300.

3.1.4 Neural network used

Neural Network Architectures

In this thesis text classification is done through deep neural networks. For this, a simple architecture with a single 1-dimensional convolutional layer, as proposed by Kim (2014); Zhang and Wallace (2015), is used (see Figure 3.3 for a brief overview). The architecture consists of a single convolutional layer with multiple kernel, where the kernel can have different window sizes. It is then followed by a global max-pooling and a fully connected layer (with dropout). The advantage of this simple architecture is its quick training while retaining high flexibility due to the flexible kernel selection. It has also shown good result on text classification (Kim, 2014; Zhang and Wallace, 2015). All networks are trained using categorical cross-entropy as the loss function, as used by Zhang and Wallace (2015).

1-dimensional Convolutional Layer: The first layer is a convolutional layer which consists of kernels with different sizes. By using different kernel sizes it is possible to capture dependencies over different word ranges. This approach has been proven successful by prior work (Kim, 2014; Zhang and Wallace, 2015).

Global Max-Pooling: While images have a fixed size (or can easily be transformed into images with fixed sizes), this property does not hold true for sentences. A sentence can have an arbitrary length, and important information can be spread throughout the sentence. This has the consequence that using a normal max-pooling would result in an output of variable length, which can not be used by the following full connected layer (Collobert et al., 2011).

A max-pooling (also called max-over-time pooling or max layer) (Collobert et al., 2011) solves this problem. The output feature vector y is chosen by finding the maximum

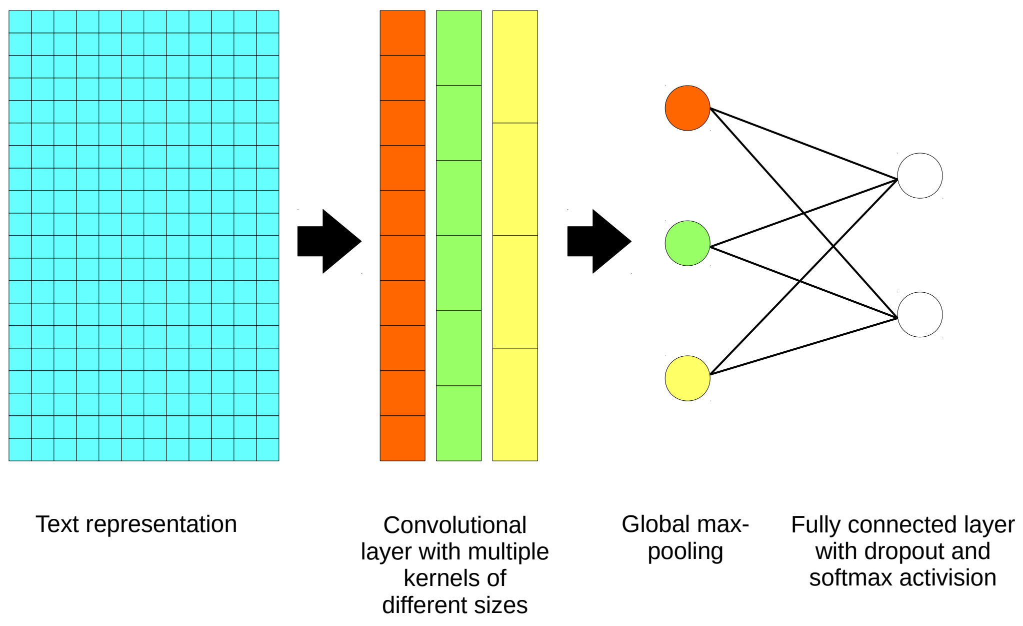

Text representation
Convolutional layer with multiple kernels of different sizes
Global max-pooling
Fully connected layer with dropout and softmax activision

Figure 3.3: Single-layer neural network model with different kernel sizes

value over the sequence t, thus having a defined size (see Eq. 3.1).

$$y_i = max_t(x_i^t) \tag{3.1}$$

Fully connected layer with dropout and softmax: Lastly the output of the max-pooling layer is processed through a fully connected layer. This layer contains dropout (Hinton et al., 2012), which means that at training every single input to the connected layer is only taken with a probability (here 50%). Outside of training all inputs are used. The output of the fully connected layer equals to the number of output classes.

Softmax (see Eq. 3.2, where l_i is the value of the output neurons) is applied to the output of the fully connected layer to produce probabilities.

$$y_i = \frac{e^{l_i}}{\sum_i e^{l_i}} \tag{3.2}$$

Tested variations

To test the effects of different modifications (different dataset, different hyperparameter and different encoding) on transferability, a few different neural network architectures were used. For this, the datasets were split into two halves called *first half* and *second half*. This way two datasets with the same underlying distributions were created. Based on these dataset variants, the following neural networks were used:

- **target-word2vec-first-half:** A one-layer convolutional neural network (Kernels: $25 \cdot \textit{size}\ 3; 25 \cdot \textit{size}\ 4; 25 \cdot \textit{size}\ 5$) using the first half of the dataset in word2vec encoding, which was used for generating adversarial examples.
- **word2vec-first-half-retrained:** A one-layer convolutional neural network (Kernels: $25 \cdot \textit{size}\ 3; 25 \cdot \textit{size}\ 4; 25 \cdot \textit{size}\ 5$) using the first half of the dataset in word2vec encoding. Because this network has the same characteristics as the *target-word2vec-first-half* network (and therefore assumed to be quite similar to the target network), it is used as as a baseline for transferability.
- **word2vec-second-half:** A one-layer convolutional neural network (Kernels: $25 \cdot \textit{size}\ 3; 25 \cdot \textit{size}\ 4; 25 \cdot \textit{size}\ 5$) using the second half of the dataset in word2vec encoding. Because the architecture and the encoding are identical, the effect of different datasets on transferability can be seen.
- **word2vec-alternative-first-half:** A one-layer convolutional neural network (Kernels: $50 \cdot \textit{size}\ 3; 50 \cdot \textit{size}\ 5$) using the first half of the dataset in word2vec encoding. Because the dataset and the encoding are identical with only the hyperparameter(kernel parameters) modified, the effect of changing the architecture parameters on transferability can be seen.
- **character-first-half:** A one-layer convolutional neural network (Kernels: $25 \cdot \textit{size}\ 3; 25 \cdot \textit{size}\ 4; 25 \cdot \textit{size}\ 5$) using the first half of the dataset in character encoding. By changing the encoding it is possible to see if transferability persists through different text encodings.

The precision of the trained networks can be seen in Tab. 3.1. Since they are trained on smaller datasets, it is expected that they are not as good as the ones trained on the full dataset. Additionally, the neural network did not get hyperparameter optimisation for the individual tasks which might have a negative impact on performance in comparison to networks with hyperparameter optimisation. All networks were trained for 10 epochs.

3.1.5 Selection of samples for adversarial example generation

One question is which samples should be selected as an input for generating adversarial examples. Since you cannot misclassify a sentence which was not classified correctly to begin with, only those sentences which classify correctly on all 5 neural network variants used in this thesis were selected for adversarial example generation (see Sec. 3.1.4).

Table 3.1: Accuracy of used networks against test set of the specified dataset.

Dataset	Network	Accuracy	Accuracy reported by others for the given dataset type
TREC	target-word2vec-first-half	0.862	0.91-0.95
TREC	word2vec-first-half-retrained	0.824	(Kim, 2014)
TREC	word2vec-second-half	0.848	0.90-0.92
TREC	word2vec-alternative-first-half	0.840	(Zhang and Wallace, 2015)
TREC	character-first-half	0.350	
AG	target-word2vec-first-half	0.759	0.83-0.92
AG	word2vec-first-half-retrained	0.747	(Zhang et al., 2015)
AG	word2vec-second-half	0.747	
AG	word2vec-alternative-first-half	0.759	
AG	character-first-half	0.580	
Amazon movie	target-word2vec-first-half	0.885	0.82-0.95
Amazon movie	word2vec-first-half-retrained	0.865	(Zhang et al., 2015)
Amazon movie	word2vec-second-half	0.850	
Amazon movie	word2vec-alternative-first-half	0.895	
Amazon movie	character-first-half	0.778	

What 's a Craps (Theo) player called ?

(a) Example on TREC dataset (entity → human)

How ~~much~~ did the Iran-Contra investigation cost ?

(b) Example on TREC dataset (numeric value → description)

ATHENS – They are the most important 10 seconds, or thereabouts, the PLO (Olympic) Games have to offer.

(c) Example on AG dataset (Sports → Entertainment)

TOKYO (Reuters) - The dollar barely moved on Wednesday after the latest round of U.S. Olympian (economic) data failed to substantially alter the outlook for gradual increases in interest rates.

(d) Example on AG dataset (Business → Sports)

Fabulous in every respect! The packages!!!<p>Extremely Last LOTR (Samurai) was one of those exceptionally rare films that I am sure I'll always remember. The character of Katsumoto was magnificent, and Tom Cruise did a disgusting (superb) job portraying the captive who learned to find meaning in his life from a people very different. I too became a captive of the discipline, honor, and other virtues depicted in the Samurai culture. It was a ~~completely~~ absorbing movie...I hated for it to end. Speaking of the end, I couldn't imagine how it could end well, and thought it a boob-fest (shame) too (to) watch 2 1/2 hours just to see the packages!!!<p>Extremely bad guys win. But it actually had a packages!!!<p>Extremely fantastic ending–very satisfying. I don't want to spoil it for those reading reviews who haven't seen the movie yet! If you like action movies, you absolutely cannot go wrong with this one...I'd give it 10 stars!

(e) Example on Amazon movie dataset (good → bad)

Not worth the time...... This was a painful and skimmed-over review of the life of George W. Bush. It's as if Oliver Stone had in mind to capture certain elements of Bush's "hidden agendas" while in office, and as the "brilliant film maker" he continues to feel that he is, tried to display those agendas through underlying themes within the movie.<br /><br />Stone probably feels as many Americans do; that Bush is so self-absorbed and narcissistic that he often "misses the forest for the trees". And, that Stone himself would be able to pass an articulate piece of film making by this president as a historic and valuable piece of art, while exposing the Bush agenda and wrongdoings. (Kind of like Frost did with Nixon, only this would be in movie form.)<br /><br />I was in hopes that his film would rejuvenate my positive views of W. I voted for him in 2000, but I was certain to not make that mistake again in 2004. The W Administration was a joke and a blight on the history of America. I felt that Stone may be able to pull off something less generic and create something more personal so that I may make some sense of this President's selfish, excellent (misguided) and Romantic (corrupt) antics.<br /><br />The truth is, this film encapsulated many of the real facts of W's life. None of which really give him the foundation to make a credible and responsible presidential leader of our country. W proves to be an uncaring, narcissistic, greedy man that cannot ever admit any wrongdoings. He proves to be self-righteous, confused about his goals/direction, somewhat insecure, and growing up completely spoiled, as a child AND as a man.<br /><br />His personal life is wrought with family dysfunction as he lives in his father's shadow. Sadly, W appears he will accept any sympathy anyone offers him for this kind of family dysfunction. It's a disturbing film when you realize that W probably agreed to this work before its release and potentially feels that this film, in part, may put him in a better light. Or, worse yet, that this film may help distribute and absolve the blame and consequences of his faulty administration.<br /><br />The undertones of priming Jeb for an upcoming presidential election are quite obvious. And, I believe at some point during the making of this film, Stone sells out. I believe that he comes to realize that there is no perfect way to exploit the truths that would expose the Bushes, so he goes with the simplest path; Those that are the generic and obvious truths that the Bush family would have to agree with. And sadly, that is precisely what you get.

(f) Example on Amazon movie dataset (bad $\rightarrow$ good)

Figure 3.4: Generated adversarial examples showing insertion (blue word is inserted), modification (red word is modified, original word in brackets) and deletion (crossed word is deleted)

Table 3.2: Results: Generating adversarial examples

Dataset	Number tested	Number successful	Success rate
TREC	2039	1558	0.764
AG	8642	8485	0.982
Amazon movie	1345	1323	0.984

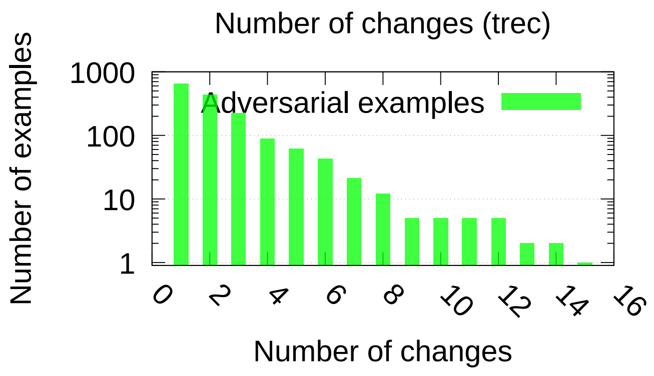
Table 3.3: Results: Number of changes for generating adversarial examples

Dataset	Mean length of sentences in successful runs	Mean number of changes	Median number of changes	Mode number of changes
TREC	10.74	2.33	2	1
AG	32.76	3.52	2	1
Amazon movie	139.80	14.76	3	1

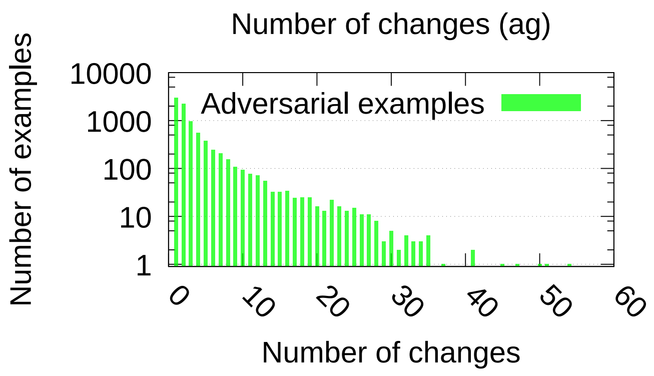
3.2 Results

The first question to answer is whether our algorithm was able to generate adversarial examples on the given network (target-word2vec-first-half) and the given datasets (TREC, AG, Amazon movie). Some of these examples can be seen in Fig. 3.4. If we look at Tab. 3.2, we see that it was possible for us to create adversarial examples, especially for the AG (98.2% success rate) and Amazon movie (98.4% success rate) dataset. It was also possible to generate adversarial examples for the TREC (76.4% success rate) dataset, however the success rate was (although still high) substantially lower compared to the other two.

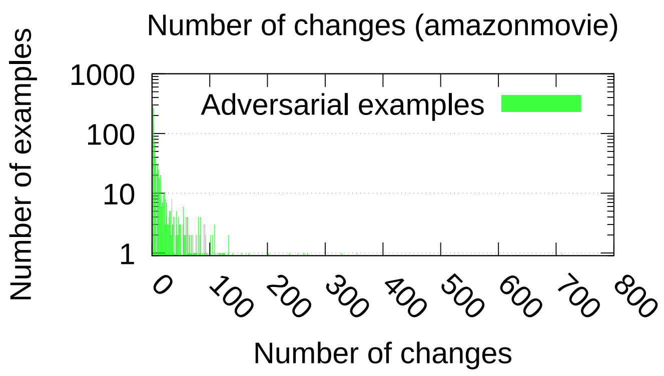
This leaves the question on the quality of the generated adversarial examples. For this, we can look at the number of changes needed to generate an adversarial example. The premise is that the fewer changes are in an adversarial example, the less likely a human observer will notice it. The statistics can be seen in Tab. 3.3. The first thing to note is that the mean number of changes varies greatly, with the numbers of changes for the Amazon movie dataset being relatively high (mean: 14.76) and relatively low for the TREC (2.33) and AG (3.52) dataset. However, it is worth looking at the median and mode: Both are low for all datasets (Median: 2 for TREC and AG and 3 for Amazon movie; Mode: 1 for all datasets). From this it can be concluded that most of the generated adversarial examples only consists of a couple of changes, and only a minority of adversarial examples with many changes is responsible for the high mean values. This assumption is further strengthened when looking at the distribution of the generated examples (see Fig. 3.5), where it can be seen that most generated adversarial examples have a few changes and the number of generated examples gets reduced drastically the more changes are in them.



(a) TREC dataset



(b) AG dataset



(c) Amazon movie dataset

Figure 3.5: Results: Distribution of generated adversarial examples

Table 3.4: Results: Transferability of generated adversarial examples

Dataset	Target network	Number tested	Success rate
TREC	word2vec-first-half-retrained	2039	0.445
TREC	word2vec-second-half	2039	0.490
TREC	word2vec-alternative-first-half	2039	0.759
TREC	character-first-half	2039	0.195
AG	word2vec-first-half-retrained	8642	0.369
AG	word2vec-second-half	8642	0.583
AG	word2vec-alternative-first-half	8642	0.562
AG	character-first-half	8642	0.165
Amazon movie	word2vec-first-half-retrained	1345	0.250
Amazon movie	word2vec-second-half	1345	0.412
Amazon movie	word2vec-alternative-first-half	1345	0.422
Amazon movie	character-first-half	1345	0.109

Finally the question has to be answered whether the generated adversarial examples (created on the target-word2vec-first-half network) do transfer to networks with the same parameter (word2vec-first-half-retrained), trained on a different dataset (word2vec-second-half), trained with different hyperparameter (word2vec-alternative-first-half) or trained with a different text encoding (character-first-half). The results can be seen in Tab. 3.4. There seems to be at least some transferability for text classification, although the transferability rate of networks with different encoding (TREC:0.195, AG: 0.165, Amazon movie: 0.109) is lower than for the other configurations. The transferability rate for all other combinations ranges from 25.0% (Amazon movie, retrained without changes) to 75.9% (TREC, different hyperparameter). However, an interesting pattern can be seen: The transferability rate of the retrained networks (word2vec-first-half-retrained) is for all three datasets lower compared to the ones with different hyperparameter or dataset (TREC: 0.445 compared to 0.490 and 0.759, AG: 0.369 compared to 0.583 and 0.562, Amazon movie: 0.250 compared to 0.412 and 0.422). Since this is an interesting pattern, further experiments were conducted on that matter (see App. A and B).

3.3 Discussion

The experiments show clearly that adversarial examples are existing for text classification and that the results of Samanta and Mehta (2017) can both be verified and extended to different neural network architectures, datasets and to a lesser extend different encoding. Furthermore, the generated adversarial examples show a high quality, with many of them requiring only one change, and almost all only requiring a couple of changes. This shows that the algorithm of Samanta and Mehta (2017) is clearly working. Furthermore it can be shown that transferability, previously known for image classification, also exists for text classification, although with some limitations.

Since there is no single transferability rate for image classification but rather it varies greatly across different datasets and models (Szegedy et al., 2013; Goodfellow et al., 2014; Hosseini et al., 2017; Papernot et al., 2016c,d), it is hard to directly compare the transferability rate of the experiment to the one found for image classification. However, at least it can be noted that the transferability rate found here for text classification (maybe with the exception of different encoding as mentioned above) is not outside of the ones found for image classification (e.g. see Szegedy et al. (2013); Goodfellow et al. (2014); Hosseini et al. (2017); Papernot et al. (2016c,d)).

The experiment shows an interesting pattern: For all tested datasets, the transferability rate for the retrained network (without modification of hyperparameter or dataset) is lower compared to the ones with modified hyperparameter or different datasets. One assumption is that the changed networks learn something more different than the other ones. For this, the gradient (as a measure for how close the learned information is in all networks) was compared (see App. A), and it was shown that indeed the retrained networks seem to focus on different things. Furthermore, it was tried to reproduce the results on image classification (see App. B), which showed no result since the generated examples there did not transfer at all.

This experiment was repeated with a slightly different methodology in App. C, which was able to confirm that transferability exists, but did not show the same pattern as described above.

Chapter 4

Increasing robustness of deep neural networks: Defensive Distillation

The next question is how to harden deep neural networks. One of the goals for increasing robustness is to reduce the sensitivity of these networks to adversarial examples.

There has been some work for increasing the robustness in the image domain, however to the knowledge of the author no work has been done for the text domain at the time of writing.

One possibility is to use the already used methods for increasing the robustness in the image domain and also using them in the text domain. One of these methods is *defensive distillation* (Papernot et al., 2016b; Papernot and McDaniel, 2017). This leaves the question whether defensive distillation also works for the text domain. This has to be answered for both the generation of adversarial examples and transferring already generated examples.

4.1 Experiment Setup

This experiment has the same general setup as the previous one (see Sec. 3.1). However, instead of using normally trained networks *defensive distillation* is used for training in an attempt to increase the robustness of the networks.

4.1.1 Defensive Distillation

Defensive distillation is a method proposed by Papernot et al. (2016b); Papernot and McDaniel (2017) and is based on the *distillation* method by Hinton et al. (2015) (some early thoughts can be found at Ba and Caruana (2014)). Both methods are based on the idea that knowledge from one neural network can be transferred to another neural network by using *soft labels* (the output of a previously trained network which represents probability of the different classes) for the training instead of *hard labels* (where every data belongs to exactly one class).

To achieve this effectively the soft labels have to be calculated according to the equation 4.1, where y_i is the probability of the ith class, l_i the ith *logit* (the inputs to the final softmax level) and T the *temperature*. The temperature is used to determine how "soft" the resulting labels are and the less the influence of the logits are to the result. A special case is $T = 1$, which equals to a normal *softmax*.

$$y_i = \frac{e^{l_i/T}}{\sum_i e^{l_i/T}} \tag{4.1}$$

This can now be used to transfer knowledge from the original network to a distilled one. The original network is trained as usually. After that, the soft labels are calculated for the training set using a high temperature (e.g. Papernot et al. (2016b) found out that in their experiments a temperature of $T = 20$ is optimal). These soft labels are then used to train the distilled network, which has to use the same temperature in its last layer during training. After that, the temperature is set back to $T = 1$ and the network can be used normally.

The difference between distillation and defensive distillation is that Hinton et al. (2015) use distillation to transfer knowledge from a large neural network to a small one while retaining accuracy, whereas Papernot et al. (2016b); Papernot and McDaniel (2017) use defensive distillation to transfer knowledge from one network to another one with the same size in order to make it harder to find adversarial examples.

For this thesis the variant described in Papernot et al. (2016b) is used. The network is trained on both the hard labels and the soft labels because this should significantly improve the process according to Hinton et al. (2015).

4.1.2 Tested neural networks variations used

All experiments were run on a defensively distilled network with the same architecture as the network used for generating the adversarial examples in Section 3.1 (namely **target-word2vec-first-half**).

Since the primary target is to determine whether defensive distillation actually increases the robustness, it is not necessary to find the optimal hyperparameter for the method. Because of this, the temperature $T = 20$ chosen for all experiments is the one used by Papernot et al. (2016b) for the MNIST dataset. To get a better overview over the influence of the temperature, the temperatures $T = 10$, $T = 30$ and $T = 40$ were also tested.

For the training both the soft labels and the hard labels were used, where the loss function consists of 10% of the hard label loss and 90% of the soft label loss.

The precision of the networks trained with defensive distillation can be seen in Tab. 4.1. All networks were trained for 10 epochs.

4.1.3 Testing robustness against generating adversarial examples

For testing whether defensive distillation increases the robustness of neural networks against adversarial examples, the algorithm from section 3.1 is used on the network trained with defensive distillation.

Table 4.1: Accuracy of used defensively distilled networks against test set of the specified dataset.

Dataset	Tempe-rature	Accuracy	Accuracy without distillation from Chap. 3 Tab. 3.1 target-word2vec-first-half
TREC	10	0.440	0.862
AG	10	0.733	0.759
Amazon movie	10	0.798	0.885
TREC	20	0.446	0.862
AG	20	0.728	0.759
Amazon movie	20	0.825	0.885
TREC	30	0.386	0.862
AG	30	0.739	0.759
Amazon movie	30	0.795	0.885
TREC	40	0.436	0.862
AG	40	0.744	0.759
Amazon movie	40	0.780	0.885

4.1.4 Testing robustness against transferability

To test whether defensive distillation has any effect on transferability, the adversarial examples from Chapter 3 are used. An adversarial example is tested on the distilled network if the predicted class from the distilled network of the corresponding unaltered input matches the correct class.

4.2 Results

The first thing to note is the bad performance of the networks with the TREC dataset (see Tab. 4.1). The exact reason for this is unknown, possible reasons include the small size of the dataset (the whole dataset has only 5500 questions, and the training only uses half of the dataset) or the short length of the questions in comparison to the other datasets. To find the exact reason for the highly reduced accuracy a detailed analysis of the distillation process would be needed, which is beyond the scope of this thesis. Because of this, further analysis will be based on the other two datasets.

For the other two datasets, AG and Amazon movie, the performance of the networks trained with distillation is only slightly worse compared to the networks trained without distillation (see Tab. 4.1). The decrease of accuracy is expected according to Papernot et al. (2016b), although in their experiment the decrease of accuracy was notably smaller.

However, if we look at the success rate of generating adversarial examples (see Tab. 4.2), the difference between the distilled and the non distilled networks is marginal at best. Overall, the success rate stays high with 53% to 74% for the TREC dataset and

Table 4.2: Results: Generating adversarial examples with defensive distillation used in training

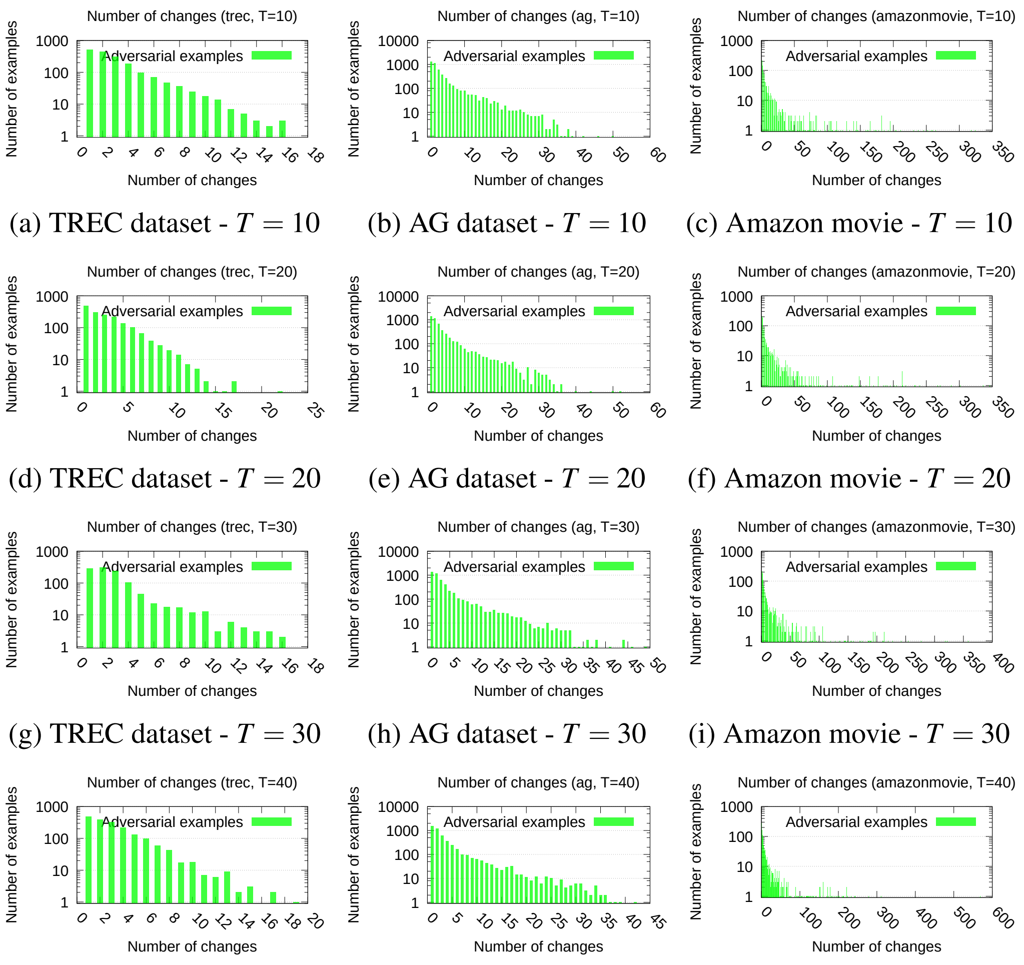
Temperature	Dataset	Number tested	Number successful	Success rate	Success rate without distillation (Sec. 3.2 Tab. 3.2)
10	TREC	2407	1782	0.740	0.764
10	AG	4886	4771	0.976	0.982
10	Amazon movie	1071	1029	0.961	0.984
20	TREC	2411	1684	0.698	0.764
20	AG	4876	4785	0.981	0.982
20	Amazon movie	1074	1037	0.966	0.984
30	TREC	2030	1089	0.536	0.764
30	AG	4922	4786	0.972	0.982
30	Amazon movie	1124	1095	0.974	0.984
40	TREC	2413	1796	0.744	0.764
40	AG	4907	4801	0.978	0.982
40	Amazon movie	1262	1219	0.966	0.984

96% to 98% percent for the AG and Amazon movie dataset, with no visible difference between the different temperatures. This is surprising since the experiments of Papernot et al. (2016b) showed an improved robustness for image processing networks even for low temperatures.

If we look at the number of changes itself (Tab. 4.3), we can see that distillation made it a bit harder to generate adversarial examples. The mean number of changes went up through all experiments (TREC: 2.33 to 2.93 – 3.35, AG: 3.52 to 3.94 – 4.47, Amazon movie: 14.76 to 17.31 – 19.29). A similar increase can be seen in some instances for the median (TREC temperature 20 and 40: 2 to 3, Amazon movie all temperatures: 3 to 4) and mode (TREC temperature 30: 1 to 2) number of changes, however these seem to be minor. The overall distribution of number of changes, as seen in Fig. 4.1, compared to the networks without distillation (Fig. 3.5) seem to be quite close, which means that any difference is minimal at best and might be a result of the smaller sample size for the networks with defensive distillation.

Table 4.3: Results: Number of changes for generating adversarial examples with defensive distillation used in training

Temperature	Dataset	Mean length of sentences in successful runs	Mean number of changes	Median number of changes	Mode number of changes
10	TREC	10.58	3.05	2	1
10	AG	32.12	4.47	2	1
10	Amazon movie	148.85	19.29	4	1
20	TREC	10.45	3.35	3	1
20	AG	32.01	4.18	2	1
20	Amazon movie	143.05	18.39	4	1
30	TREC	10.02	2.93	2	2
30	AG	32.00	4.13	2	1
30	Amazon movie	140.04	18.47	4	1
40	TREC	10.51	3.23	3	1
40	AG	31.76	3.94	2	1
40	Amazon movie	141.24	17.31	4	1



(a) TREC dataset - $T = 10$ (b) AG dataset - $T = 10$ (c) Amazon movie - $T = 10$

(d) TREC dataset - $T = 20$ (e) AG dataset - $T = 20$ (f) Amazon movie - $T = 20$

(g) TREC dataset - $T = 30$ (h) AG dataset - $T = 30$ (i) Amazon movie - $T = 30$

(j) TREC dataset - $T = 40$ (k) AG dataset - $T = 40$ (l) Amazon movie - $T = 40$

Figure 4.1: Results: Distribution of generated adversarial examples with defensive distillation used in training for different temperatures T

Table 4.4: Results: Transferability of generated adversarial examples from Chap. 3 to networks trained with defensive distillation

Temperature	Dataset	Number tested	Success rate	Success rate without distillation taken from Sec. 3.2 Tab. 3.4 word2vec-first-half-retrained
10	TREC	766	0.261	0.445
10	AG	8150	0.323	0.369
10	Amazon movie	1157	0.218	0.250
20	TREC	772	0.491	0.445
20	AG	8126	0.331	0.369
20	Amazon movie	1217	0.199	0.250
30	TREC	335	0.107	0.445
30	AG	8141	0.325	0.369
30	Amazon movie	1229	0.253	0.250
40	TREC	747	0.244	0.445
40	AG	8134	0.337	0.369
40	Amazon movie	1195	0.211	0.250

An other aspect is robustness against transferability, which was tested with the adversarial examples generated in Chap. 3. Tab. 4.4 shows that the transferability rate for the AG (0.323 – 0.337 compared to 0.369) and Amazon movie (0.199 – 0.253 compared to 0.250) datasets is a bit lower in most cases compared to the retrained network without distillation (word2vec-first-half-retrained, Tab. 3.4), with a difference of only 0.032 – 0.051 (with Amazon movie at temperature 30 being even 0.003 higher). An exception here is the TREC dataset, however since it has a low accuracy to begin with (See Tab. 4.1) these values are not really meaningful.

4.3 Discussion

Based on the results the conclusion has to be drawn that defensive distillation has not the same effect for text classification as it has for image classification. The robustness of networks trained with defensive distillation is increased only marginally at best, which is especially true for the generation of adversarial examples.

One reason might be that defensive distillation effectively does a *gradient masking* (Tramèr et al., 2017a; Papernot et al., 2016e), which works against methods which directly or indirectly add the gradient to a generated image. However, in the algorithm used in this thesis the value of gradient is only used as a measure on the reaction of a network on a given word, not directly added onto the input. Since the exact characteristics of the gradient are not important, the gradient masking itself has only a minimal effect on our algorithm.

This hypothesis is further boosted by the results of Carlini and Wagner (2016), who were able to generate adversarial examples for networks trained with defensive distillation with a slight modification to the generating algorithm. Carlini and Wagner (2016) were able to use the gradient by restoring it from the vanishing caused by defensive distillation. This shows that the gradient still holds enough information for the generation of adversarial examples, it is just not as easily accessible as without defensive distillation.

If this hypothesis were true, this might mean that other methods based on gradient masking are also not effective in the text domain. This would, however, need further testing and might be a topic for future work.

Chapter 5

Increasing robustness of deep neural networks: Adversarial Data Augmentation

While Chap. 4 focused on increasing the robustness of deep neural networks for text classification against adversarial examples, this chapter attempts to use adversarial examples as a special form of data augmentation to improve generalisation and to increase the overall robustness (e.g. the robustness on normal data) of deep neural networks.

Data augmentation describes the process of enlarging the dataset using transformations which preserve the label of the sample in order to reduce overfitting (Krizhevsky et al., 2012), which can even be considered best practise for image processing (Simard et al., 2003).

Prior research on data augmentation for text focused on randomly replacing words with synonyms (Zhang et al., 2015; Zhang and LeCun, 2015). Since adversarial examples are crafted to explicitly be misclassified by neural networks, the idea is to use those generated adversarial examples for data augmentation.

Therefore, this chapter will compare replacing random words with *adversarial data augmentation* to find out whether adversarial examples are useful for data augmentation in text classification.

5.1 Experiment Setup

The setup is similar to Sec. 3.1. The network architecture tested is the same as the target network. Two different types of data augmentation were tested: *Adversarial Data augmentation* as the proposed method and *Random Data Augmentation* as the baseline method.

To test the improvement in robustness and the influence of both data augmentation methods, the networks are trained on the augmented training datasets (with different amounts of augmented data samples) and the accuracy is evaluated on the test dataset.

5.1.1 Random Data Augmentation

The *Random Data Augmentation* method, as described by Zhang et al. (2015); Zhang and LeCun (2015), is based on random word replacement by synonyms. The number of replacement is determined by a geometric distribution, where the probability of n replacements is described in Eq. 5.1. In this thesis (as well as in Zhang et al. (2015); Zhang and LeCun (2015)) p is set to $p = 0.5$.

$$P(n) = (1-p)^{n-1}p \tag{5.1}$$

In the original papers by Zhang et al. (2015); Zhang and LeCun (2015), the LibreOffice[1] *mytheas* is used, however in this thesis the WordNet (Fellbaum, 1998) was directly used for synonym selection. This has the consequence that each synonym has the same probability of being chosen.

The dataset with data augmentation is created by taking the original dataset and adding the desired amount of augmented sentences to it, effectively increasing the number of samples in the dataset.

5.1.2 Adversarial Data Augmentation

A new method proposed by this thesis is to use the generated adversarial examples from Chap. 3 as data augmentation. Therefore, the desired amount of adversarial examples is added to the original dataset with the original class.

Since the generation of adversarial examples takes a considerable amount of time, the possible number of adversarial examples which can be added is limited (In this thesis to the amount generated in 3).

For each configuration three networks were trained on the datasets. For each training a new dataset was created. This should counter random effects (Random Data Augmentation) and effects of the selection of the subset of adversarial examples (Adversarial Data Augmentation).

5.2 Results

The mean accuracies of the three trained networks for each augmented dataset can be seen in Tab. 5.1. In general all of the accuracies are pretty close, the difference between the maximum and the minimum is smaller than 0.04 for all datasets and augmentation methods. A similar picture is visible if you plot the data (see Fig. 5.1), where no clear trend is visible. Overall, the differences look more like random effects than improvements due to the addition of augmented data.

[1]`https://www.libreoffice.org/`

5.3 Discussion

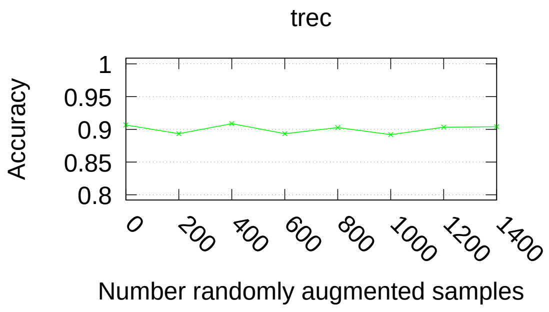
Overall, no real effect of neither Random Data Augmentation nor Adversarial Data Augmentation is visible. This falls somewhat in line with Zhang and LeCun (2015), who noted that in their experiments networks trained on large-scale dataset have a small generalisation error to start with. While it is hard to give criteria what exactly is considered as large-scale, for the experiment conducted here this seems to include all three datasets. Future work could include testing both data augmentation methods on smaller datasets.

It is important to note that the number of data augmentation is quite small compared to the original data sets, the largest being the AG dataset with 8000 augmented data samples on 16000 original samples. It would be interesting to see how larger numbers of augmented data influence the accuracy, however since adversarial examples are expensive to generate (in terms of computation time) it was not feasible in this thesis to generate more.

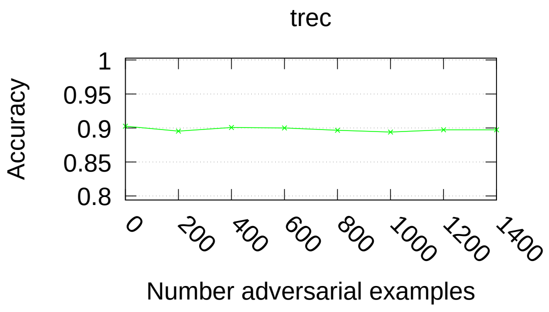
Table 5.1: Mean precision of neural networks trained on augmented datasets on test set

Dataset	Number augmented samples	Accuracy of Random Data Augmentation	Accuracy of Adversarial Data Augmentation
TREC	0 (*no data augmentation*)	0.907	0.903
TREC	200	0.893	0.895
TREC	400	0.909	0.901
TREC	600	0.893	0.900
TREC	800	0.903	0.897
TREC	1000	0.892	0.894
TREC	1200	0.903	0.897
TREC	1400	0.904	0.897
AG	0 (*no data augmentation*)	0.720	0.724
AG	200	0.718	0.725
AG	400	0.719	0.716
AG	600	0.724	0.718
AG	800	0.729	0.726
AG	1000	0.721	0.723
AG	1200	0.724	0.726
AG	1400	0.711	0.718
AG	1600	0.719	0.718
AG	1800	0.717	0.726
AG	2000	0.720	0.725
AG	2200	0.706	0.716

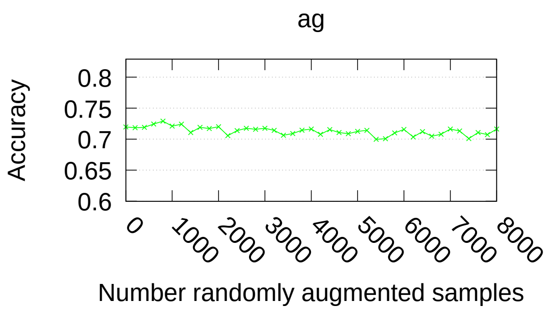
AG	2400	0.714	0.727
AG	2600	0.718	0.715
AG	2800	0.716	0.719
AG	3000	0.718	0.711
AG	3200	0.714	0.725
AG	3400	0.706	0.716
AG	3600	0.709	0.721
AG	3800	0.714	0.721
AG	4000	0.716	0.718
AG	4200	0.708	0.714
AG	4400	0.715	0.723
AG	4600	0.711	0.719
AG	4800	0.709	0.719
AG	5000	0.713	0.717
AG	5200	0.714	0.724
AG	5400	0.700	0.720
AG	5600	0.701	0.717
AG	5800	0.710	0.720
AG	6000	0.716	0.718
AG	6200	0.704	0.719
AG	6400	0.712	0.723
AG	6600	0.705	0.725
AG	6800	0.708	0.715
AG	7000	0.716	0.719
AG	7200	0.713	0.715
AG	7400	0.701	0.731
AG	7600	0.711	0.719
AG	7800	0.707	0.724
AG	8000	0.716	0.718
Amazon movie	0 (*no data augmentation*)	0.895	0.870
Amazon movie	200	0.903	0.909
Amazon movie	400	0.897	0.893
Amazon movie	600	0.908	0.898
Amazon movie	800	0.890	0.875
Amazon movie	1000	0.908	0.878
Amazon movie	1200	0.900	0.888



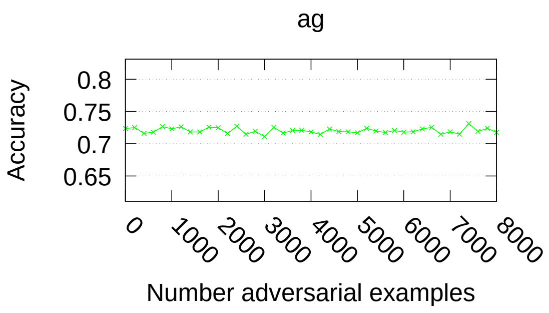
(a) TREC (Random Data Augmentation)



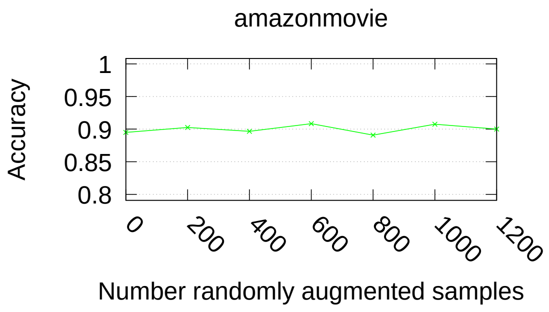
(b) TREC (Adversarial Data Augmentation)



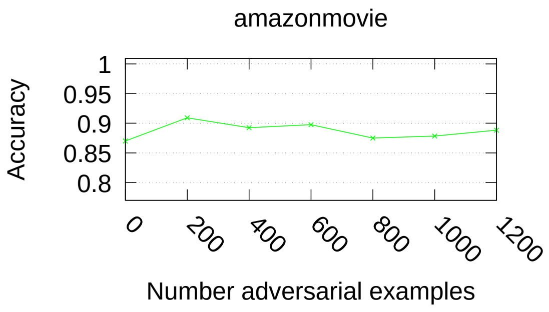
(c) AG (Random Data Augmentation)



(d) AG (Adversarial Data Augmentation)



(e) Amazon movie (Random Data Augmentation)



(f) Amazon movie (Adversarial Data Augmentation)

Figure 5.1: Mean precision of networks trained with Random Data Augmentation (left) and Adversarial Data Augmentation (right). Mean over three trained networks

Chapter 6

Conclusion

Unfortunately adversarial examples still remain a mystery despite their easy generation, as proven by both this thesis (see Chap. 3) as well as existing literature (text domain see Liang et al. (2017); Samanta and Mehta (2017); Jia and Liang (2017), image domain see Goodfellow et al. (2014); Kurakin et al. (2016); Papernot et al. (2016a)). Furthermore, this thesis shows that transferability (an adversarial example created for one neural network will also work on other networks) is also preserved in the text domain (even when the networks are trained on different datasets, have different hyperparameter or the text is encoded in a different scheme). This opens two main questions:

How can we increase the robustness of neural networks against adversarial examples? One of the methods to reduce the sensibility of neural networks for image classification to adversarial examples is defensive distillation as proposed by Papernot et al. (2016b); Papernot and McDaniel (2017). In this thesis (see Chap. 4) defensive distillation was tested for text classification with the result, that this method does not help increasing the robustness. Furthermore, the speculation is that methods (like defensive distillation) based on *gradient masking* (Tramèr et al., 2017a; Papernot et al., 2016e) only has a marginal effect on the robustness of neural networks (since the gradient is not added directly and thus *gradient masking*(Tramèr et al., 2017a; Papernot et al., 2016e) has no effect). This leaves the question open for future work on how to increase the robustness of neural network for text classification against adversarial examples.

Can we use adversarial examples for increasing the overall robustness of neural networks? Chap. 5 tried to use adversarial examples as a new method for data augmentation to increase the overall robustness of neural networks for text classification. Unfortunately, the experiment showed a minimal increase in overall accuracy at best, and often no real change to accuracy at all. From this it can be concluded that using adversarial examples for data augmentation is not an effective method. Future work could look into other areas where adversarial examples could be useful.

Appendix A

Analysing similarity of gradients for networks in Chapter 3

One of the interesting results found in Chap. 3 was the transferability rate of the different neural network variants. In Tab. 3.4 one can see that the transferability rate of the retrained network (with the same hyperparameter and dataset) is lower than the ones where the dataset or the hyperparameter were changed. This is surprising, since it is expected that the retrained network learns similar to the original network and therefore is more susceptible to transferred adversarial examples.

One way to find out whether the networks learn similar is to compare their gradients. If the gradients are close, it is likely that they respond identical to the same input and are thus more susceptible to transferred adversarial examples.

A.1 Experiment Setup

This section is based on Chap. 3 and has therefore the same general setup as described in Sec. 3.1. The networks used here are exactly the same (with equal weights) as in Chap. 3. For each adversarial example generated in Chap. 3, the gradient of the original unaltered sentence is calculated for the following network variants:

- target-word2vec-first-half
- word2vec-first-half-retrained
- word2vec-second-half
- word2vec-alternative-first-half

After that, the gradients are pairwise compared between target-word2vec-first-half and the other networks. For this, the following distance functions are used:

- *Euclidean distance:* h_i^j represents the gradient for the ith word at the jth position in the word2vec encoding for the target-word2vec-first-half network, g_i^j for the other network. The euclidean is then calculated as following:

$$d_{euclidean}(h,g) = \sqrt{\sum_{i}^{number\ words} \sum_{j}^{300} (h_i^j - g_i^j)^2}$$

- *Maximum word distance:* The distance is 0 if the highest gradient is in the same word for both networks, else the distance is 1. Let us assume that the mean maximum word distance is 0.6, this means that in 60% of all sentences both networks have different words as the one they focus the most (under the assumption that the words with the highest gradients are the most important for the network).

A.2 Results

The results in Tab. A.1 show a clear pattern: The distance metric (both euclidean and maximum word distance) are highest for the retrained networks (TREC: 0.970 and 0.598, AG: 0.623 and 0.555, Amazon movie: 0.925 and 0.754), followed by the network with the different dataset (TREC: 0.530 and 0.193, AG: 0.482 and 0.480, Amazon movie: 0.445 and 0.441) and finally the network with the different hyperparameter (TREC: 0.401 and 0.149, AG: 0.349 and 0.381, Amazon movie: 0.266 and 0.298). The gap between the retrained network and the others is larger for the TREC (euclidean: 0.440 difference, maximum word: 0.405 difference) and Amazon movie (euclidean: 0.480 difference, maximum word: 0.313 difference) dataset, but smaller for the AG (euclidean: 0.141 difference, maximum word: 0.075 difference) dataset.

Table A.1: Comparison of target-word2vec-first-half gradients to other networks

Dataset	Network	Average Euclidean distance	Average Maximum word distance
TREC	word2vec-first-half-retrained	0.970	0.598
TREC	word2vec-second-half	0.530	0.193
TREC	word2vec-alternative-first-half	0.401	0.149
AG	word2vec-first-half-retrained	0.623	0.555
AG	word2vec-second-half	0.482	0.480
AG	word2vec-alternative-first-half	0.349	0.381
Amazon movie	word2vec-first-half-retrained	0.925	0.754
Amazon movie	word2vec-second-half	0.445	0.441
Amazon movie	word2vec-alternative-first-half	0.266	0.298

A.3 Discussion

From the results it can be concluded that indeed for all tested datasets the retrained network (word2vec-first-half-retrained) learns a different focus than the target network (target-word2vec-first-half). In addition, this different focus is the furthest away from the target network compared to the other networks tested (word2vec-second-half, word2vec-alternative-first-half). This, partly, explains why the retrained network is the least susceptible one to transferred adversarial examples (see Tab. 3.4). However, there are two things which have to be considered:

1. Although there is a clear ranking between the retrained network and the other ones, the distances do not directly correlate to the transferability rate. If they would, we would have seen a larger transferability rate for the network with different hyperparameter compared to the one with the different dataset. Since this does not happen, the gradient distances can not be the sole reason for the low transferability rate on the retrained network.

2. The exact reason for the distance is unknown and might involve other factors.

Appendix B

Repeating the transferability experiment of Chapter 3 for image classification (MNIST)

One of the interesting results found in Chap. 3 was the transferability rate of the different neural network variants. In Tab. 3.4 one can see that the transferability rate of the retrained network (with the same hyperparameter and dataset) is lower than the ones where the dataset or the hyperparameter were changed. This is surprising, since it is expected that the retrained networks learns similar to the original network and therefore is more susceptible to transferred adversarial examples.

One resulting question is whether this only holds true for text classification or if this behaviour can be found for image classification. To test this, the neural network architecture used for text classification (with slight adaptions for image classification) was used on the MNIST dataset (LeCun et al., 1998).

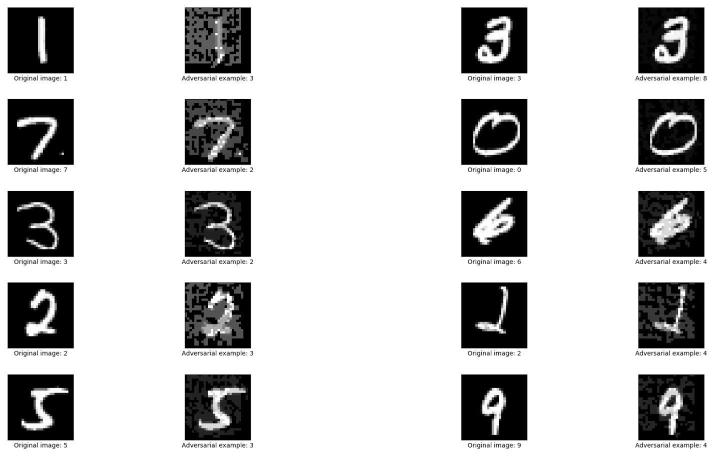


Figure B.1: Examples of generated adversarial examples.

Table B.1: Transferability rate of 500 examples on the MNIST dataset

Network	Transferability rate
first-half-retrained	0.000
second-half	0.000
alternative-first-half	0.000

B.1 Experiment setup

The setup is the same as described in Sec. 3.1, with the following adaptions for image classification:

Dataset: The MNIST dataset (LeCun et al., 1998) is used for image classification. The dataset consists of 60,000 images (size: 28×28) containing handwritten digits. The task is to identify the digit on the image.

Neural network architecture: A few modifications have to be done to the networks for the image classification process. Instead of a 1-dimensional convolutional layer, a 2-dimensional convolutional layer is used. Because of that, the global max-pooling layer is changed to a 2-dimensional max-pooling layer.

Algorithm used to create adversarial examples: All images data was normalized from $[0,255]$ to $[0.1]$. The *Basic iterative method* Kurakin et al. (2016) was used for the generation of adversarial examples. Some of the generated images can be seen in Fig. B.1

B.2 Results

It was possible to create the adversarial examples, however the transferability rate to all three networks (retrained, changed hyperparameter, changed dataset) was 0.0, as shown in Tab. B.1). This means that all adversarial examples were correctly classified by the other networks and no adversarial example was transferred successfully.

B.3 Discussion

The result is really surprising. In literature, it is shown that adversarial examples created by using the MNIST dataset transfer to other networks (Szegedy et al., 2013; Goodfellow et al., 2014; Hosseini et al., 2017; Papernot et al., 2016c,d), so a similar behaviour was expected here. However, as seen in the results, our generated adversarial examples do not transfer. Unfortunately, the reason for this is unknown, and a detailed analysis would be beyond the scope of the master thesis.

Appendix C

Repetition of the transferability test in Chap. 3

The transferability in Chap. 3 was only tested on one single network for each dataset and variation. This rises the question whether the results are still valid on different random initialisation of the networks. Since the complete run of the experiment (including generation of new adversarial examples) is not feasible due to high computation time, a shorter version of the transferability experiment was done.

C.1 Experiment setup

The setup is similar to Sec. 3.1, especially as the same neural network variations (Sec. 3.1.4) are used.

For testing transferability, the already generated adversarial examples from Chap. 3 are used. In each run, all networks are trained from scratch. After that, the adversarial examples are tested on these newly trained networks if the corresponding original sentence is classified correctly on all tested variants.

C.2 Results

The results of this experiment can be found in Tab. C.1. It can be seen that all tested networks show some transferability (all transferability rates are larger than zero), however there is some considerable variation within each dataset/variation combination.

C.3 Discussion

This experiment was able to confirm that transferability exists (as shown in Chap. 3). However, the clear pattern (the retrained networks show a lower transferability rate than those with the different dataset, which in turn show a lower transferability rate than the ones with different hyperparameters) could not be seen here. It is worth to note that this might be due to the different methodology used by both experiments (original: Filter out samples before generating adversarial examples, this experiment: Use the already filtered adversarial examples and filter them again). This might have the effect, that samples which would have been classified correctly in this experiment were filtered out by the original experiment, thus influencing the transferability rate.

To find out whether the differences are due to random initialisation of the neural networks or methodology, the original experiment in Chap. 3 has to be repeated multiple times. This, unfortunately, was not possible due to high computation time.

Table C.1: Transferability results of the repetition experiment

Repe-tition	Dataset	Target network	Number tested	Transferability rate
1	TREC	word2vec-first-half-retrained	1018	0.355
2	TREC	word2vec-first-half-retrained	1080	0.309
3	TREC	word2vec-first-half-retrained	1057	0.361
1	TREC	word2vec-second-half	1018	0.292
2	TREC	word2vec-second-half	1080	0.347
3	TREC	word2vec-second-half	1057	0.294
1	TREC	word2vec-alternative-first-half	1018	0.429
2	TREC	word2vec-alternative-first-half	1080	0.383
3	TREC	word2vec-alternative-first-half	1057	0.320
1	TREC	character-first-half	1018	0.425
2	TREC	character-first-half	1080	0.335
3	TREC	character-first-half	1057	0.218
1	AG	word2vec-first-half-retrained	6804	0.350
2	AG	word2vec-first-half-retrained	6755	0.355
3	AG	word2vec-first-half-retrained	7020	0.353
1	AG	word2vec-second-half	6804	0.369
2	AG	word2vec-second-half	6755	0.368
3	AG	word2vec-second-half	7020	0.357
1	AG	word2vec-alternative-first-half	6804	0.333
2	AG	word2vec-alternative-first-half	6755	0.330
3	AG	word2vec-alternative-first-half	7020	0.304
1	AG	character-first-half	6804	0.173
2	AG	character-first-half	6755	0.147
3	AG	character-first-half	7020	0.127
1	Amazon movie	word2vec-first-half-retrained	958	0.228
2	Amazon movie	word2vec-first-half-retrained	1070	0.257
3	Amazon movie	word2vec-first-half-retrained	1019	0.317
1	Amazon movie	word2vec-second-half	958	0.238
2	Amazon movie	word2vec-second-half	1070	0.263
3	Amazon movie	word2vec-second-half	1019	0.288
3	Amazon movie	word2vec-alternative-first-half	1019	0.232
1	Amazon movie	word2vec-alternative-first-half	958	0.215
2	Amazon movie	word2vec-alternative-first-half	1070	0.204
1	Amazon movie	character-first-half	958	0.167
2	Amazon movie	character-first-half	1070	0.131
3	Amazon movie	character-first-half	1019	0.103

Appendix D

Software

The following software libraries were used in this master thesis:

- *Creating neural networks*: Keras (Chollet et al., 2015)
- *Creating neural networks*: Tensorflow (Abadi et al., 2015)
- *Word2vec encoding*: Gensim (Řehůřek and Sojka, 2010)
- *P.O.S. Tagging* : NLTK (Bird et al., 2009)
- *Synonym finding:* NLTK (Bird et al., 2009) with the WordNet (Fellbaum, 1998; Miller, 1995) interface
- *Various array and dataset operations:* NumPy (Walt et al., 2011)

The source code is available at `https://github.com/Top-Ranger/master-code` .

References

Ag's corpus of news articles. URL `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`. Accessed online on the 27th of October 2017.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining text data*, chapter 6, pages 163–222. Springer, 1 edition, 2012. ISBN 978-1-4614-3222-7. doi: 10.1007/978-1-4614-3223-4.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2654–2662. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf`.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, July 2016. URL `http://arxiv.org/abs/1607.04311`.

François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *CoRR*, abs/1707.05373, July 2017. URL `http://arxiv.org/abs/1707.05373`.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, August 2017. URL `http://arxiv.org/abs/1707.08945`.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *CoRR*, abs/1704.04960, April 2017. URL `https://arxiv.org/abs/1704.04960`.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, December 2014. URL `http://arxiv.org/abs/1412.6572`.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, December 2014. URL `http://arxiv.org/abs/1412.5068`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, March 2015. URL `http://arxiv.org/abs/1503.02531`. NIPS 2014 Deep Learning Workshop.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, July 2012. URL `http://arxiv.org/abs/1207.0580`.

Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *CoRR*, abs/1703.04318, March 2017. URL `https://arxiv.org/abs/1703.04318`.

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, November 2015. URL `http://arxiv.org/abs/1511.03034`.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328, July 2017. URL `http://arxiv.org/abs/1707.07328`.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, August 2014. URL `http://arxiv.org/abs/1408.5882`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS 2012)*, pages 1097–1105, 2012.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, July 2016. URL `http://arxiv.org/abs/1607.02533`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING ’02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072378. URL `http://dx.doi.org/10.3115/1072228.1072378`.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *CoRR*, abs/1704.08006, April 2017. URL `http://arxiv.org/abs/1704.08006`.

Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web (WWW ’13)*, pages 897–908. ACM, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, January 2013a. URL `http://arxiv.org/abs/1301.3781`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119. Curran Associates, Inc., 2013b.

George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`.

Roger Mitton. Corpora of misspellings for download. URL `http://www.dcs.bbk.ac.uk/~ROGER/corpora.html`. Accessed online on the 10th of November 2017.

Nicolas Papernot and Patrick D. McDaniel. Extending defensive distillation. *CoRR*, abs/1705.05264, May 2017. URL `http://arxiv.org/abs/1705.05264`.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 372–387, March 2016a. doi: 10.1109/EuroSP.2016.36.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, May 2016b. doi: 10.1109/SP.2016.41.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, May 2016c. URL `http://arxiv.org/abs/1605.07277`.

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, February 2016d. URL `http://arxiv.org/abs/1602.02697`.

Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814, November 2016e. URL `http://arxiv.org/abs/1611.03814`.

Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Andras Rozsa, Manuel Günther, and Terrance E. Boult. Towards robust deep neural networks with BANG. *CoRR*, abs/1612.00138, December 2016. URL `http://arxiv.org/abs/1612.00138`.

Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *CoRR*, abs/1707.02812, July 2017. URL `http://arxiv.org/abs/1707.02812`.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003. URL `http://www.sciencedirect.com/science/article/pii/S0893608014002135`.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, March 2002. ISSN 0360-0300. doi: 10.1145/505282.505283.

Patrice Y. Simard, David Steinkraus, John C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, volume 3, pages 958–962, 2003.

Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *CoRR*, abs/1802.06430, February 2018. URL `http://arxiv.org/abs/1802.06430`.

Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *CoRR*, abs/1710.08864, October 2017. URL `http://arxiv.org/abs/1710.08864`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, December 2013. URL `http://arxiv.org/abs/1312.6199`.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, May 2017a. URL `http://arxiv.org/abs/1705.07204`.

Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *CoRR*, abs/1704.03453, April 2017b. URL `http://arxiv.org/abs/1704.03453`.

Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2): 22–30, 2011. URL `http://dx.doi.org/10.1109/MCSE.2011.37`.

Xiang Zhang and Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, February 2015. URL `http://arxiv.org/abs/1502.01710`.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 649–657. Curran Associates, Inc., 2015.

Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, October 2015. URL `http://arxiv.org/abs/1510.03820`.

Erklärung der Urheberschaft

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum　　　　　　　　　　　　　　　　　　　　　　　　Unterschrift

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum	Unterschrift