TRANSFORMERS ON A DIET

Semi-Supervised Generative Adversarial Networks for Reducing Transformers' Hunger for Data in Aspect-Based Sentiment Analysis

MASTER THESIS
Dirk Koelewijn
15-12-2022

SUPERVISORS
Dr. N. Strisciuglio
Dr. S. Wang
A. Vandor MSc

UNIVERSITY OF TWENTE.

accenture

Abstract

The vast amount of reviews and opinions being shared online for practically all available goods and services has an enormous potential value. Although the current state-of-the-art Aspect-Based Sentiment Analysis (ABSA) methods show impressive results in extracting valuable opinions, these Transformer-based models require large high-quality annotated training datasets. Datasets that are not always available and which are very costly to create. To reduce the hunger of these models for annotated data, we for the first time apply Generative Adversarial Networks (GANs) to ABSA. We investigate using both regular and Wasserstein semi-supervised GANs to generate artificial word embeddings, with varying amounts of unlabelled data and varying generator complexity. We show that adding such a GAN can significantly improve performance, even without using unlabelled data. Furthermore, we identify how much unlabelled data works best and show that generators with more hidden layers perform better. Altogether, we show that our method allows for reducing annotated data by 50% while still achieving similar performance.

Acknowledgements

This endeavour would not have been possible without my supervisors, who generously provided their feedback, knowledge and expertise. I would like to express great appreciation for my university supervisors, dr. Nicola Strisciuglio and dr. Shenghui Wang, who helped me a great deal with their constructive feedback that helped me to make this thesis of more scientific value. Additionally, words cannot express my gratitude to Adam Vandor, my supervisor from Accenture who I could always reach for great ideas, advice, motivational support and of course a huge passion for GANs. Thanks should also go to Accenture Netherlands, who gave me a carte blanche to devise and execute this research.

Considering the road to finishing this thesis was not always fun or easy, this acknowledgement would not be complete without thanking my friends and family. I especially want to thank my girlfriend, fraternity and amazing people at the organisation of the Kick-In. Their belief in me and the awesome memories we have shared have greatly motivated me during this journey.

Contents

Chapter 1

Introduction

The introduction of Transformers significantly improved the then firmly established state-of-the-art in machine translation, mainly by its ability to better model long-term relationships and to greatly reduce the training cost [68]. In particular pre-trained Transformer-based word embedding models have been revolutionary in natural language processing, with models like BERT, RoBERTa and various GPT models outperforming the previous state-of-the-art on many benchmark datasets in various fields of natural language processing [10, 19, 42, 57, 58]. These models have been pre-trained on enormous quantities of data by organisations like Google, Facebook and OpenAI. As a result, these models can be fine-tuned to specific domains or tasks in a relatively short time and with relatively little training data, often outperforming the current state-of-the-art [19]. As a result, it needs little explanation that this technology is highly promising for researchers, companies and society in general.

Although fine-tuning to a specific task or domain may require a lot less data than pre-training a Transformer-based model, there is still a significant amount of high-quality annotated data required to achieve a decent performance [14]. Creating or collecting a large high-quality dataset is a very expensive and time-consuming process [14], while the availability of existing datasets is limited. Consequently, the huge potential of these models can at this moment not easily be realized for certain tasks and domains.

One of the tasks for which Transformer-based word embedding models have proved their potential is *sentiment analysis* [8]. Nowadays, people share their opinions online on a massive scale, with platforms like Amazon, Google, Facebook and Trustpilot containing billions of reviews on practically all kinds of goods and services imaginable. Considering that most platforms are open, the data there is just waiting to be converted into valuable information. However, manually processing such a large amount of data is expensive and time-consuming, if not practically impossible. Subsequently, sentiment analysis using artificial intelligence has become more and more important [51]. Based on text documents, sentiment analysis extracts and analyses the author's sentiment, either in general or towards something specific [40].

Generic sentiment analysis has the goal to extract a sentiment polarity for a certain piece of text, usually *positive*, *negative* or *neutral*. Additionally, sentiments can also be expressed in more classes or even as a continuous score. Table 1.1 contains a few examples of brief restaurant reviews and their respective sentiments. Generic sentiment analysis has a major drawback: it is not known towards which aspect the sentiment is expressed, while this aspect may provide essential information for many potential applications [38]. A lot of information can be missed as a result, in particular when a text contains two or more aspects with different sentiments such as in text 4 of table 1.1.

#	Text	Sentiment
1	The food in this restaurant is absolutely amazing.	Positive
2	It's located near the highway.	Neutral
3	I would not recommend this place.	Negative
4	The food was amazing, but it has unfriendly staff.	Conflict

Table 1.1: Sentiments for brief example review of a restaurant

The relatively new field of Aspect-Based Sentiment Analysis (ABSA) aims to identify which sentiment is expressed towards a certain aspect, where a text may contain multiple aspects with different sentiments. Figure 1.1 gives an example of ABSA. As many useful applications for ABSA exist, the field has gained significant popularity [11]. However, sentiment analysis and in particular ABSA are usually difficult tasks considering the large variation in writing styles and sentence structuring.

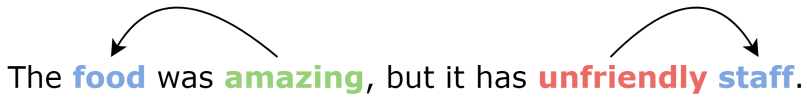

Figure 1.1: Simple ABSA example that should identify the aspect categories *food* and *staff* and then classify a positive and negative sentiment polarity, respectively.

Transformer-based word embedding models have proved to be able to achieve and improve state-of-the-art performance in ABSA [8]. However, the common ABSA training datasets used to achieve this performance still contain thousands of sentences each, with each sentence manually annotated by one or multiple experts [20, 33, 56]. As a result, persons or organisations wanting to apply ABSA to other sub-domains than those of the common datasets still need to create a large high-quality dataset themselves.

To reduce the hunger for annotated data of Transformer-based word embedding models, Generative Adversarial Networks (GANs) [25] could provide a feasible solution. In a GAN, a generator is trained to produce artificial samples that resemble the real data, while a discriminator attempts to tell the real from artificial samples. By playing an adversarial 'game' with the discriminator, the generator will learn to produce more and more realistic samples. GANs have been particularly successful in the computer vision domain, for example in applications such as image generation and image manipulation [32].

Over recent years, GANs have also been increasingly popular in the natural language domain, as it has been used in state-of-the-art approaches for text generation [1]. Research has proved that adding GANs to text classification tasks can indeed increase performance if less annotated data is available [14], it can increase model robustness [4] and that semi-supervised approaches are viable solutions [64]. All things considered, GANs are promising to reduce the need for annotated data in natural language applications such as ABSA.

In this master thesis, conducted as a part of the Data Science & Technology specialization of the master Computer Science at the University of Twente and executed at Accenture Netherlands, we investigate whether we can use a GAN in combination with Transformer-based ABSA to reduce the need for annotated data in training. To the best of our knowledge, no Transformer-based ABSA method exists that has proved to achieve good performance with a limited amount of annotated data. Additionally, semi-supervised approaches or GANs have never been applied

to ABSA with very little work on adversarial training for ABSA existing in general. Considering this, we define the following research questions:

RQ1 Can applying a semi-supervised GAN to generate artificial word embeddings improve the performance of Transformer-based ABSA, in particular when smaller amounts of annotated data are available?

RQ2 How does the amount and ratio of the labelled and unlabelled data used in a semi-supervised GAN influence the performance of Transformer-based ABSA?

To answer these questions, we first dive into the theoretical basis of Transformers, Transformer-based word embedding models and GANs in chapter 2. Next, we discuss existing related work on both ABSA and GANs in natural language processing in chapter 3, followed by how we combine a Transformer-based ABSA method with GANs and how we evaluate our experiments in chapter 4. We present the results in chapter 5 and discuss the implications and limitations of our research in chapter 6. Finally, we draw our conclusions on our research questions in chapter 7.

At the end of this thesis, the Appendix contains more detailed results and additional plots that visualise these results. All code to build, train, and test the model is publicly accessible at: `https://github.com/DirkKoelewijn/Transformers-on-a-diet`

Chapter 2

Theoretical background

This chapter introduces the core concepts that will be used in this research, as well as additional concepts that are useful for understanding the related work. For convenience, we split this theoretical background into four main areas of interest. Figure 2.1 gives a graphical overview of the concepts included in this chapter.

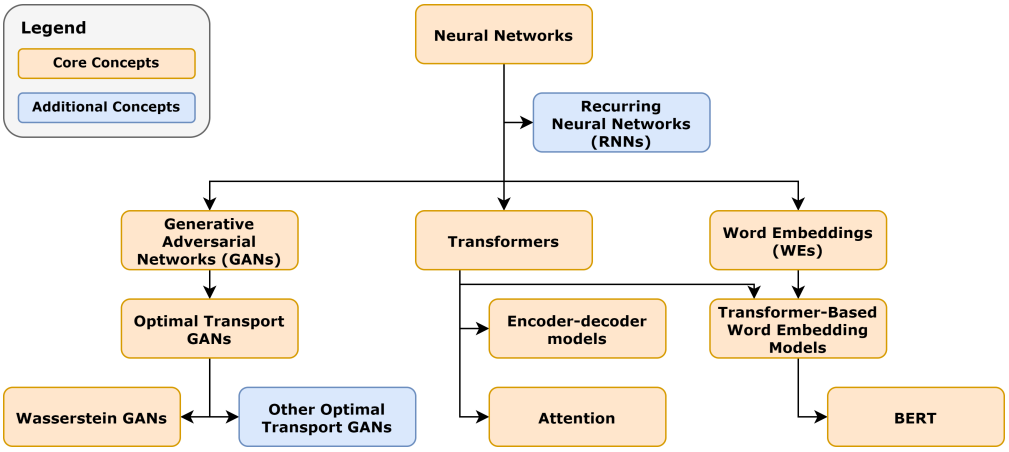

Figure 2.1: Overview of core and additional concepts included in this theoretical background

Section 2.1 elaborates on neural networks as a shared basis for most other concepts in this chapter, together with frequently used variants in the research area. In section 2.2 we discuss the generative adversarial networks with a special focus on those using optimal transport. Transformers and their core elements are then introduced in section 2.3. Last, we discuss word embedding models in section 2.4 with a focus on the contextual Transformer-based models used in this research.

2.1 Neural Networks

Since the introduction of the first artificial neuron by McCulloch and Pitts in 1943 [45] and the multilayer perceptron by Rosenblatt in 1958 [60], artificial neural networks have come a long

way. Although Rosenblatt already proved artificial neural networks could be used as universal function approximators, it was not until Rumelhart et al.'s [61] invention of gradient descent in 1986 that these networks could learn efficiently. Nowadays, various types of neural networks produce state-of-the-art performance in various artificial intelligence applications, such as image classification [16] and natural language processing [19, 42].

2.1.1 Feed-forward Neural Networks

The Feed-forward Neural Networks (FNNs or NNs) widely used today consist of layers of artificial neurons. These neurons represent a mathematical function that loosely mimics the functioning of neurons found in the brain [46]. Equation 2.1 shows the mathematical equation for a neuron.

$$y' = a(w^T x + b) \tag{2.1}$$

Each artificial neuron receives a set of inputs, the vector x. The neuron multiplies each input by weight using weight vector w and then adds a bias term b. The resulting value z is then put into an activation function a that determines to what extent the neuron 'fires' output y'. Figure 2.2 shows a graphical representation of this process.

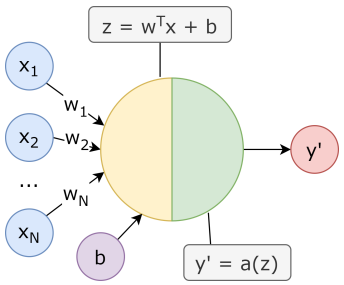

Figure 2.2: Model of an artificial neuron.

In NNs, these artificial neurons are used in layers consisting of an arbitrary number of neurons, where each layer receives the previous layer's output as input. The first layer is the input layer and takes the shape of the input, the end layer is the output layer that outputs the model's prediction. The layers in between are the hidden layers. Figure 2.3 shows an NN example schematically.

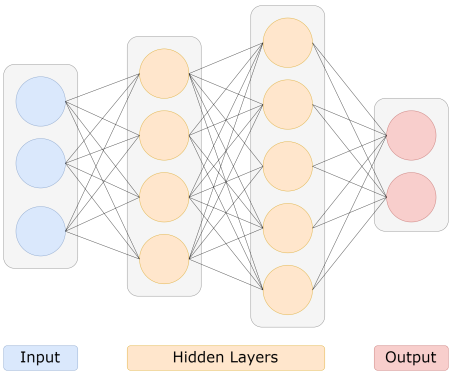

Figure 2.3: Schematic of an example feed-forward neural network.

Loss Function

The ultimate goal of NNs is to find the perfect weights w and biasses b for all its neurons, such that the model predicts the correct output value y' for all possible inputs x. As these perfect parameters in principle cannot be calculated directly, this results in an optimization problem aiming to reduce the prediction error. In order to produce a single scalar value that represents the prediction error, a loss function L is added. The lower the loss value and the closer to zero, the better the prediction and thus the better the parameters. Frequently used loss functions are the mean squared error for classification problems and the cross-entropy loss for regression problems [59]. Considering that the NN is optimized based on the loss value, the choice of the loss function can significantly affect the training of the NN.

Gradient Descent & Backpropagation

In 1986, Rumelhart et al. invented gradient descent, which is used in the efficient process of backpropagation to update the parameters of NNs [61]. Gradient descent calculates the gradient of the loss with respect to a single parameter, allowing to change the parameter in the direction in which the loss value decreases. In backpropagation, the derivative of the loss function is first calculated for the last layer after which it goes layer by layer to the first layer, using the chain rule and the derivative from the later layer to efficiently calculate the new derivative. To update the parameter, the gradient of the loss function multiplied by a learning rate is subtracted from the original parameter. Equation 2.2 shows the formula for updating the i^{th} parameter θ in layer l of the NN with learning rate α with respect to loss function L.

$$\theta_i^{(l)} \leftarrow \theta_i^{(l)} - \alpha \cdot \frac{\delta L}{\delta \theta_i^{(l)}} \tag{2.2}$$

2.1.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [61] are specialized to process a sequence of inputs, often supporting a variable length. RNNs use the idea of parameter sharing to transform from a regular feed-forward neural network to a recurrent network, sharing the parameters over time. For at least one layer of the RNN, the output of a hidden layer is fed back to themselves for the next item in the sequence. Figure 2.4 gives an example of a simple RNN.

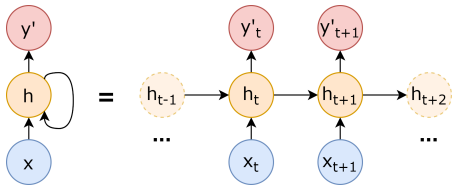

Figure 2.4: Example of a simple RNN.

Equation 2.3 shows the formula for output y' of a simple recurrent neuron at time t with input vector x, weight vector w, recurrent weight vector u, bias term b and activation function a:

$$y'^{(t)} = a(w^T x^{(t)} + u^T (w^T x^{(t-1)}) + b) \tag{2.3}$$

The challenge to model long-term dependencies is particularly problematic for RNNs [24]. This is because the weight w of equation 2.3 is effectively multiplied by itself many times for large

differences of t, leading to either exploding gradients if w is large or vanishing gradients if w is small. Bengio et al. [5, 6] proved that if we want to robustly store past knowledge with small perturbations, the RNN must have gradients that vanish. The network can learn in this way, but in practice, this will take very long. Experiments have shown that the probability of successfully capturing over 10 to 20 steps in time becomes near impossible [6].

Long Short-Term Memory

Hochreiter and Schmidhuber [29] introduced a model that, with an essential addition by Gers et al. in 2000 [22], is known as the Long Short-Term Memory (LTSM). Compared to the regular RNN cells, LTSM cells add an input gate i, forget gate f and an output gate o. A schematic of the LTSM is shown in 2.5. Without getting into too much detail, the input and forget gate make it possible to apply different weights to the input and the hidden state and thus applying different importance to each one. The output gate makes it possible to shut the output off in specific situations. These three gates are all computed based on the current input and the last output. Gated RNNs with LTSM are nowadays often used in the most effective sequence models in practice [24].

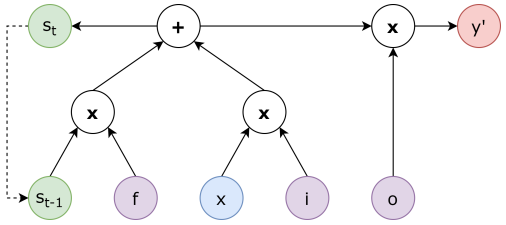

Figure 2.5: Schematic simplification of an LTSM cell. The calculation of the gates and the σ and tanh activation functions have been omitted for simplicity.

2.2 Generative Adversarial Networks

Deep Neural Networks (DNNs) promise to deliver models that can represent all kinds of probability distributions over all sorts of data [7]. Until 2014, the success of DNNs was mainly for discriminative classification models. That changed when Goodfellow et al. introduced a framework to create generative models by using an adversarial process: the Generative Adversarial Network (GAN) [25]. Since then, GANs have achieved incredible results in generating highly realistic samples in many computer vision and natural language applications [32].

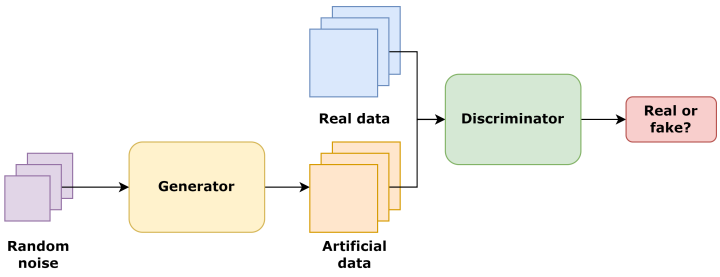

Figure 2.6: Basic overview of a GAN.

As the GAN uses an adversarial process, it contains two models that are trained roughly simultaneously. First, the model consists of a generative model G, also called the generator, that attempts to capture the distribution of the real data and generate samples from it based on random noise. Secondly, there is a discriminative model D, the discriminator, that attempts to discriminate whether a real or generated sample belongs to the distribution of the real data. Figure 2.6 shows a schematic overview of the structure of a GAN.

The training procedure of the GAN framework corresponds to a minimax game between the generator G and the discriminator D [25]. G is trained to maximize the chance of D making a mistake, whereas D attempts to identify artificially generated samples. Considering that both G and D are neural networks, the entire GAN can be trained with backpropagation. Equation 2.4 provides a mathematical formulation for this minimax game with the loss function $L(G, D)$:

$$\min_G \max_D L(G, D) = \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{2.4}$$

where x is a sample from the real distribution of the data p_d, z is a sample from the noise distribution p_z and $D(x)$ is the probability that x came from p_d rather than from p_z. For an optimal D, Goodfellow et al. have shown that this minimax game minimizes the Jensen Shannon Divergence (JSD) between the distribution of G and the true distribution p_d [25]. However, their work also shows that training G to maximize $\log(D(G(z))$ provides stronger gradients in early training.

Challenges

Research has shown that regular GANs are very hard to train in practice, with Arjovsky and Bottou arguing that the main problem for regular GANs is seeking to minimize the JSD [2]. They show that gradients vanish if G tries to optimize $\log(1 - D(G(z))$ for an optimal discriminator, while the more practical alternative $-\log(D(G(z)))$ leads to unstable training signals instead.

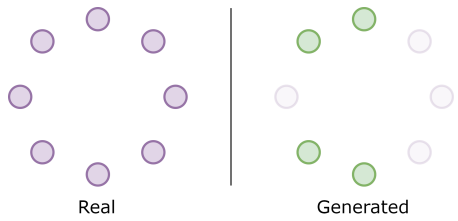

Figure 2.7: Example of mode collapse: While the real distribution consists of 8 sub-distributions, only 4 of those are generated.

GANs also suffer from a phenomenon called mode collapse. In this case, the generator only learns to reproduce only a part of the real data distribution p_d. This is a result of the training objective: G is trained to simply fool D and G only produces a certain type (mode) of samples that D will consider to be genuine [63]. Figure 2.7 shows a visualization of mode collapse. Regular GANs have additional shortcomings, such as the Nash equilibrium, internal covariate shift and the lack of proper evaluation metrics for generated samples [32], but these are not discussed in this background.

2.2.1 Wasserstein GANs

To tackle the problem of vanishing gradients and mode collapse, Arjovsky et al. introduced the Wasserstein GAN (WGAN) [3], a GAN that has a loss function that is derived from the Earth Movers Distance (EMD), in mathematics also known as the Wasserstein distance. The EMD measures the distance between probability distributions. Informally, the EMD is defined as two different possible piles of dirt (earth), where each possible pile could be seen as a probability distribution. The EMD calculates the minimum amount of (cost) that has to be done to turn one pile into another, where the cost is the amount moved times the distance. Equation 2.5 gives the mathematical formulation for the EMD:

$$W(p_d, p_g) = \inf_{\gamma \in \Pi(p_d, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{2.5}$$

where p_d is the real data distribution, p_g is the generated data distribution and $\Pi(p_d, p_g)$ represents the set of all joint distributions of p_d and p_g. As the infimum term (inf) is highly intractable [3], WGANs reformulate equation 2.5 as the Kantorovich-Rubinstein duality [62] to make this optimization applicable in practice. Equation 2.6 contains the duality:

$$W(p_d, p_g) = \sup_{\|C\|_L \leq 1} \mathbb{E}_{x\sim p_d}[C(x)] - \mathbb{E}_{x\sim p_g}[C(x)] \tag{2.6}$$

where C needs to be a 1-Lipschitz function [3]. By providing some additional mathematics, Arjovsky et al. show that solving this in a WGAN comes down to:

$$\min_G \max_C \mathbb{E}_{x\sim p_d}[C(x)] - \mathbb{E}_{z\sim p_z}[C(G(z))] \tag{2.7}$$

where G is again the generator and C is the discriminator, now named the critic. The discriminator is renamed to the critic as it now provides information about how far both distributions are from each other, instead of providing a binary classification. Using a WGAN has shown to be an effective way to prevent vanishing gradient and greatly reduce the risk of mode collapse.

An important challenge for WGANs is enforcing the 1-Lipschitz continuity. Arjovsky et al. achieve this by clipping the weights of the critic to a fixed interval, naming it a terrible way to ensure 1-Lipschitz continuity. As an alternative Gulrajani et al. [27] add the gradient penalty to the WGAN critic. Equation 2.8 gives the formula for the added gradient penalty:

$$\lambda \mathbb{E}_{\tilde{x}\sim p_{\tilde{x}}}[(\|\nabla_{\tilde{x}} C(\tilde{x})\|_2 - 1)^2] \tag{2.8}$$

with λ a scalar and

$$\tilde{x} = \epsilon \cdot x + (1 - \epsilon) \cdot G(z) \tag{2.9}$$

where $\epsilon \sim U[0, 1]$. The WGAN with gradient penalty (WGAN-GP) showed more stable training and better performance than the standard WGAN, allowing for a wide variety of architecture with little hyperparameter tuning [27].

2.2.2 Other Optimal Transport GANs

Although the Wasserstein GAN prevents vanishing gradients for continuous data, the gradient still vanishes for discrete data (i.e. text) after a few iterations for both regular WGANs and WGAN-GP [23]. As a result, various research has been performed to apply the EMD in GANs that deal with discrete data, mainly in the field of text. Several solutions use the Sinkhorn divergence [15] to solve an entropy-regularized EMD. However, the solution is very sensitive

to hyperparameter settings [13]. As a solution, Chen et al. introduce the use of the Inexact Proximal point method for Optimal Transport (IPOT) where the hyperparameter only affects the convergence rate [13]. Compared to Sinkhorn-based solutions, the IPOT-based solution is reported to significantly improve performance and effectiveness.

2.3 Transformers

Until 2017, RNNs and in particular the LTSM and gated RNNs (see section 2.1.2) were the state-of-the-art in domains such as machine translation and language modelling problems [68]. Although various attempts had been done to improve upon this firmly established state-of-the-art, the introduction of the Transformer in 2017 [68] was the first to significantly improve it. As an additional major advantage, Transformers did this with only a fraction of the training costs compared to the existing state-of-the-art. This section discusses the most important ideas that have led to Transformers, encoder-decoder models and attention, and introduces the Transformer architecture.

2.3.1 Encoder-decoder models

Encoder-decoder models convert (encode) input to a representation with a lower dimensionality to then create (decode) a similar output. A data science website[1] makes the effective comparison with Pictionary, illustrated in figure 2.8. The first player receives a concept (input) and makes a drawing of this (the encoding). A second player 'decodes' this picture to the concept he thinks it is. Encoder-decoder models use this concept to, for instance, translate text to another language or provide captions to images.

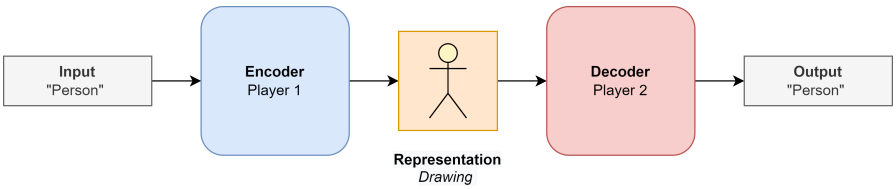

Figure 2.8: Encoder-decoder models as Pictionary.

A simple yet popular type of encoder-decoder model is the Autoencoder (AE) [32]. The AE's encoder compresses the input to a small representation compared to the input, while the AE's decoder attempts to reconstruct the data. The loss function L_{AE} for AEs is simply a form of distance between input and reconstruction, e.g. the Euclidean distance. Equation 2.10 shows an example loss function for AEs:

$$L_{AE} = \frac{1}{N}\sum_{i=1}^{N}[x_i - D(E(x_i))]^2 \tag{2.10}$$

where E is the encoder and D is the decoder.

[1] `https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a`

2.3.2 Attention

In the paper in which Vaswani et al. introduce the Transformer, they state that *"the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution"* [68]. Attention or more specifically self-attention is an essential component of the Transformer: it is the mechanism that relates different parts of a single sequence to compute the sequence's representation [68]. The attention used in the Transformer is scaled dot-product attention, which is then used to create multi-head attention.

Scaled Dot-Product Attention

Transformers use Scaled Dot-Product Attention (SDPA) as their basic attention mechanism. SDPA takes three matrices as arguments, a query Q, a key K and a value V. As the name suggests, SDPA first calculates the dot-product of the query and the key, after which it applies a scaling factor. After applying the softmax activation function, the result is multiplied by the value. Equation 2.11 gives the function for SDPA.

$$\text{SDPA}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.11}$$

where d_k is the shared dimension of both Q and K, with Q having dimension $m \times d_k$ and K having dimension $n \times d_k$. This is identical to the attention of Luong et al [43], except Vaswani et al. added the scaling factor $\frac{1}{\sqrt{d_k}}$.

Multi-head attention

Vaswani et al. found it beneficial linearly project the queries, keys and values h times with learned linear projections. For these linear project queries, keys and values the SDPA is calculated simultaneously. The concatenated result is then linearly projected into the output shape. Equation 2.12 gives the formula for this so-called Multi-Head Attention (MHA):

$$\text{MHA}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, head_h)W^O \tag{2.12}$$

where W^O is the output projection matrix and

$$\text{head}_i = \text{SDPA}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.13}$$

where W_i^Q, W_i^K, and W_i^V are the projection matrices for the query, key and value, respectively.

2.3.3 Transformer Architecture

The Transformer is an encoder-decoder model using stacked self-attention and fully connected layers for both the encoder and decoder. The model architecture is shown in figure 2.9, with the encoder and decoder on the left and right sides of the figure, respectively.

In the original architecture, the Transformer features $N = 6$ identical encoding stacks. Each encoding layer consists of an encoder MHA layer and a fully connected feed-forward network, with a residual connection around each layer. Likewise, the decoder has also 6 identical stacks, consisting of a masked decoder MHA layer, an encoder-decoder MHA layer and a feed-forward fully connected layer, also with a residual connection around each layer. Where the encoder and decoder (masked) MHA layers receive their key, query and value from the previous stack or the

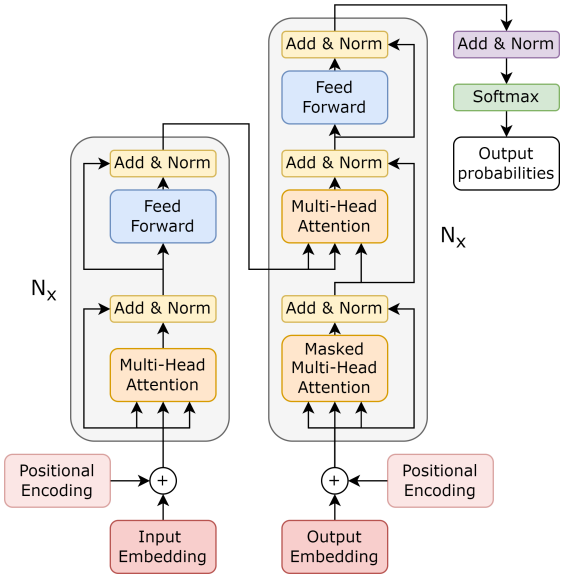

Figure 2.9: Transformer model architecture.

input, the encoder-decoder MHA layer receives the query and key from the encoder output and the value from the decoder MHA layer.

2.4 Word Embeddings

Before 2013, natural language processing mostly treated words as atomic units like an index in the dictionary or a specific N-gram. This stored no information about the similarity of certain words and also nothing about the meaning of words. That changed when the Word2Vec [47] model was introduced with the idea that words could have multiple degrees of similarity. The Word2Vec model showed that a vector can be used to represent the semantic relationships of a word in a vector, a word embedding (WE), capturing whether a word is likely to appear in similar contexts [47].

Other WE models like GloVe [52] followed and researchers quickly realized that the use of WEs pre-trained on very large datasets could greatly help create models for problems with small datasets. However, the embeddings created by models such as Word2Vec or GloVe have one big disadvantage: they create static word embeddings that disregard the context in which the sentence was used. For instance, the word *stick* can both be used as a verb and a noun, yet these static WE models would always return the same vector.

As a response to this problem, researchers started to work on contextual WE models that produce word embeddings that change based on the context the word is used in. Models such as Embeddings from Language Models (ELMo) [53] and later Universal Language Model Fine-Tuning (ULM-FiT) [30] followed. Both ELMo and ULM-FiT use LSTMs (also see section 2.1.2) and in particular ULM-FiT showed that it can effectively be pre-trained on vast amounts of data and then also effectively be fine-tuned to different language tasks [30]. Both models introduced outperformed the state-of-the-art on various problems at the time of their introduction.

2.4.1 Transformer-based Word Embeddings

The introduction of the Transformer (also see section 2.3) and the significantly improved performance on machine translation [68] led to the idea that Transformers could replace LTSM, in particular as they seemed to better represent long-term dependencies and could be trained in parallel. One of the earliest Transformer-based WE models was OpenAI's Generative Pre-Training (GPT) [57] model. GPT uses a 12-layer, decoder-only Transformer and was trained for much longer and much more data than any model before. It significantly improved the state-of-the-art [57], confirming the promising idea of Transformer-based architectures. Many Transformer-based variants like GPT-2 [58], GPT-3 [10] and XLNet [71] have followed, but for this background we will only discuss BERT.

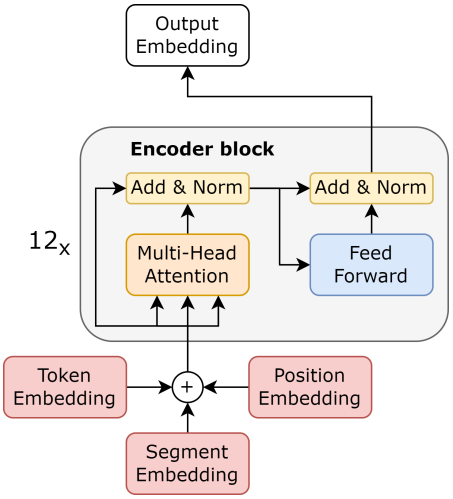

Figure 2.10: BERT model architecture.

Bidirectional Encoder Representations from Transformers

In 2019, Google AI's Devlin et al. argued that the techniques of then (amongst which GPT) would restrict the power of pre-trained word embedding models with the main argument that these techniques are all uni-directional, which limits the choice of architecture during pre-training [19]. They introduced Bidirectional Encoder Representations from Transformers (BERT), which uses only the encoder blocks of the Transformer as shown in figure 2.10. To alleviate the unidirectional constraint, BERT uses a Masked Language Model that randomly masks some input tokens with the goal to predict the masked tokens. For a part of the training, BERT even masks out the entire following sentence to be able to perform next-sentence prediction. In contrast to most others at that time, BERT has a deeply bidirectional architecture instead of a shallow concatenation of two unidirectional models. More importantly, BERT was shown to be the first fine-tuning-based model that outperformed state-of-the-art on both token *and* sentence level tasks [19].

Chapter 3

Related work

This chapter discusses works and datasets related to this research. First, section 3.1 elaborates on relevant datasets for Aspect-Based Sentiment Analysis (ABSA). Next, we discuss state-of-the-art ABSA models in section 3.2 Last, section 3.3 discusses works related to applying Generative Adversarial Networks (GANs) to natural language problems. In both section 3.2 and section 3.3 we focus on works using Transformer-based word embedding models.

3.1 Datasets

Aspect-Based Sentiment Analysis (ABSA) has become a popular research topic due to the many useful applications [11] and various public datasets available. In this section, we discuss the popular SemEval and Twitter dataset(s), as well as the more challenging MAMS dataset. Table 3.1 shows the details of these databases.

Dataset	**Domain**	**Pos.**	**Neutr.**	**Neg.**	**Total**
SemEval-2014 [56]	Reviews (Laptops)	1328	629	994	2951
	Reviews (Restaurants)	2894	829	1001	4724
SemEval-2015 [55]	Reviews (Laptops)	1644	185	1094	2923
	Reviews (Restaurants)	1652	98	749	2499
SemEval-2016 [54]	Reviews (Laptops)	1540	154	869	2563
	Reviews (Restaurants)	1802	104	623	2529
Twitter [20]	Social Media (Various)	1735	3470	1735	6940
MAMS [33]	Reviews (Restaurants)	4183	6253	3418	13854

Table 3.1: Dataset details, with samples labelled *Conflict* excluded.

The International Workshop on Semantic Evaluation (**SemEval**) featured ABSA tasks in 2014 [56], 2015 [55] and 2016 [54]. All these datasets contain laptop and restaurant reviews in English, where in particular the 2014 datasets have become a benchmark for deep-learning-based ABSA approaches [8, 33]. The SemEval tasks are based on aspect-term and aspect-category, where the latter groups terms into categories and is only featured in the restaurant datasets.

The **Twitter** dataset by Dong et al. [20] was also introduced in 2014 and is frequently used in ABSA research, although less popular than SemEval-2014 [8]. This dataset is bigger than the SemEval datasets, with neutral samples making up half of the dataset. Only the aspect-term task is featured in the dataset [20].

Last, the **Multi-Aspect Multi-Sentiment (MAMS)** dataset [33] introduced in 2019 also concerns restaurant reviews. However, the dataset only features sentences with multiple aspects that each have a different sentiment for each aspect. As a result, the MAMS dataset usually leads to lower performance and can therefore be considered more challenging [33]. Additionally, the dataset is also significantly larger with more than 10K samples for both tasks.

3.2 Aspect-Based Sentiment Analysis

A survey by Brauwers and Frasincar [8] shows that more than 60 works on ABSA have been published in the last decade. Since 2019, Transformer-based models make up most of the state-of-the-art architectures, practically all using Devlin et al.'s Bidirectional Encoder Representations from Transformers (BERT) [19] as a core component. In this section, we provide an overview of various Transformer-based ABSA approaches relevant to this research.

Type	Binary	Sentence
QA	Yes	The polarity of the aspect safety of X is positive
NLI	Yes	X, safety, positive
QA	No	What do you think of the safety of X?
NLI	No	X, safety

Table 3.2: Example auxiliary sentences for entity X as generated by Sun et al. [67]

One of the earliest applications of BERT for ABSA is that of Sun et al. [67]. They create an auxiliary sentence based on Question Answering (QA) or Natural Language Inference (NLI) to make ABSA a sentence pair classification task, using both binary and non-binary questions. Table 3.2 contains example sentences for each combination. The sequence output (CLS) vector of $\text{BERT}_{\text{BASE}}$ is followed by a fully connected layer with a softmax classification layer, achieving an 85.9% accuracy on the aspect-category sentiment classification task of SemEval-2014. Hoang et al. [28] use a similar approach.

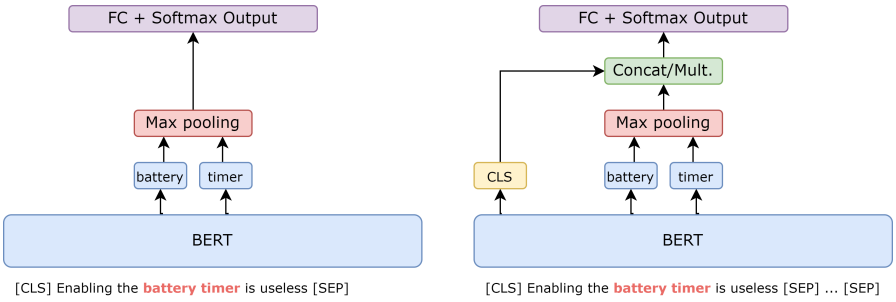

Figure 3.1: Variants proposed by Goa et al. [21] without (left) and with (right) CLS vector.

Goa et al. [21] use the word embeddings from the target followed by max pooling, instead of or in addition to the CLS vector from $\text{BERT}_{\text{BASE}}$. Variants that use the CLS vector concatenate or multiply this with the pooling outcome, after which all variants end fully connected layer with a softmax output layer. Results on SemEval-2014 show that the variant without CLS performs best accuracy-wise, but multiplication with CLS (laptops) or even Sun et al.'s aforementioned approach (restaurants) performs better on F_1-score.

Hu et al. [31] use a multi-target extractor to determine where the aspect is located and use $\text{BERT}_{\text{LARGE}}$. This extractor mainly consists of a BERT model with a fully connected layer with softmax to determine the start and end position of a target. They combine three years of SemEval datasets, making the results hard to compare, but claim to achieve state-of-the-art results at that time. Xu et al. [69] implement a similar approach but post-train $\text{BERT}_{\text{LARGE}}$ for 70K-140K steps on a review dataset they introduce in their paper. They show that post-training on a related domain can significantly increase the performance, yet their method (BERT-PT) comes at a severe computational cost that is even larger due to the use of $\text{BERT}_{\text{LARGE}}$.

Zeng et al. [73] introduce a more complex approach called Local Context Focus (LCF). LCF uses a local and global context processor, both using BERT followed by two additional layers of Multi-Head Attention (MHA, also see section 2.3.2). For the local context processor, they add a Context features Dynamic Mask (CDM) or Context features Dynamic Weighted (CDW) layers between the two last MHA layers. The output from both the local and global context processors is then combined, followed by another MHA layer and a pooling layer to produce the output. Despite its complexity, the LCF approach yields impressive results with 87.1% and 82.5% accuracy on the SemEval-2014 restaurants and laptops dataset respectively.

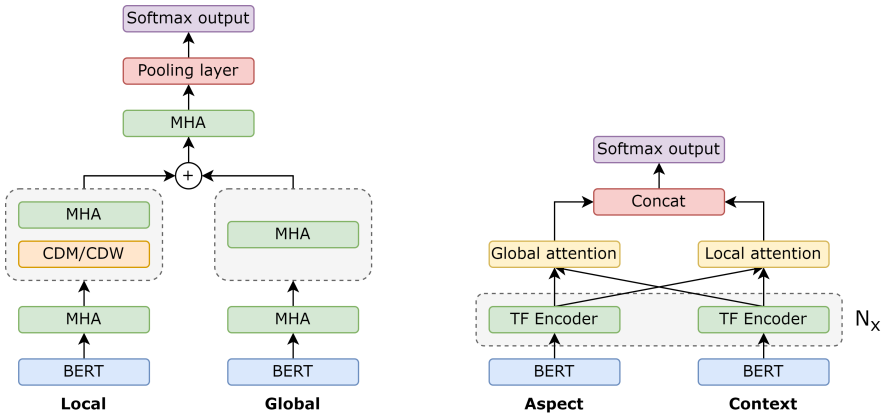

Figure 3.2: Left: Local Context Focus [73]. Multi-Attention Network (MAN) [70].

An alternative approach is proposed by Xu et al. [70] in the form of the Multi-Attention Network (MAN). Both aspect and context are put into their own BERT models, each followed by several Transformer encoders to create a hidden representation for both the aspect and context. The outputs of both threads are then jointly used as input for a global and local intra-attention layer. The intra-attention outputs are then concatenated and connected with a fully connected softmax layer to produce the output. MAN is significantly outperformed by LCF and also does not perform significantly better than the simpler architectures outlined before. Figure 3.2 shows a schematic of the LCF and MAN architectures.

Early 2021 Karimi et al. [35] introduced the first application of adversarial training in ABSA, BERT Adversarial Training (BAT), build on the post-trained BERT-PT approach by the aforementioned Xu et al. [69]. They use an adversarial white-box attack [35] based on the gradient of the loss function on BERT's input embeddings, by solving the minimization problem shown in equation 3.1:

$$r_{adv} = \arg \min_{r, \|r\| \leq \epsilon} \log p(y|x + r; \hat{\theta}) \tag{3.1}$$

where r_{adv} is the perturbation, ϵ the maximum perturbation, $p(y|x;\theta)$ the probability that

the correct label y is predicted for input x on a model with parameters θ and $\hat{\theta}$ is a copy of the model parameters to prevent gradient propagation. BAT showed to improve BERT-PT in practically all cases reaching 86.0% accuracy on the SemEval-2014 restaurant dataset. However, the choice of ϵ can significantly impact performance and [35] shows optimal ϵ values differ per dataset.

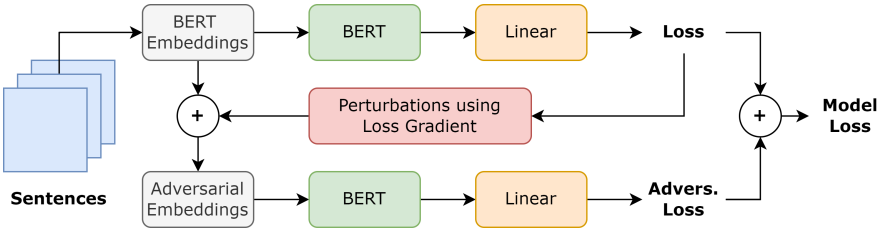

Figure 3.3: BERT Adversarial Training (BAT) from Karimi et al. [35].

In the same year, Su et al. [66] introduced Progressive Self-Supervised Attention (PSSA) learning, a new method of training attention-based ABSA models. They show that attention mechanisms are susceptible to context words that very often occur with the same sentiment. This results in wrong predictions, as too much attention is given to these words. PSSA masks out the word that is given the most attention in the previous training phase to address this problem, leading to state-of-the-art performance on both the SemEval-2014 datasets and the Twitter dataset. Additionally, they show that their method not only works for BERT-based models, but also for older attention-based models.

3.3 GANs for Natural Language

Although initially mainly used in computer vision applications, Generative Adversarial Networks (GANs) have been applied successfully in the field of natural language [1, 32]. This section discusses two variations of using GANs for natural language. First, we discuss GANs that have the goal to generate text and what methods they use to deal with the discrete nature of text. Second, we focus on GANs applied to Transformer-based word embedding models, in particular those that use GANs to improve classification performance rather than to generate text.

3.3.1 GANs for Text Generation

Several recent surveys show that a significant amount of works have been published that use GANs for text generation [1, 17]. A significant challenge for GANs is that they are designed to generate continuous data, whereas the discrete nature of text limited the effectiveness of early GANs [1]. We discuss how this problem can be overcome.

Reinforcement Learning-based approaches

A popular approach for generating text with GANs is Reinforcement Learning (RL), where an agent learns its behaviour by punishment or reward based on its actions. Promising results have been shown in GANs in which the generator is the agent and the discriminator guides the training by determining the reward. SeqGAN [72] was one of the earliest applications, modelling the GAN's generator as a stochastic RL policy and using RL gradient policy update. Lin et al. [39] introduced the use of adversarial ranking, RankGAN, where the discriminator

is a ranker that ranks human-written and machine-written sentences. MaliGAN [12] is another approach that uses a maximum-likelihood training objective for GANs to address the challenge of backpropagating on discrete data like text.

However, RL-based approaches face some major challenges that in general lead to poorly generated text [1]. First, RL-based approaches have a gigantic state-action space that can only partially be explored, leaving a large part of potential actions unconsidered. Additionally, these approaches can get easily trapped at local optimums and gradient estimates using RL gradient policy can have very high variances [44, 74].

Other approaches

Considering the drawbacks of RL-based approaches, researchers have tried to find alternative solutions that address these drawbacks, so-called *RL-free* models. A popular method is to use a continuous approximation of discrete sampling [1], such as using Gumbel-softmax in GSGAN [37] or soft-argmax operator in TextGAN [74] as a trick to provide a continuous approximation of the distribution. RelGAN [50] is one of the first approaches that can generate realistic text using Gumbel-softmax and produces state-of-the-art results.

An alternative approach to approximate a continuous contribution is to use the Earth Mover's Distance (EMD) as an alternative GAN training objective, as discussed earlier in section 2.2.2. The Future Mover's GAN (FM-GAN) [13] implements this, leading to a model with a critic instead of a discriminator that can overcome problems such as mode collapse.

3.3.2 GANs with Transformer-based Word Embeddings

GANs and Transformer-based word embedding models are both very popular, and various works applying GANs to text exist [8, 32, 1]. However, the research on combining GANs with Transformer-based word embedding models such as BERT is limited, with the text-to-image domain as an important exception [18, 48, 49]. In this section, we discuss the work on using GANs with Transformers-based architectures.

Shang et al. [65] use the TextGAN [74] and Transformer GAN (TransGAN) [34] architecture to generate additional augmented data to improve sentiment classification using BERT-variant RoBERTa. TransGAN was originally built to generate images without using any convolutional operation but was tweaked to generate text instead. They have shown that the combination of the TransGAN with RoBERTa yields superior results and that using the TransGAN for data augmentation increases performance regardless of which classifier is used.

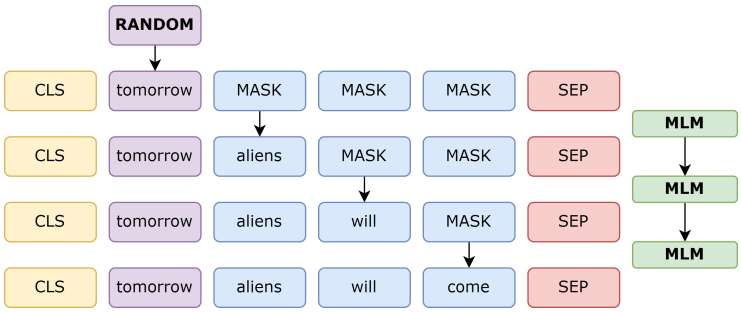

Figure 3.4: Using BERT's Masked Language Modelling (MLM) to generate a lie.

Barsever et al. [4] predict random sentences to better detect lies by using BERT's Masked Language Modeling. They first generate a random seed for the first token, after which they

use BERT to predict the remainder of the tokens. This is visualized in figure 3.4. The second instance of BERT is then used as a discriminator to detect lies. They show that this method significantly increases performance compared to other published methods and is state-of-the-art on the used dataset. However, the paper does not include any result without using the GAN, leaving it unclear if any performance is gained.

GAN-BERT

A different method, GAN-BERT, is proposed by Croce et al. [14]. They argue that the huge amount of annotated data that is required to make Transformer-based word embedding models like BERT perform is highly unpractical: obtaining this data is very expensive and time-consuming unless a large, high-quality dataset already exists. As unlabelled examples for a target task are often easy to collect, they propose a semi-supervised approach based on the Semi-Supervised GAN (SS-GAN) [64].

In SS-GANs, a k class discriminator is trained for a $(k+1)$ class objective, where the $(k+1)^{\text{th}}$ class is for artificially generated samples. SS-GANs split the loss into a supervised loss and an unsupervised loss, where supervised loss corresponds to the cross-entropy loss and the unsupervised loss measures the error for labelling a real example as artificial and vice-versa. SS-GANs use a feature matching loss in the generator, aiming to make sure that hidden layer activations are similar for generated and real samples across batches. In contrast to the regular GAN, the generator and discriminator now play their own minimization game with different losses instead of a shared objective.

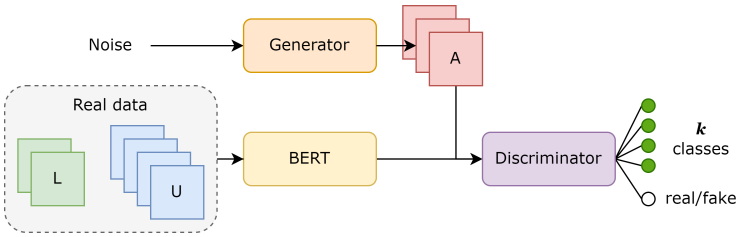

Figure 3.5: GAN-BERT Architecture.

Rather than a GAN that generates text, Croce et al.'s GAN-BERT only generates the encoding that BERT produces as output with a very simple one-layer MLP as both generator and discriminator. Figure 3.5 gives an overview of the rather simple GAN-BERT architecture. GAN-BERT uses $\text{BERT}_{\text{BASE}}$. In their experiments, GAN-BERT is trained for only 3 epochs and uses a ratio of 100 unlabelled samples per labelled sample, while repeating labelled samples to make sure every batch contains them. Despite the extremely simple generator and classifier, GAN-BERT shows a significant increase in performance when limited annotated data is available, for all 6 tested natural language tasks tested.

Multi-Task GAN-BERT (**MT-GAN-BERT**) has been proposed by Breazzano et al. [9] as an extension to GAN-BERT based on MT-DNN [41], to allow small training sets even further. MT-GAN-BERT shares a single BERT model for multiple classification tasks, each with its own generator and discriminator. This makes it possible to combine multiple datasets that share a domain but have different labels, to improve model robustness and generalisation. Although it seems that not all natural language tasks are fit to combine in a multi-task setting, MT-GAN-BERT reports promising results for those that are related.

Chapter 4

Method

This chapter discusses our method to investigate the effect of applying Generative Adversarial Networks (GANs) on a Transformer-based Aspect-Based Sentiment Analysis (ABSA) method. First, we specify the task that our models will solve, followed by our baseline model. Next, we state how will implement both a regular and Wasserstein GAN variant of this baseline model, followed by the datasets used. Last, we discuss how we evaluate our models.

4.1 Task

In this research, we combine aspect detection and sentiment classification in one task. For a given piece of text, this means that the model has to determine *if* a sentiment about the given aspect is expressed and if so, *what* sentiment is expressed. Usually, these two sub-tasks are used separately, but we combine the two to create a more challenging task. If a text says nothing about an aspect, we apply the label *Not Mentioned (NM)*. Otherwise, the sentiment polarity labels *Positive*, *Neutral*, *Negative* and optionally *Conflict* are used. Figure 4.1 visualises this.

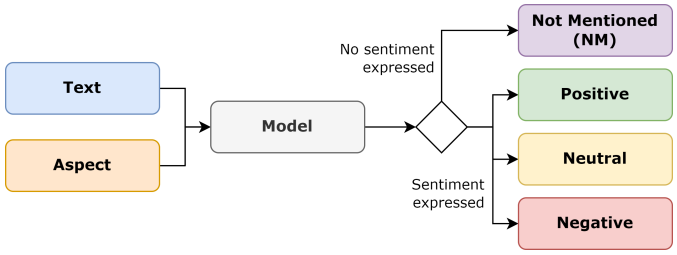

Figure 4.1: Visualisation of the combined detection and sentiment classification task.

We group aspects into 5 or 8 categories depending on the dataset used. For example, terms like *salad*, *lasagne* and *zucchini* will be grouped together under the aspect-category *food*. As a result of the categories, we do not have to detect an (almost) infinite amount of terms, which makes it possible to perform both detection and sentiment classification in one task. This makes that each text has 5 or 8 tasks, one for each category present in the dataset. The categories for each dataset can be found in table 4.3 on page 25.

4.2 Baseline method

We first select a Transformer-based ABSA method that will function as a baseline. This research focuses on using limited amounts of labelled training data and not necessarily on improving state-of-the-art performance, allowing us to select a not-too-complex model that has a decent performance. Considering that BERT is designed to work with a very minimal classification part appended and taking the ABSA methods as seen in 3.2 into account, we chose to use the method of Sun et al. [67] as the baseline model. The method has decent performance with an acceptable, modest difference in performance when compared to more complex state-of-the-art models such as LCF [73]. However, the model is relatively simple to implement and does not use $\text{BERT}_{\text{LARGE}}$which is deemed unpractical due to its large size that results in long training times and high GPU memory requirements.

Sun et al.'s model [67] generates an auxiliary sentence to make BERT perform a sentence pair classification task. For this research, we use their non-binary question-answering method to generate this auxiliary sentence. This means that for a given target category, we add the question *what do you think of the {target category} of it?* after the original sentence, separating them with a `[SEP]` token. Figure 4.2 shows an example of this, together with the architecture of the model.

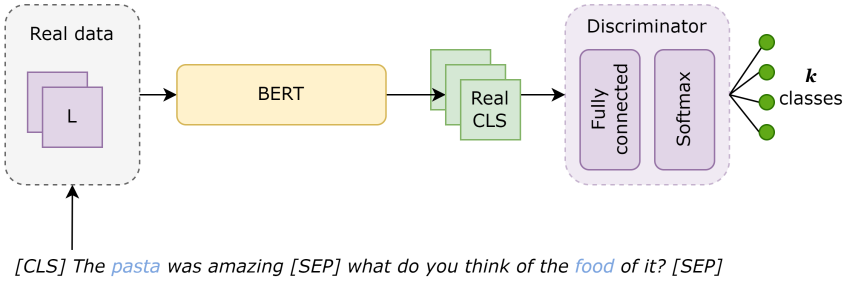

Figure 4.2: Overview of the baseline method based on Sun et al. [67]

We use the default BERT preprocessor to tokenize sentences using a maximum sequence size of 128 tokens, considering the relatively small size of the input texts. These are put into $\text{BERT}_{\text{BASE}}$, which will produce a sequence output (CLS) vector of size 768 representing the entire sentence. The CLS vector is put into a single fully connected layer of the same dimension with Leaky ReLU as activation function. During training, this layer uses a dropout percentage of 10% to improve regularization. In the end, there is a fully connected softmax output layer. The baseline model is trained for 8 epochs using a batch size of 16 and uses the categorical cross-entropy loss with an Adam [36] optimizer with a learning rate of $1e{-}5$.

4.3 Regular GAN

As a basis for our GAN, we will apply the semi-supervised GAN-BERT [14] as earlier discussed in section 3.3.2 to Sun et al.'s baseline method. The application is straightforward, considering that GAN-BERT's discriminator architecture is similar to that of the classification layers of the baseline. In practice, $\text{BERT}_{\text{BASE}}$ and the classification layers stay the same, except for the extra class for artificial samples that is added to the output layer. The main change is the addition of a generator that feeds artificial sequence output (CLS) vectors to the discriminator. A schematic overview of the combination of the two models can be seen in figure 4.3.

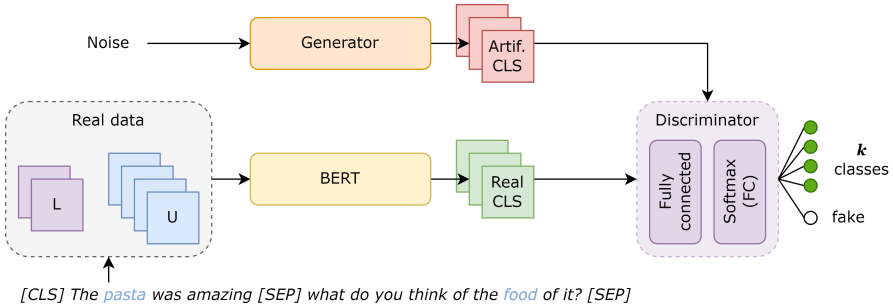

Figure 4.3: GAN-BERT [14] applied the approach of Sun et al. [67]

As a discriminator does not necessarily need real samples to be labelled, it allows using of unlabelled data next to the labelled data in training. To accommodate this, the model uses distinct losses for the generator and discriminator as introduced by the Semi-Supervised GAN (SS-GAN) [64]. Equation 4.1 states the formula for the discriminator loss, where p_d denotes the distribution of the real data, p_z be the distribution of the latent vector z, $D(\hat{y} = y|x)$ the probability that the discriminator predicts the class y given sample x, $k+1$ the class used to denote an artificial sample and $G(x)$ the output of the generator for input x.

$$\begin{aligned} L_D =& L_D^{sup.} + L_D^{unsup.} \\ L_D^{sup.} =& -\mathbb{E}_{x,y\sim p_d} \log(D(\hat{y} = y|x)) \\ L_D^{unsup.} =& -\mathbb{E}_{x\sim p_d} \log(1 - D(\hat{y} = k+1|x)) \\ & -\mathbb{E}_{z\sim p_z} \log(D(\hat{y} = k+1|x = G(z))) \end{aligned} \tag{4.1}$$

The discriminator loss L_D consists of a supervised and an unsupervised component. The supervised component $L_D^{sup.}$ is calculated based on the labelled data only and corresponds to the categorical cross-entropy loss as used in the baseline. For the unsupervised component $L_D^{unsup.}$, the first part takes the categorical cross-entropy loss based on the probability that a real sample is not classified as artificial. It then adds the same loss for the probability that an artificial sample is classified as such, after which both components are summed together.

$$\begin{aligned} L_G =& L_G^{feat.} + L_G^{unsup.} \\ L_G^{feat.} =& \|\mathbb{E}_{x\sim p_d} a(x) - \mathbb{E}_{z\sim p_z} a(G(z))\| \\ L_G^{unsup.} =& -\mathbb{E}_{z\sim p_z} \log(1 - D(\hat{y} = k+1|x = G(z))) \end{aligned} \tag{4.2}$$

The generator loss L_G shown in equation 4.2 also consists of two components. Its unsupervised component $L_G^{unsup.}$ takes the cross-entropy loss based on the probability that an artificial sample is classified as real. Additionally, SS-GANs use a feature matching loss $L_G^{feat.}$ aiming to make sure that hidden layer activations $a(x)$ are similar for artificial and real samples.

This research uses two generator architectures, one with one hidden layer ($G_{h=1}$) and one with three hidden layers ($G_{h=3}$). Both generators receive a latent vector sampled from a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. This noise is put into the fully connected hidden layers using Leaky ReLU as activation function and followed by a 10% dropout. Finally, each generator consists of an output layer consisting of 768 units with a linear activation which outputs the artificial CLS. The dimensions for the two different generators are shown in figure 4.4.

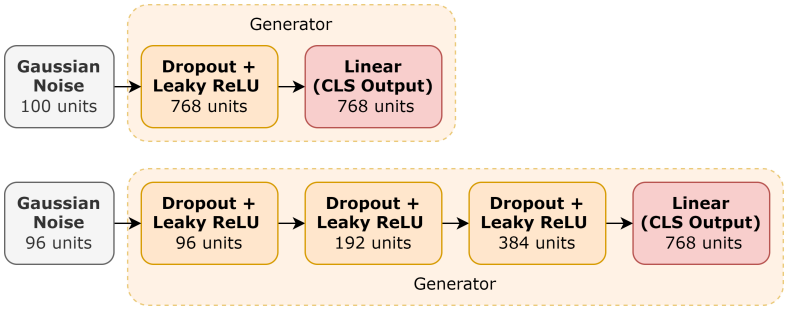

Figure 4.4: Architecture for $G_{h=1}$ (top) and $G_{h=3}$ (bottom).

In GAN-BERT, Croce et al. train both the generator and the discriminator once per batch. However, empirical tests show that the generator loss quickly increased, while the unsupervised discriminator loss quickly dropped to near-zero values. This results in sub-optimal performance. To keep the generator sufficiently powerful, we introduce the maximum generator loss L_G^{max}. If $L_G > L_G^{max}$ directly after the generator was trained, the generator is trained again. We consider the generator to be sufficiently powerful if, on average, the discriminator assigns a 70% or higher probability that a generated sample is real. This corresponds to a maximum loss value $L_G^{max} = 0.5$. To avoid very long training times, the generator is never updated more than 5 times per batch. Both the generator and discriminator are trained for 8 epochs using batch size 16 and use the same Adam optimizer as in the baseline model with a learning rate of 1e−5.

4.4 Wasserstein GAN

Aiming to improve the diversity of artificial samples, we also propose a Wasserstein GAN (WGAN) [3] variant with gradient penalty [27]. The main challenge for applying a WGAN to a classification problem is that it has a critic instead of a discriminator. This critic could help to avoid problems such as vanishing gradients and mode collapse, but it cannot be used for classification. To still be able to classify, our WGAN uses a hybrid approach with both a critic and a discriminator.

This hybrid approach is shown in figure 4.5 and can be viewed as two models that are trained together. One of these models is a classification or discrimination model, essentially our regular GAN but without the generator. Instead, it only focuses on the semi-supervised discrimination task, receiving its artificial samples from the other model, a regular WGAN. This WGAN attempts to produce artificial sequence output (CLS) vectors by minimizing the Wasserstein distance while enforcing the 1-Lipschitz constraint by applying gradient penalty (GP). Please see section 2.2.1 for more details on the Wasserstein distance and gradient clipping. Similar to the regular GAN, the same two generator architectures are used. The critic has a similar architecture to that of the discriminator, but its output layer now only consists of one unit with linear activation.

For each batch, the training step begins with training the WGAN. In line with the recommendations in [3, 27] both generator and critic are updated using the RMSProp optimizer with a learning rate of 5e−5, the critic is updated 5 times for each time the generator is updated and the GP-weight is set to 10. Both BERT, which supplies the real CLS vectors, and the discriminator are not updated during this process. After the WGAN has been trained, the artificial samples it generates are used to train the classification model. In this step, the WGAN supplies the artificial CLS vectors but is not updated. The discriminator uses the same loss function L_D

as defined in section 4.3 and an Adam optimizer with the same learning rate of 1e−5 as in the regular GAN.

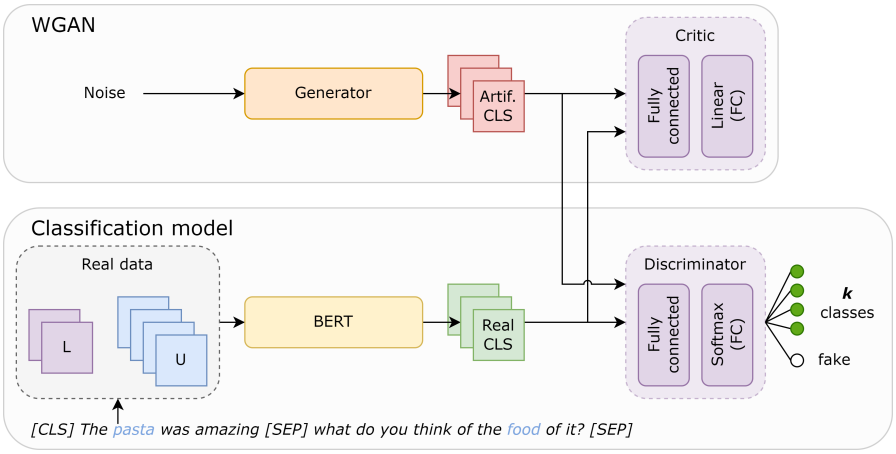

Figure 4.5: Schematic overview of our hybrid WGAN approach.

Similar to the regular GAN, the WGAN faces the challenge that the generator might not be sufficiently powerful to fool the discriminator. Considering that the generator loss is now the Wasserstein distance, this cannot be set to a predefined target value such as the maximum generator loss. Instead, we manually set the number of times the generator is trained per step to $n_G^{train} = 3$ after empirical results showed this to be the best-performing value.

4.5 Data

In this section, we describe the data that is used to train and evaluate the methods presented earlier in this chapter. First, we describe the labelled data that both the baseline and GAN models will use for training and evaluation. Next, we specify how we use unlabelled data, which can be added to the GAN models next to the labelled data. Last, this section will also discuss the preprocessing of the data.

4.5.1 Labelled data

The Multiple Aspects Multiple Sentiments (MAMS) [33] is used as the main dataset for this research. This dataset is considered to be more challenging than others, considering that each sentence contains at least two different aspects with at least two different sentiment polarities. Additionally, this dataset has three more categories compared to the SemEval-2014 [56] dataset. We use the restaurant reviews dataset, where table 4.1 shows an overview of the distribution of the different sentiment polarities within the dataset, where *NM* denotes *Not Mentioned*.

Type	Sentences	Positive	Neutral	Negative	NM	Total
Train	3149	1929	3077	2084	18102	25192
Validation	400	241	388	259	2312	3200
Test	400	245	393	263	2299	3200

Table 4.1: MAMS dataset statistics for all aspect-categories combined. [33]

Next to the MAMS dataset, we use the SemEval-2014 [56] dataset to compare the performance of our baseline implementation to the reported performance by Sun et al. [67]. We again use the restaurant reviews dataset, which compared to MAMS has the additional sentiment polarity *Conflict*. Table 4.2 shows an overview of the distribution of the different sentiment polarities within the dataset.

Type	**Sentences**	**Positive**	**Neutral**	**Negative**	**Conflict**	**NM**	**Total**
Train	3041	2177	500	839	195	11494	15205
Test	800	657	94	222	52	2975	4000

Table 4.2: SemEval-2014 dataset statistics for all aspect-categories combined. [56]

In both datasets, some categories are significantly more prevalent than others. Table 4.3 shows the distribution of categories in the training data for the respective datasets. It can be seen that food, staff and miscellaneous occur most, while the ambience and the price are only mentioned infrequently.

Dataset	**Ambience**	**Miscellaneous**	**Food**	**Menu**	**Place**	**Price**	**Service**	**Staff**
MAMS	324	954	2307	475	694	322	631	1383
SemEval-14	431	1130	1232			321	597	

Table 4.3: Number of sentences in which a sentiment is expressed per category.

As this research focuses on performance with a reduced amount of labelled training data available, most experiments do not use all training data. Therefore, we introduce the ratio of labelled data r_L, which represents the percentage of sentences in the training dataset that is used. Based on the performance of the baseline method with values of r_L ranging from 2.5% to 100%, we use the ratios of labelled data $r_L \in \{10\%, 20\%, 50\%, 100\%\}$ for our (W)GAN experiments.

If the ratio of labelled data is smaller than 100%, sentences will be randomly sampled from the dataset. The sampled data is then balanced using oversampling for all sentiment polarities, while the label *Not Mentioned (NM)* is undersampled to match the sentiment polarities. The latter is due to the large number of cases in which an aspect category is not present. Balancing is only applied to the training data and is not applied for validation or testing. Table 4.4 shows to which number the MAMS and SemEval-14 datasets are balanced for $r_L = 100\%$.

Dataset	**Positive**	**Neutral**	**Negative**	**Conflict**	**None**	**Sampling target**
MAMS	1929	3077	2084		18102	3077
SemEval-14	2177	500	839	195	11494	2177

Table 4.4: Distribution of sentiment polarities and corresponding sampling target (all data)

4.5.2 Unlabelled data

Both our regular and Wasserstein GAN allow for the usage of unlabelled data in addition to the labelled data. We combine the training data of the SemEval-2014 [56], SemEval-2015 [55] and SemEval-2016 [54] restaurant review datasets to create one big collection of unlabelled data. Table 4.5 shows an overview of the distribution of sentiment polarities within these datasets. Please note that the labels themselves will not be used.

Dataset	Sentences	Categories	Pos.	Neut.	Neg.	Con.	NM	Total
SemEval-2014	3041	5	2177	500	839	195	11494	15205
SemEval-2015	1315	13	1060	49	345	24	15617	17095
SemEval-2016	2000	12	1464	95	662	37	21742	24000
Total	**6356**		**4701**	**644**	**1846**	**256**	**48853**	**56300**

Table 4.5: Statistics for the datasets used for unlabelled data.

Considering that this research aims to investigate the effect of the amount of unlabelled data, we also define the unlabelled data ratio r_U. This ratio represents the amount of unlabelled data compared to the amount of labelled data. Let the amount of labelled data for labelled data ratio r_L be denoted by $|L_{r_L}|$, the amount of unlabelled data $|U|$ is then given by:

$$|U| = r_U |L_{r_L}| \tag{4.3}$$

Similar to the labelled data, the unlabelled data is sampled on a sentence level to simulate collecting unlabelled data in the real world. Where Croce et al.'s GAN-BERT uses a set value for the unlabelled data ratios r_U, we conduct experiments for different values. For $r_L \in \{10\%, 20\%\}$ we use $r_U \in \{0, 1, 2, 4, 8, 16\}$, for $r_L = 50\%$ we use $c_U \in \{0, 1, 2, 4\}$ and for $r_L = 100\%$ we use $c_U \in \{0, 1, 2\}$. Only the smaller unlabelled data ratios are used for larger labelled data ratios to assure sufficient unlabelled data is available and to keep training times limited.

4.6 Evaluation

The survey by Brauwers and Frasincar [8] shows that the accuracy and macro-F_1-score are the most widely used metrics for the evaluation of deep-learning ABSA methods. While accuracy is a very straightforward metric, the macro-F_1-score can be considered to be more strict as it weighs each class equally, regardless of the number of occurrences. Next to these metrics, we use precision and recall for the error analysis. Expressed in terms of true positives (TP), false positives (FP) and false negatives (FN), the precision, recall and F_1-score are given by:

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F_1 &= 2\frac{precision \cdot recall}{precision + recall} \end{aligned}$$

The metrics in the equations above are all measured per class. The macro-F_1-score is the average of the F_1-scores for all individual classes. We conduct an experiment for every combination of (W)GAN, generator architecture, r_L and r_U, with each experiment being repeated at least three times. Excluding the baseline, this has resulted in a total of 582 experiments for the 76 combinations mentioned above.

Chapter 5

Results

In this chapter we present the results of the experiments defined in the previous chapter, aiming to draw valid conclusions on our research questions later. In the first section, we start with the effect of our Generative Adversarial Networks (GANs) on the overall performance of our Transformer-based Aspect-Based Sentiment Analysis (ABSA) model. We then take a closer look at the performance for the two sub-tasks we combined, aspect-category detection and sentiment classification. Finally, we take a closer look by analysing errors and attempting to explain model performance.

5.1 Overall performance

Before we can analyse and understand the effect of adding a GAN, in particular when less annotated training is data available, we first need to know how the ABSA model performs with less training data in general. Figure 5.1 shows how the baseline performance decreases as the ratio of labelled data r_L decreases, with the error bands showing the 95%-confidence interval over at least 5 runs.

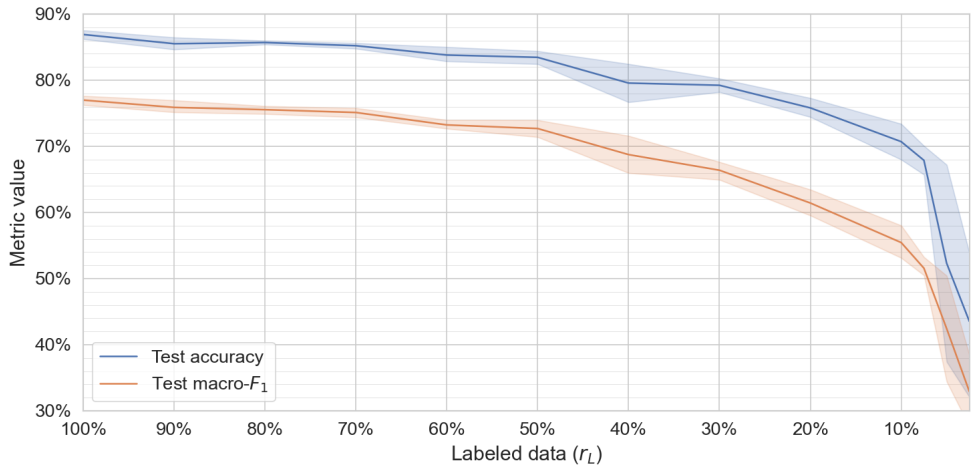

Figure 5.1: Overall baseline performance with r_L from 100% to 2.5%.

We can clearly see that performance decreases more steeply as the amount of labelled data available decreases. At first, this decrease is limited: at 50% of labelled data, both the accuracy and macro-F_1-score have dropped by 3.5 and 5 points respectively. However, when the amount of training data is reduced to 20%, roughly halving the data again, performance has decreased by respectively 11 and 16 points total. After another division by two in terms of training data, the performance takes a significant dive down. As the positive effect of adding additional training data on performance can be expected to decrease as more training data is already available, it is not surprising that the difference in performance converges as r_L increases. The exact values for all data points in figure 5.1 can be found in the Appendix.

Adding one of our GANs almost always improves the overall performance of the ABSA model, even without the use of unlabelled data. Figure 5.2 shows that only the Wasserstein GAN (WGAN) with one hidden layer is an exception, performing below baseline levels for a ratio of labelled data $r_L \geq 50\%$. The largest improvements can be seen when small amounts of labelled data are available, where both accuracy and macro-F_1-score improve by up to 5 points.

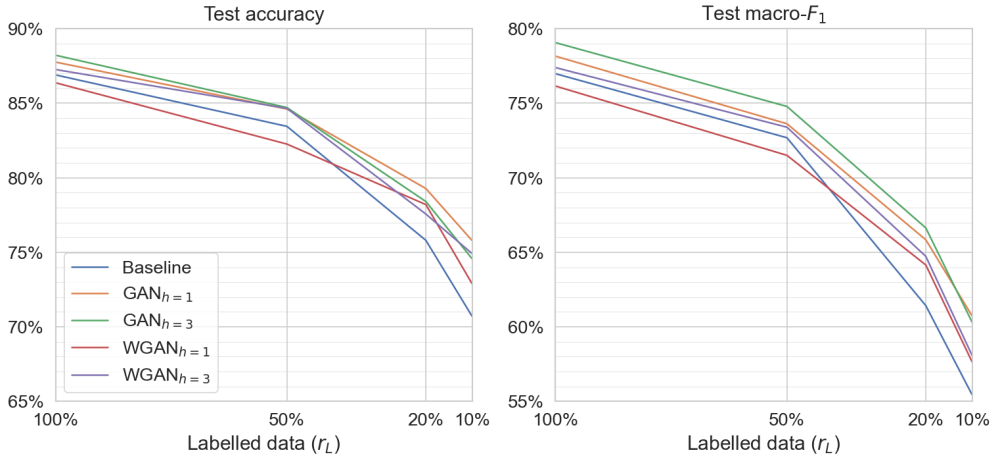

Figure 5.2: Overall performance for GANs without using unlabelled data.

Table 5.1 shows that regular GANs outperform WGANs in practically all cases, albeit by a few points at most. This could be due to the fact that the gradients of the discriminator are directly used by regular GANs to generate artificial samples, similar to adversarial training methods such as that of Karimi et al. [35]. Wasserstein GANs do not have this advantage, having to generate very realistic samples instead of being able to directly attack the discriminator.

	$r_L = \mathbf{10\%}$		$r_L = \mathbf{20\%}$		$r_L = \mathbf{50\%}$		$r_L = \mathbf{100\%}$	
Model	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**
Baseline	0.707	0.554	0.758	0.614	0.834	0.727	0.869	0.770
GAN$_{h=1}$	**0.758**	**0.607**	**0.793**	0.658	0.846	0.736	0.878	0.782
GAN$_{h=3}$	0.746	0.603	0.784	**0.666**	**0.847**	**0.748**	**0.882**	**0.791**
WGAN$_{h=1}$	0.729	0.576	0.782	0.641	0.823	0.715	0.864	0.761
WGAN$_{h=3}$	0.749	0.581	0.776	0.647	**0.847**	0.734	0.873	0.774

Table 5.1: Overall performance for GANs without using unlabelled data.

Particularly for larger amounts of labelled data, it stands out that both regular and Wasser-

stein GANs with 3 hidden layers generally produce better results than their variants with only one hidden layer. For $r_L = 10\%$, however, the variants with only one hidden layer perform better. Likely, it becomes easier for the discriminator to learn more complex patterns in BERT sequence output (CLS) vectors as more training data is available, while generators with one hidden layer find it more difficult to reproduce such patterns artificially.

While adding a GAN with only labelled data already improves performance, choosing the right amount of unlabelled data can significantly add to this. Figure 5.3 shows the performance for the best-performing ratios of unlabelled data r_U, selected based on the macro-F_1-score. Where the GANs without unlabelled data kept relatively close to the baseline performance, we now clearly see very significant gaps. This difference is particularly visible for lower ratios of labelled data r_L, with an additional performance increase by over 3 and 5 points for the accuracy and macro-F_1-score respectively.

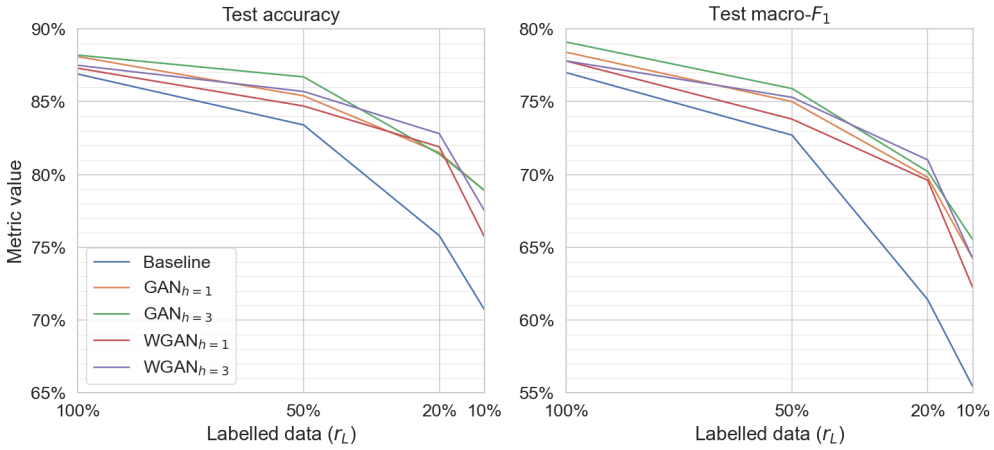

Figure 5.3: Overall performance for the best-performing ratios of unlabelled data r_U.

In total, using a GAN with the right amount of unlabelled data can increase accuracy by up to 8 points and the macro-F_1-score by up to 10 points when lower amounts of labelled data are available. The difference in performance again converges as r_L increases, but also with the full training dataset available performance is still increased by 1 and 2 points for the accuracy and macro-F_1-score respectively. Comparing to the baseline performance we see that the best-performing versions of the models can use approximately half of the training data while keeping the test accuracy approximately the same. For the macro-F_1-score this is half to two-thirds of the training data.

	r_L = **10%**			r_L = **20%**			r_L = **50%**			r_L = **100%**		
Model	r_U	**Acc**	**mF$_1$**	r_U	**Acc**	**mF$_1$**	r_U	**Acc**	**mF$_1$**	r_U	**Acc**	**mF$_1$**
Baseline		0.707	0.554		0.758	0.614		0.834	0.727		0.869	0.770
$GAN_{h=1}$	**8.0**	**0.789**	0.642	4.0	0.815	0.698	1.0	0.854	0.750	2.0	0.881	0.784
$GAN_{h=3}$	**8.0**	**0.789**	**0.655**	8.0	0.814	0.702	**1.0**	**0.867**	**0.759**	**0.0**	**0.882**	**0.791**
$WGAN_{h=1}$	16.0	0.757	0.622	16.0	0.819	0.696	4.0	0.847	0.738	1.0	0.873	0.778
$WGAN_{h=3}$	16.0	0.775	0.642	**8.0**	**0.828**	**0.710**	4.0	0.857	0.753	1.0	0.875	0.778

Table 5.2: Overall best performing values of r_U per model.

Similar to the GANs without unlabelled data, we see that the GANs with three hidden layers

perform better than their one hidden layer variants. With the addition of unlabelled data this now also holds for when very little amounts of labelled data are available. Probably, this is the result of the discriminator now also learning more complex patterns in BERT's CLS vectors at lower amounts of labelled data, considering that the total amount of data has grown due to the unlabelled data.

Table 5.2 shows the exact performance values and the ratios of unlabelled data r_U for which they were achieved. It shows that the optimal unlabelled data ratio r_U varies, but we can say that the ideal ratio of r_U decreases as the amount of labelled data r_L increases. For sufficient amounts of labelled data, even using no unlabelled data at all might be optimal as $\text{GAN}_{h=3}$ shows for $r_L = 100\%$. This suggests that given sufficient labelled data, the presence of unlabelled data negatively impacts the training of the model. A reason for this could be that the discriminator focuses too much on the discrimination part, instead of the classification part of its task. The WGANs seem to allow for higher ratios of unlabelled data compared to the regular GANs, potentially needing the additional data to produce artificial samples that can confuse the discriminator.

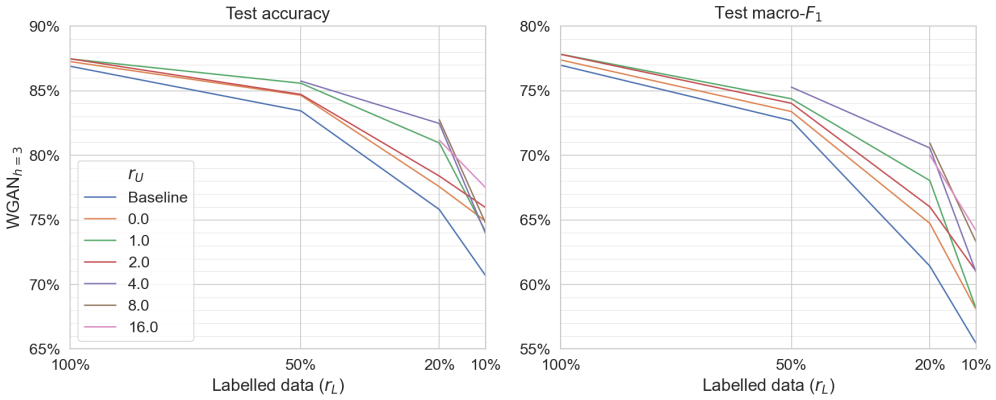

Figure 5.4: Overall performance for $WGAN_{h=3}$ with different ratios of unlabelled data r_U.

The overall performance for the WGAN with three hidden layers, shown in figure 5.4, gives a similar impression. From 100% of labelled data available to 10%, the optimal unlabelled ratio r_U goes from 16 to 8 and then from 4 to 1. With the full dataset, the performances converge to a low value of r_U, something we see for all tested models. The exact performance values and the graphs for all our GAN variants can be found in the Appendix.

5.2 Sub-task performance

Our model combines two sub-tasks, namely detecting whether a category is mentioned and if so, what sentiment is expressed towards this category. In various other works, such as our baseline method, these tasks are trained and tested separately. To verify our re-implementation of the baseline we tested it on the SemEval-2014 dataset, for which the results are added in the Appendix. We see that our re-implementation closely matches the 91.5% detection macro-F_1-score and the 85.9% classification accuracy as reported by Sun et al [67]. Our results are slightly lower with scores of 90.4% and 84.8% respectively, which could be expected as we train for both sub-tasks at once instead of using separately trained models for both sub-tasks. For MAMS, the performance of the baseline model on the two sub-tasks can be seen in figure 5.5, which shows a clear difference in detection and classification performance.

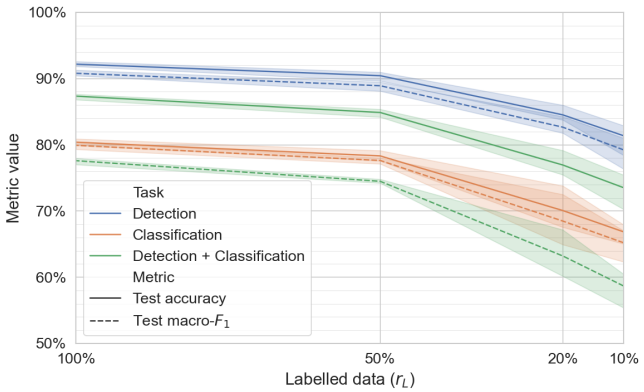

Figure 5.5: Baseline sub-task performance.

Remarkably, the accuracy and macro-F_1-score only differ by up to one point for both sub-tasks, while this difference is very significant for the combined task. To explain this, we take a look at how we split the combined task into two sub-tasks. By using softmax our model predicts the probability of each class. We consider a category not detected in the detection sub-task if the probability of *Not Mentioned (NM)* is larger than 50%, while NM just having the highest probability suffices in the combined task. As the highest probability could be lower than 50%, categories that are detected in the sub-task might not be in the combined task. Consequently, the combined prediction can be wrong while the predictions for the individual sub-tasks are correct. Considering that NM is the most-prevalent class the impact on accuracy is limited, but the impact on the macro-F_1-score can be very significant.

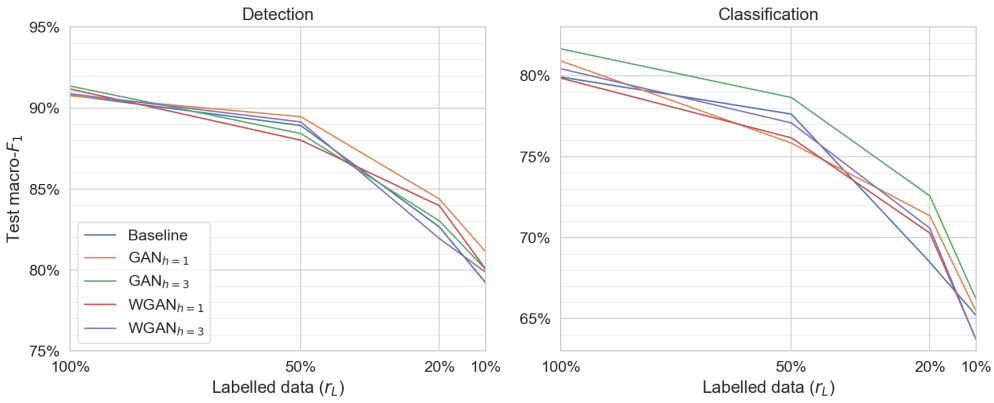

Figure 5.6: Sub-task performance without using unlabelled data.

Looking at the performance of our GANs with only labelled data in figure 5.6, we can see that purely adding a GAN does not necessarily improve sub-task performance, while this is the case for the overall performance. GAN variants with one hidden layer can make a significant difference in detection for lower ratios of labelled data r_L, while in most other cases the difference with the baseline is very limited. The relative simplicity of the detection task may explain why the one hidden layer variants prevail, as a more complex generator might not help improve performance.

For classification the regular GAN with three hidden layers makes a significant difference, outperforming the baseline by 1 to 4 points for all values of r_L. The direct access to the discriminator weights and the more complex classification task probably gives the $\text{GAN}_{h=3}$ model an edge, while other models struggle to improve the baseline. The latter is particularly the case when more labelled data is already available.

Although we have seen that our models struggle to improve the baseline performance with only labelled data, this is not the case with unlabelled data. All models can make significant improvements with the optimal amount of unlabelled data, improving performance by up to 5 and 6.5 points for detection and classification respectively. Similar to the overall performance the graph shows that the differences in performance converge, making the most impact at lower ratios of labelled data r_L. Figure 5.7 shows a visualisation of these best-performing results, with the ratio of unlabelled data r_U selected based on the macro-F_1-score. The exact results together with the best-performing r_U values can be seen in table 5.3 for detection and table 5.4 for classification.

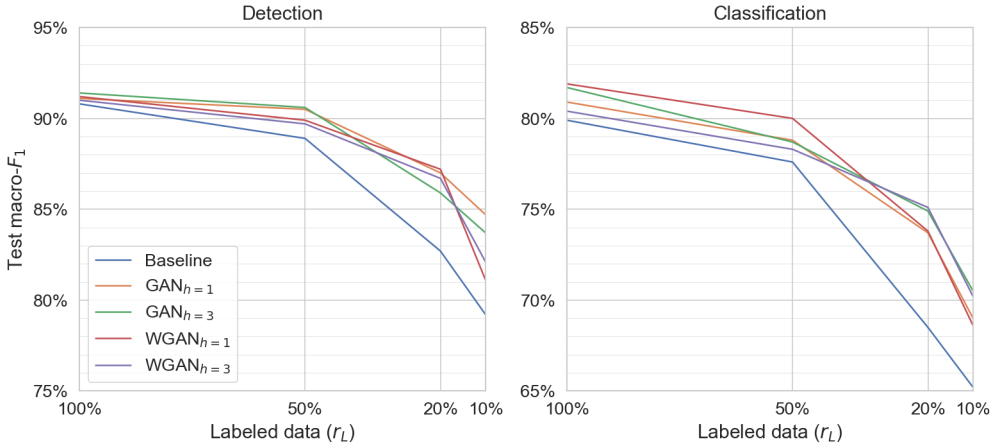

Figure 5.7: Sub-task performance for the best-performing values of r_U.

For detection, all models can improve performance, but the differences are rather limited for $r_L \geq 50\%$. For these values of r_L, all GANs perform within 1 point of each other with regular GANs performing slightly better than their WGAN counterparts. For smaller amounts of labelled data, the GAN with only one hidden layer performs better than its three hidden layer variants. The simpler structure of the one hidden layer GANs might allow them to more quickly learn the relatively simple task of detection, particularly with small amounts of labelled data.

	$r_L = \mathbf{10\%}$			$r_L = \mathbf{20\%}$			$r_L = \mathbf{50\%}$			$r_L = \mathbf{100\%}$		
Model	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$
Baseline		0.814	0.792		0.845	0.827		0.904	0.889		0.922	0.908
$\text{GAN}_{h=1}$	**8.0**	**0.869**	**0.847**	4.0	0.887	0.870	**2.0**	**0.920**	0.905	1.0	0.924	0.911
$\text{GAN}_{h=3}$	8.0	0.857	0.837	1.0	0.876	0.859	**1.0**	**0.920**	**0.906**	**0.0**	**0.927**	**0.914**
$\text{WGAN}_{h=1}$	8.0	0.828	0.811	**16.0**	**0.892**	**0.872**	4.0	0.914	0.899	0.0	0.926	0.912
$\text{WGAN}_{h=3}$	16.0	0.841	0.821	4.0	0.884	0.867	1.0	0.912	0.897	1.0	0.924	0.910

Table 5.3: Detection performance for the best-performing value of r_U.

Furthermore, it stands out that WGANs need more unlabelled data for optimal performance,

similar to the overall performance. The WGANs probably need this data to produce the necessary high-quality artificial samples as they cannot attack the discriminator's weights directly. In general, the ideal amount of unlabelled data still decreases as labelled data increases like for the overall performance, but in particular for the WGANs this pattern does not always hold.

The classification performance shows slightly larger increases in performance compared to detection and shows a large variation in which model performs best. For low amounts of labelled data, the models with three hidden layers closely compete for the best performance. For higher amounts of labelled data, the Wasserstein GAN with one hidden layer performs better, successfully managing to use more unlabelled data compared to the other models. The ideal ratio of unlabelled data still increases as the number of labelled data decreases. For the GAN with three hidden layers adding unlabelled data even has a negative effect for $r_L \geq 50\%$.

	$r_L = \mathbf{10\%}$			$r_L = \mathbf{20\%}$			$r_L = \mathbf{50\%}$			$r_L = \mathbf{100\%}$		
Model	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$	r_U	**Acc**	$\mathbf{mF_1}$
Baseline		0.669	0.652		0.701	0.685		0.783	0.776		0.804	0.799
$\text{GAN}_{h=1}$	16.0	0.708	0.690	8.0	0.745	0.737	1.0	0.793	0.788	0.0	0.814	0.809
$\text{GAN}_{h=3}$	**8.0**	0.715	**0.705**	8.0	0.759	0.749	0.0	0.792	0.787	0.0	0.821	0.817
$\text{WGAN}_{h=1}$	4.0	0.705	0.686	2.0	0.749	0.738	**4.0**	**0.805**	**0.800**	**1.0**	**0.823**	**0.819**
$\text{WGAN}_{h=3}$	**8.0**	**0.716**	0.702	**8.0**	**0.760**	**0.751**	4.0	0.792	0.783	0.0	0.808	0.804

Table 5.4: Classification performance for the best-performing value of r_U.

While regular GANs generally performed better for detection, the WGANs do so for classification. As a more complex and nuanced task, it has more varying samples, which probably benefits the GANs using the Wasserstein distance. An interesting result is that the WGANs with three hidden layers perform better for lower amounts of labelled data, while their one hidden-layer variants perform better for more labelled data. A potential explanation for this could be that $\text{WGAN}_{h=3}$ performs better with more unlabelled data in a classification setting, an advantage when less labelled data is available.

There is also an important thing to note comparing the overall results to these sub-task results. We see The GANs with three hidden layers perform best on the combined task while their one hidden layer variants are generally better for the individual sub-tasks. This is very counter-intuitive and a result of the way we split the sub-tasks as explained earlier in this section. Mathematically, this can only be the case if the models with $h = 3$ correctly classify sentiment polarities with a higher probability, making sure this probability is higher than that of *Not Mentioned*. Altogether, it should be noted that the models have been trained on the combined tasks and not specifically on the individual sub-tasks. Therefore, these results should always be viewed in the context of the combined task.

For both sub-tasks, the results for all configurations and additional graphs can be found in the Appendix.

5.3 Error analysis

In this section we analyze the errors that our method makes, focusing on our overall best-performing methods as presented in table 5.3 and comparing them with the baseline. First, we quantitatively discuss the errors, looking into class confusion and performance on a category level. After this we use a qualitative approach, manually analysing errors to see how our method improves the performance and where the method could still improve.

5.3.1 Quantitative error analysis

We start this quantitative analysis by analysing the predicted classes, focusing on which classes are confused with each other. Figure 5.8 shows the confusion matrices for the baseline method based on the test dataset, where the values are summed over 3 experiments.

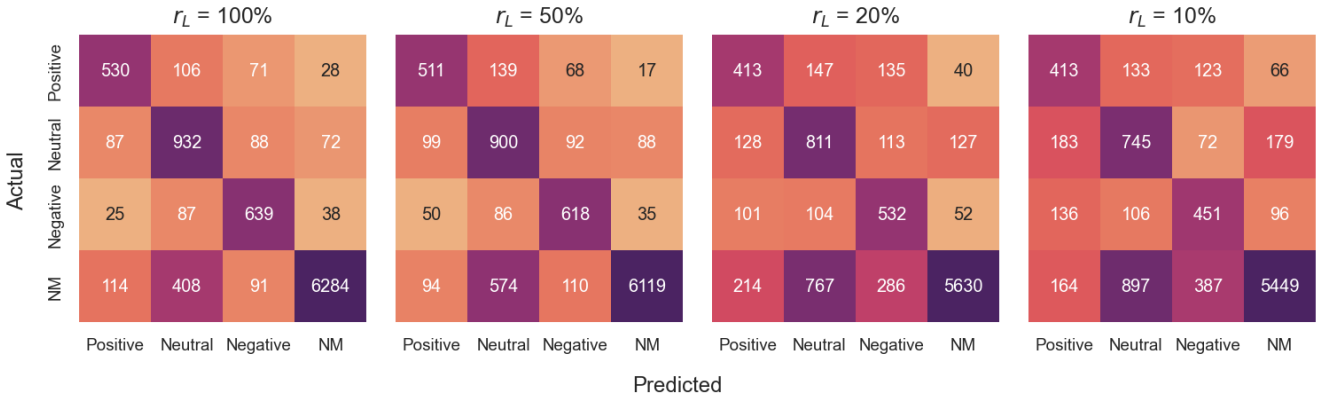

Figure 5.8: Baseline confusion matrices.

It becomes clear that both precision and recall drop significantly for all classes as the ratio of labelled data r_L decreases. The baseline model finds it increasingly difficult to correctly classify sentences in which the category is not mentioned, with the recall for not mentioned (NM) dropping from 91.1% to 79.0%. Considering the large number of sentences in which a category is not mentioned, this has a very significant effect on the precision of the other classes. This particularly holds for the neutral class, for which the precision drops from 60.7% to 39.4%, making it most likely that a predicted neutral is actually not mentioned for $r_L = 10\%$. The confusion between neutral and not mentioned seems logical, as usually not much sentiment is expressed in both cases.

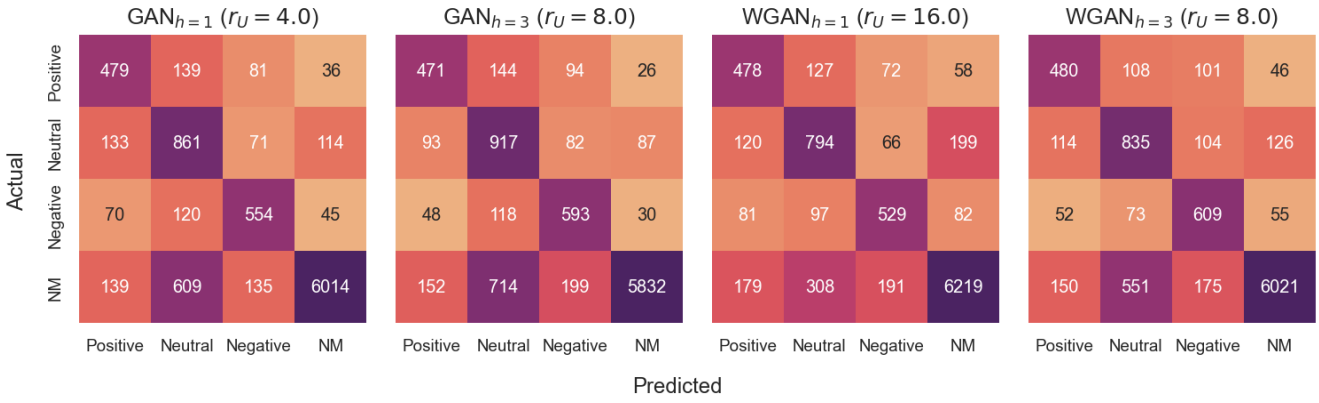

Figure 5.9: Confusion matrices for best-performing models for $r_L = 20\%$.

When we look at the best-performing models for $r_L = 20\%$ shown in figure 5.9, we can see significant differences between models. All the shown models improve the macro-F_1-score by at

least 8 points, where WGAN$_{h=3}$ has the best macro-F$_1$-score but is closely followed by the other models. Compared to the baseline we can see that the confusion between positive and negative has greatly reduced for all models, which also holds for the confusion of positive and negative with neutral. The main challenge for all models is the confusion between neutral and the not mentioned class. Both the one hidden layer GANs and the WGANs seem best able to improve the precision on neutral compared to their counterparts, but do so at the cost of its recall. For the WGANs this could be the result of the higher variance in the generated samples, spreading the confusion over both the neutral and not mentioned recall.

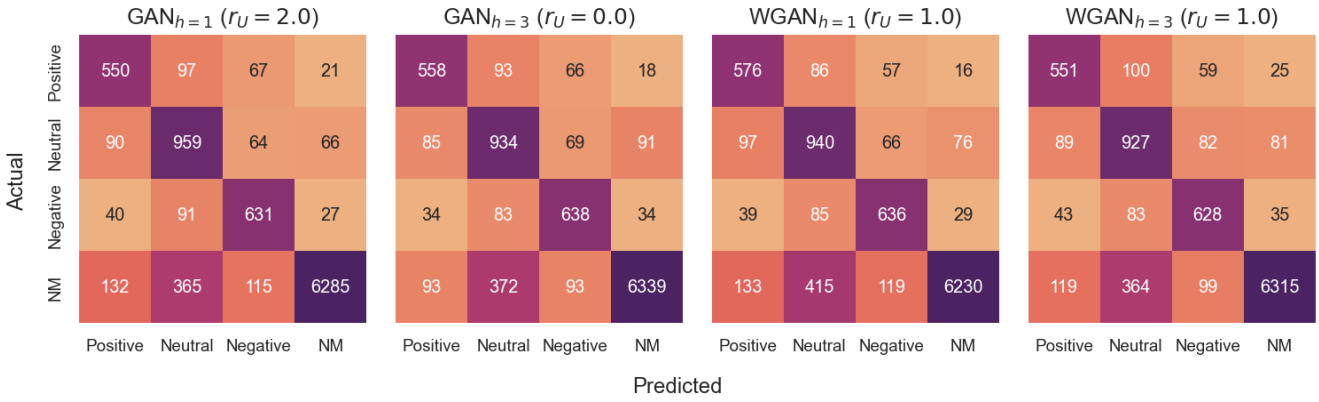

Figure 5.10: Confusion matrices for best-performing models for $r_L = 100\%$.

Looking at the same visualisation for the full training dataset, shown in figure 5.10, we can see both differences and similarities. A major difference is that our GANs improve significantly less for $r_L = 100\%$, with GAN$_{h=3}$ on average improving most with 1.3 points in terms of macro-F$_1$-score. All models still manage to improve the positive recall and most also do this for neutral. The major difference, however, is again made in the sentences in which a category is not mentioned. GAN$_{h=3}$ most successfully improves recall for not mentioned to 91.9%, partially also by its ability to better reduce the confusion of positive and negative with the not mentioned class.

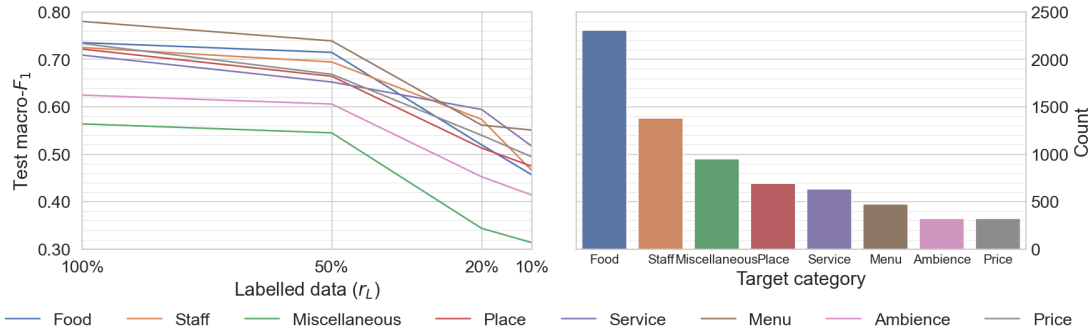

Figure 5.11: Baseline performance (left) and training dataset prevalence (right) for the different categories.

Similar to the classes we also analyze the errors with respect to the 8 categories that are present

in the dataset. Figure 5.11 shows the macro-F_1-score of the baseline model broken down for the different categories, together with the prevalence of those categories in the training dataset. We can see that the model has large differences between categories. In particular, the baseline model performs significantly worse for the *miscellaneous* and *ambience* categories. Both these categories have difficulties in classification, but for the miscellaneous category, this is mainly a result of poor detection. This can be expected as expressions belonging to this category can highly vary, but is also somewhat surprising considering how prevalent this category is.

As for the overall performance, our GANs can significantly improve the performance for the categories but also show significant differences. Figure 5.12 shows the difference between the baseline and the best-performing models, where a positive value denotes that a model performs better than the baseline. It stands out that the category *menu* generally gains the least from the addition of a GAN, as this category already performs very well in the baseline method.

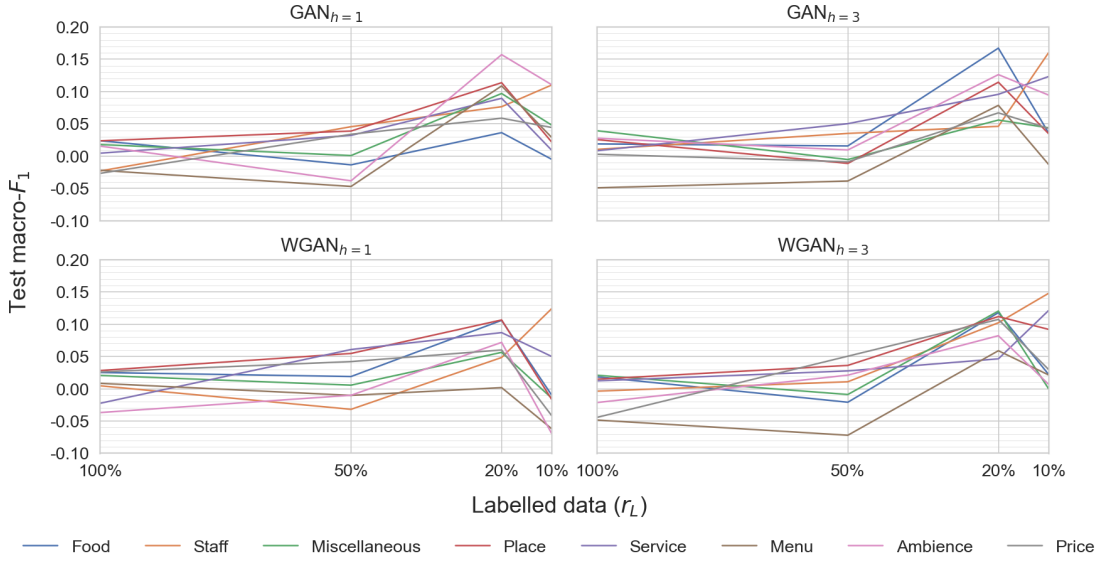

Figure 5.12: Difference in performance between baseline and best-performing models for the different categories.

The models realize the largest increase in performance for a ratio of labelled data $r_L = 20\%$, but there seems to be no clear pattern as to which categories are improved by how much. What does stand out is that the *staff* and *service* categories are the only ones that have a larger improvement for $r_L = 10\%$ than for $r_L = 20\%$. Furthermore, the ambience seems to benefit more from the regular GANs while miscellaneous seem to profit more from the Wasserstein GANs. The WGAN's generally higher variance in generated samples might be an advantage for the miscellaneous category, as this category likely has a high variance of itself as well.

5.3.2 Qualitative error analysis

In this qualitative error analysis, we manually investigate incorrect predictions that our models make. Considering that this task is highly time-consuming, we focus on comparing the baseline and the best-performing models for the ratios of labelled data r_L of 20% and 100%, respectively WGAN$_{h=3}$ and GAN$_{h=3}$. For each model, we gather the incorrect predictions on the test dataset

over three different experiments, which are all trained and evaluated separately. We then randomly sample 100 incorrect predictions for manual investigation, without duplicates and using how many experiments an error occurs as the sampling weight.

Label	Sentence
Neutral	*The food is fantastic*, Gourmet comfort food and has gotten progressively better over the past year, as did the service.
Not mentioned	*The food is average, at best* (I have been repeatedly underwhelmed by the mediocre fare) and the service is only good if Elaine is within range.
Not mentioned	The overpriced *food* that's supposed to come off as homestyle diner just doesn't work when the atmosphere is supposed to be super cool but *the food is super bland*.

Table 5.5: Examples of actual labels for the category food that are at least debatable.

Looking at the incorrect predictions in general, there are a few observations that stand out for all models and values of r_L. First, the MAMS dataset seems to contain various test cases in which the supplied label is at least debatable, if not incorrect. For the models at $r_L = 100\%$, this can amount to over 30% of the sampled incorrect predictions. Table 5.5 shows three examples of such cases for the category food.

Having the lowest macro-F_1-score of all target categories, the miscellaneous category also significantly contributes to the errors. Its detection seems the main difficulty for the models, as confusion between the neutral and not mentioned label for miscellaneous makes up for up to 25% of all sampled errors. Similarly to the models, we find it very difficult to assess the presence of this category manually. Table 5.6 shows an example of two sentences that both tell an anecdote related to staff or service, yet only is labelled to contain the miscellaneous category[1].

Sentence
The service, which was horrendous - they charged us for two appetizers when we asked to split one (when we brought it up, he insisted that the appetizer really did only consist of 2 sliced tomatoes and 1 slice of cheese).
To top it off, when we mentioned it to our lovely waitress, she responded as though it was no big deal along with Well now you know for the next time.

Table 5.6: Examples for the miscellaneous category: one is labelled as neutral and one as not mentioned.

Next to some debatable labels and the difficult miscellaneous category, however, there are also some distinct types of errors that we see. A good example of this is the confusion between the neutral and not mentioned label outside of the miscellaneous class. For the baseline at $r_L = 100\%$, a text in which a category is not mentioned is often predicted to be neutral if the category is mentioned while nothing is said about it. For example, words like *kitchen* and *table* seem to trigger detection of the category *place*, while *served* seems to trigger service and *wine room* is sometimes recognised as talking about *food*. At $r_L = 20\%$ a clear difference is visible, now also frequently detecting a category while there is nothing in the text that is obviously related to it.

[1] The bottom sentence is labelled to be neutral.

This is particularly the case for the category *food* when little labelled data is available, as it seems that uncommon words are sometimes enough to 'detect' food. This could be due to dishes and recipes usually having a name that is not necessarily a common English word. As a result, the BERT preprocessor uses the unknown word token and the model might associate these tokens with food. When more data is present the models seem better at handling this, possibly as a result of a better understanding of the context around these uncommon words.

Another matter that can be observed for lower amounts of labelled data is that the model seems to mix up certain categories with each other. *Food* and *price* are sometimes difficult to keep separated as, for example, *overpriced food* could also mean that the food is of insufficient quality. Furthermore, this also happens for *staff* and *service*, as staff members usually provide the service of an establishment. Last, the *place* and *ambience* can be very related as well as a good place can usually be expected to have a good ambience and vice-versa. Table 5.7 shows two example sentences in which two categories seem to get mixed up.

Detected	**Sentence**
Service	Our last experience: Waiting for a table at the bar (we always make reservations), the bartender ignored us until my husband intervened with one of the owners.
Place	Small dishes, a bit pricier than you'd pay in Miami or LA, but the atomsphere is on the sexy side (however pared down) and its cozy.

Table 5.7: Examples of the wrong category being detected.

Applying our best-performing GANs reduces most of the aforementioned problems, but some remain a significant challenge. This in particular holds for the detection of food while no related words are available, as our GANs do not seem to be able to improve here. For categories outside of food, our method can make a real improvement, but the challenges here remain significant. For reducing mixed-up categories, however, our GAN shows to work well. Additionally, our methods seem to better handle sentences that have a negative undertone, which the baseline frequently marks to be negative if given only 20% of the labelled data.

For $r_L = 100\%$, no clear differences could be found between the errors of the baseline and best-performing models in terms of qualitative analysis. The miscellaneous class and the detection of categories that are mentioned but about which nothing is said therefore remain the main challenge for the methods trained with a full dataset.

5.4 Additional analysis

In addition to the results shown before, we have also generated t-distributed Stochastic Neighbour Embeddings (t-SNE) to represent the output sequence (CLS) vectors that BERT produces in a two-dimensional space. We produced these embeddings aiming to gain more insights into the effect of applying regular and Wasserstein GANs, by visualising if artificial and real CLS vectors are easily separable by t-SNE. We produce these embeddings on a balanced sample of the training dataset with 500 samples per class, as otherwise certain classes would not be visible.

Before applying t-SNE, we first perform Principal Component Analysis (PCA) on the CLS vector to reduce the number of input components from 768 to 48. In all our experiments these 48 components explain almost all of the variance in the original CLS vectors. We then apply t-SNE analysis to reduce the number of components to 2, using 1000 iterations and a perplexity of 30.

Starting without using any unlabelled data, an interesting phenomenon becomes visible when we compare the t-SNE plots of the regular and Wasserstein GAN for $h = 3$, as shown in figure 5.13. The real data classes are more clustered and separated from each other at the regular GAN for both values of r_L and we have seen that the GAN achieves a higher test performance. However, the artificial samples generated by the regular GAN are very clustered as well, despite they should replicate all real classes.

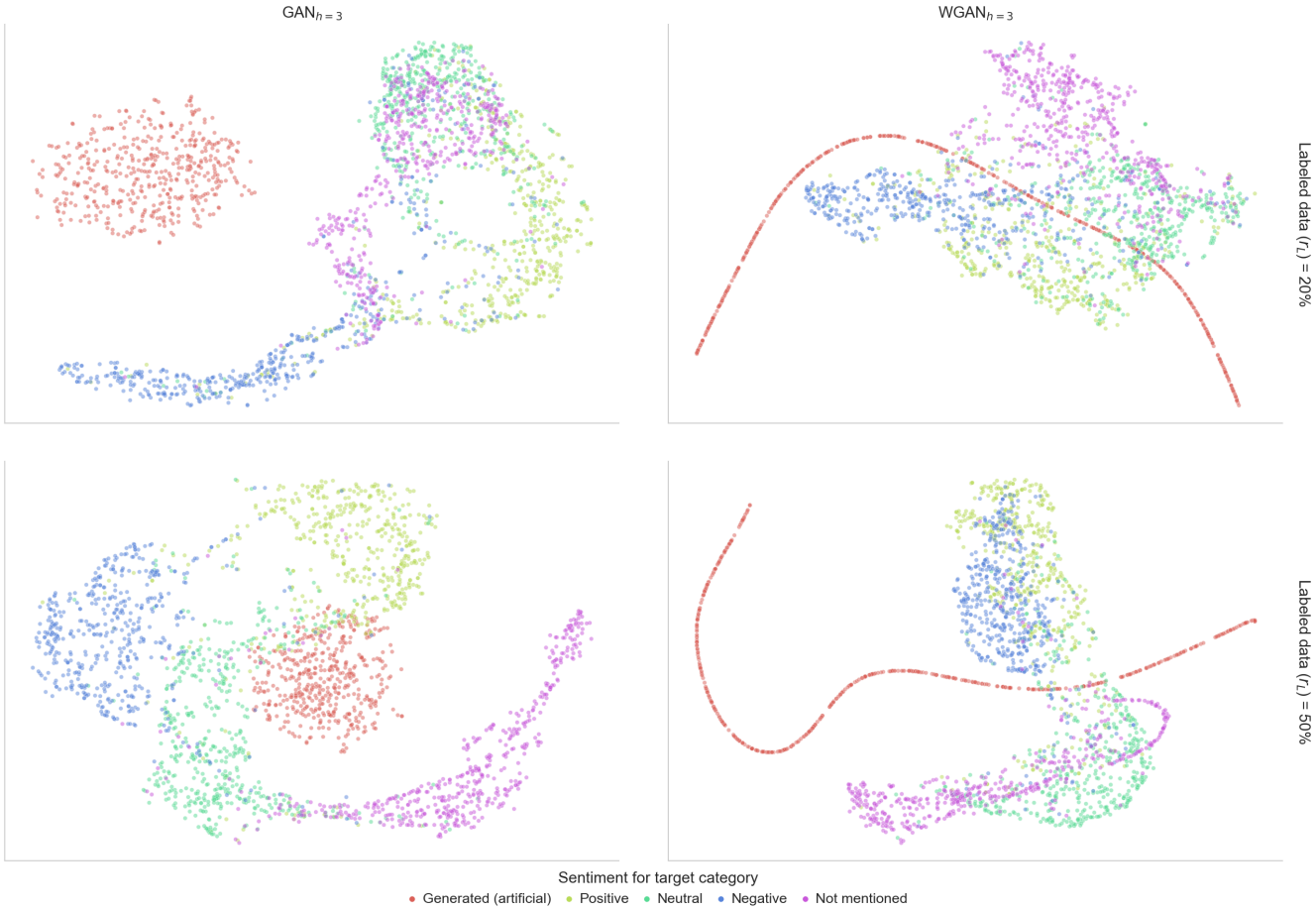

Figure 5.13: Comparison of t-SNE plots for $r_U = 0\%$.

In contrast, the artificial samples created by the Wasserstein GAN generator show a curved line pattern in the t-SNE analysis. This indicates that its samples relate to some of its other samples, but are also very unrelated to some other samples generated by the same model. The higher variation in-between samples that can be observed here can be seen as one of the WGAN's well-known strong suits: the ability to create highly diverse samples. Comparing the samples with the regular GAN, it seems as if the regular GAN suffers from mode collapse despite the special part of the generator loss aiming to minimize the distance between features L_G^{feat} as defined in equation 4.2.

When adding unlabelled data, again some interesting phenomena are showing for the regular and Wasserstein GAN, as is shown in figure 5.14 for $r_L = 20\%$. The two variants show different

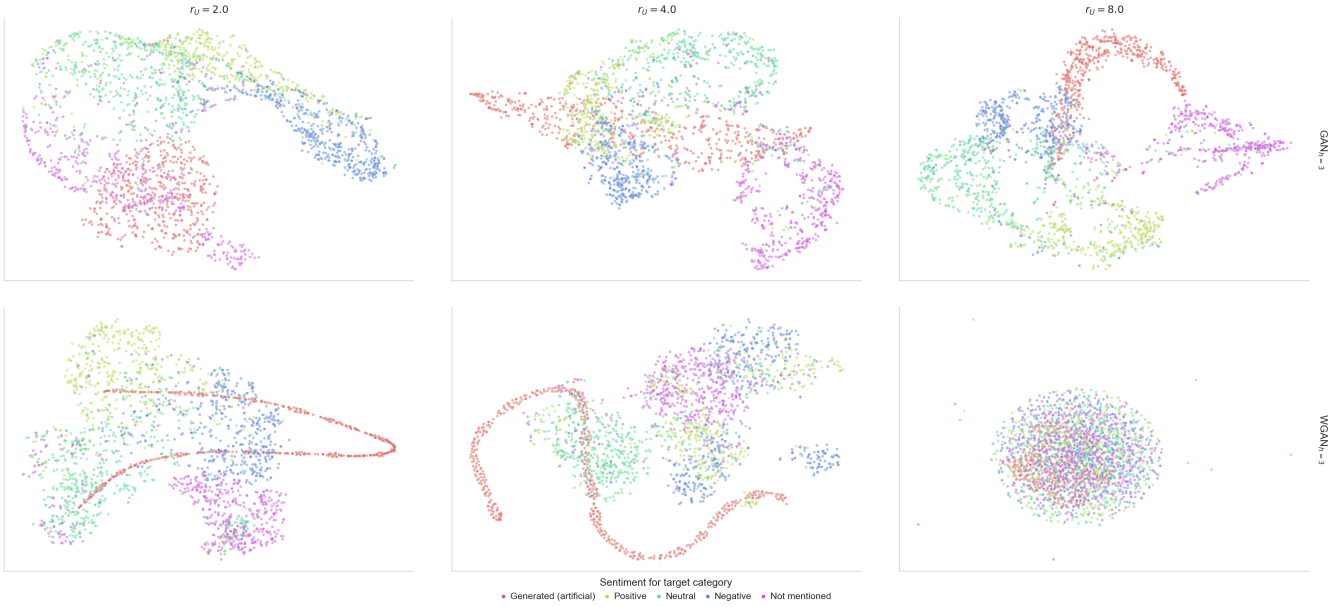

Figure 5.14: Comparison of t-SNE plots for different values of r_U at $r_L = 20\%$.

patterns when it comes to artificially generated data. For the regular GAN, this data as shown in red forms more of a round shape for $r_U = 2.0$, slowly stretching out and curving as the ratio of unlabelled data increases. In the process, the real classes get slowly more separated from each other. For the Wasserstein GAN, the artificial data points form a long and thinly stretched line that expands when r_U increases to 4.0, indicating a larger variance within these artificial samples. For an unlabelled ratio of 8.0, however, all data points start to form one big cluster with all classes more or less mixing. Strangely enough, this is where the WGAN achieves its best performance, potentially as a result of being able to very well confuse the discriminator.

5.5 Summary

- Reducing the amount of labelled training data severely reduces the performance of the baseline Transformer-based Aspect-Based Sentiment Analysis (ABSA) method. With only 20% of the labelled dataset available, the performance can drop by 10-15 points and this decreases further when less data is available. This drop in performance affects both category detection and sentiment classification.

- Using a regular or Wasserstein GAN as described in our method improves performance without using unlabelled data. When trained with a full dataset, this performance improvement is limited to only one or two points. With fewer data, however, this difference can get up to 5 points. Both regular and Wasserstein GANs can improve sub-task performance for lower amounts of available labelled data, but regular GANs generally achieve higher scores. When no unlabelled data is used, the models using a generator with three hidden layers generally perform better than their one hidden-layer counterparts.

- The addition of unlabelled data can very significantly increase performance, increasing accuracy and macro-F_1-score by up to 8 and 10 points respectively when 20% or less of the original dataset is present. This can decrease the need for labelled data by approximately 50% in terms of accuracy, while for the macro-F_1-score this can be decreased by approximately 40-50%. Similar to the GANs trained without unlabelled data, the variants with a generator with three hidden layers achieve the best results. Additionally, the use of unlabelled data can improve both category detection and sentiment classification by up to 5 points in terms of macro-F_1-score.

- The chosen ratio of unlabelled data, however, has a large effect on performance. There is not one ratio that performs best in all circumstances. In general, the optimal ratio of unlabelled data decreases when the amount of labelled data increases and can even be zero. Regular GANs generally perform better with a lower ratio compared to their Wasserstein counterparts. For GANs, a combination of labelled and unlabelled data that together totals between 80% and 180% of the size of the original labelled training dataset seems optimal. For Wasserstein GANs this equals between 1.5 to 2.5 times the size of the original dataset. Adding unlabelled data can significantly improve both category detection and sentiment classification performance.

- Error analysis reveals that the main challenge with less labelled data is the confusion regarding categories that are not mentioned, where in particular neutral sentiment is often predicted. In particular the miscellaneous is difficult, presumably as a result of the high variety within this category. Additionally, some words seem to incorrectly trigger the detection of certain categories. Our best-performing models reduce these errors and generally improve performance for all class labels and aspect categories, but some errors - mainly those regarding the miscellaneous category - remain a significant challenge.

- Additional analysis of the real and generated word embeddings indicates that the Wasserstein GAN produces artificial word embeddings with a higher variance, with generators with three hidden layers producing the largest variance. Despite this increase in variance, it does not necessarily lead to an increase in performance when compared to the regular GAN. It is not fully clear why this is the case, but this could be due to the absence of direct feedback from the discriminator weights.

Chapter 6

Discussion

We have investigated the use of semi-supervised Generative Adversarial Networks (GANs) to improve the performance of Transformer-based Aspect-Based Sentiment Analysis (ABSA). We have shown that our method can significantly improve performance, in particular when less labelled training data is available. In this chapter, we will discuss the implications of our research and its results, together with its limitations.

Next to showing that Croce et al.'s GAN-BERT method [14] can be applied to the ABSA domain, our research has various other implications. First, we have shown that the method is generally also beneficial if the unlabelled data is left out. Additionally, our research shows strong indications that, at least for our dataset, the unlabelled ratio of 100 as used by Croce et al. is too high: the optimal ratio of unlabelled data seems to decrease as the amount of labelled data increases. As a result, the ideas of Croce et al. can likely more easily be applied than the original paper suggests, as less unlabelled data seems necessary to make it work. Another significant difference to GAN-BERT is the use of generators with more hidden layers. Our results show that the more complex generators can significantly increase performance, most likely due to their ability to generate more complex structures in the artificial word embeddings. Although regular GANs usually produce better performance, our research has also shown that using a Wasserstein GAN can be beneficial in specific situations.

The Wasserstein GANs can produce artificial word embeddings with more variance compared to the regular GAN, but in those cases performance not necessarily improves. It is not fully clear why this is, but a potential explanation might be that the WGAN does not have access to the gradients of the discriminator and, therefore, cannot perform the "white-box attack" that the GAN can do in its attempt to mislead the discriminator. In essence, the usage of the GAN in this setup could somewhat correlate to the effect that white-box attacks like the Fast Gradient Sign Method [26] have, which is known to be able to improve performance when used in adversarial training. However, the current method makes it difficult to analyze the performance of the regular or Wasserstein GANs as the 'real' word embeddings produced by BERT constantly change. The adaptation of BERT to provide more easily classifiable embeddings most likely improves performance but makes it hard to see progress in the GAN. Additionally, as nearly no research exists on using GANs to generate artificial word embeddings, there is also no indication of which GAN architectures or configurations could work well in this setting.

Looking more specifically at the ABSA task, it would have been better to keep the two sub-tasks of detection and sentiment analysis separated. For this research, combining the two sub-tasks seemed the better choice, as this allows for a more complex task and allows us to train only one model. This reduction of computational costs also has a more impractical side

as the combined training will most likely perform differently for both sub-tasks compared to when trained separately, while most other models are trained separately as well. Likely, the performance on the sub-tasks could have been increased when trained separately. In addition, this would have made it possible to conclude whether certain configurations would work better for one of the sub-tasks. For example, there are indications that Wasserstein GANs seem to have a more positive influence on sentiment classification. To prove this, however, the sub-tasks should be trained separately.

The grid-search way in which the research was approached also somewhat limited the research. This method led to a large number of combinations between model types, amounts of labelled data and amounts of unlabelled data. In total, this resulted in approximately 90 model configurations that have been tested in over 500 experiments. This has resulted in a lot of results and insights but has the downside that sometimes choices had to be made due to the large computational cost of training BERT a large number of times.

Last, the research also showed that for lower amounts of labelled data available, the variance in the results increases. While each experiment has been repeated approximately 5 times to gain a more reliable average performance, the variance made it still hard to compare models where the results are very minimal. In combination with the computational costs described above it was not realistically possible to repeat experiments 10 or 15 times, but maybe other steps could have been taken to reduce this variance. A way of producing more stable results with very low amounts of labelled data would have been beneficial.

Chapter 7

Conclusions

We have seen that less labelled training data significantly decreases the performance of our baseline Transformer-based Aspect-Based Sentiment Analysis (ABSA) method. In this research, we investigate whether applying a Generative Adversarial Network (GAN) can improve this performance:

RQ1 Can applying a semi-supervised GAN to generate artificial word embeddings improve the performance of Transformer-based ABSA, in particular when smaller amounts of annotated data are available?

RQ2 How does the amount and ratio of the labelled and unlabelled data used in a semi-supervised GAN influence the performance of Transformer-based ABSA?

We have performed experiments for various amounts of labelled and unlabelled data, with both a regular and a Wasserstein GAN and with two different generator architectures. Based on the results presented in the previous chapter, we draw the following conclusions:

- With labelled data only, applying a GAN to generate artificial word embeddings improves the performance of our Transformer-based ABSA method in almost all cases. This performance improvement becomes larger as less labelled data is available and can increase accuracy and the macro-F_1-score by up to 5 points.
- Adding unlabelled data can further increase performance significantly by up to 5 additional points. This allows our best-performing models to reach the performance of a baseline model that uses 80-100% more labelled data. The largest performance increase is observed when only 10% or 20% of the labelled data is available.
- There is not one single optimal ratio of unlabelled data and more is not necessarily better. The best-performing regular GANs generally use less unlabelled data compared to their Wasserstein counterparts. A total amount of data, including both labelled and unlabelled data, varying between 80% and 180% of the size of the full dataset seems ideal for regular GANs. For Wasserstein GANs these percentages are between 150% and 250%.
- Despite that Wasserstein GANs seem to produce more varying artificial word embeddings, regular GANs perform better in most cases by a limited difference. However, both regular and Wasserstein GANs using a generator with three hidden layers produce significantly better performance than their one hidden-layer counterparts.

Altogether, we conclude that applying a semi-supervised GAN can significantly improve the performance of a Transformer-based ABSA method when less training data is available, with a semi-supervised GAN being able to increase performance as if almost twice as much training data was available.

7.1 Future work

While the conclusions of this research are promising, it is to the best of our knowledge also the first application of semi-supervised GANs with more than one hidden layer to BERT and the first application of a Wasserstein GAN with BERT in general. Besides this, there has been (nearly) no research on applying GANs in the ABSA domain in general. As a result, various questions still have to be answered in future work.

First, separating the two detection and sentiment classification sub-tasks might be beneficial to the performance of the model. Also, the effect of which unlabelled data is used might severely impact the result: in this case, very similar unlabelled data was available, but this might not always be the case in practice. Additionally, future work could show whether this technique works more universally for other text-based domains where Transformer-based word embedding models make up the state-of-the-art as well.

With regards to the GANs, there are a lot of other configurations that still could be tested. The training and updating procedure could be adapted, as well as the architectures for both the generator and discriminator. For the WGANs, it should also be investigated to what extent they can be used to improve detection accuracy while not using the gradients of the discriminator as part of adversarial training. As mentioned earlier in the discussion, the differences and similarities with adversarial training should be investigated as well.

Last, it would be interesting to see whether a similar method can also be used at other Transformer-based word embedding models and if the same principle could also be applied at different depths of the model. In the same sense, this training method could potentially also yield success on various other deep learning methods (e.g. domains outside natural language), where it could potentially also enable the usage of less training data to achieve still reasonably good performance.

References

[1] Izzat Alsmadi, Nura Aljaafari, Mahmoud Nazzal, Shadan Alhamed, Ahmad H. Sawalmeh, Conrado P. Vizcarra, Abdallah Khreishah, Muhammad Anan, Abdulelah Algosaibi, Mohammed Abdulaziz Al-Naeem, Adel Aldalbahi, and Abdulaziz Al-Humam. Adversarial Machine Learning in Text Processing: A Literature Survey. *IEEE Access*, 10:17043–17077, 2022.

[2] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862 [cs, stat]*, January 2017. arXiv: 1701.04862.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR, July 2017. ISSN: 2640-3498.

[4] Dan Barsever, Sameer Singh, and Emre Neftci. Building a Better Lie Detector with BERT: The Difference Between Truth and Lies. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2020. ISSN: 2161-4407.

[5] Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3, March 1993.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994. Conference Name: IEEE Transactions on Neural Networks.

[7] Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, January 2009.

[8] Gianni Brauwers and Flavius Frasincar. A Survey on Aspect-Based Sentiment Classification. *ACM Computing Surveys*, page 3503044, December 2021.

[9] C. Breazzano, D. Croce, and R. Basili. MT-GAN-BERT: Multi-Task and Generative Adversarial Learning for sustainable Language Processing. In *Proceedings of the 5th Workshop on Natural Language for Artificial Intelligence*, volume 3015, 2021. ISSN: 1613-0073.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot

Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[11] Erik Cambria, Björn Schuller, Yunqing Xia, and Catherine Havasi. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21, March 2013. Conference Name: IEEE Intelligent Systems.

[12] Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-Likelihood Augmented Discrete Generative Adversarial Networks. *arXiv:1702.07983 [cs]*, February 2017. arXiv: 1702.07983.

[13] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial Text Generation via Feature-Mover' s Distance. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[14] Danilo Croce, Giuseppe Castellucci, and Roberto Basili. GAN-BERT: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2119, Online, July 2020. Association for Computational Linguistics.

[15] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[16] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. CoAtNet: Marrying Convolution and Attention for All Data Sizes. In *Advances in Neural Information Processing Systems*, volume 34, pages 3965–3977. Curran Associates, Inc., 2021.

[17] Gustavo H. de Rosa and João P. Papa. A survey on text generation using generative adversarial networks. *Pattern Recognition*, 119:108098, November 2021.

[18] K. Deorukhkar, K. Kadamala, and E. Menezes. Fgtd: Face generation from textual description. *Lecture Notes in Networks and Systems*, 311:547–562, 2022. cited By 0.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[20] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[21] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-Dependent Sentiment Classification With BERT. *IEEE Access*, 7:154290–154299, 2019. Conference Name: IEEE Access.

[22] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, September 1999. ISSN: 0537-9989.

[23] Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised Cipher Cracking Using Discrete GANs. *arXiv:1801.04883 [cs]*, January 2018. arXiv: 1801.04883.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[26] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, March 2015. arXiv:1412.6572 [cs, stat].

[27] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv:1704.00028 [cs, stat]*, December 2017. arXiv: 1704.00028.

[28] Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. Aspect-Based Sentiment Analysis using BERT. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland, September 2019. Linköping University Electronic Press.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[30] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[31] Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. Open-Domain Targeted Sentiment Analysis via Span-Based Extraction and Classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 537–546, Florence, Italy, July 2019. Association for Computational Linguistics.

[32] Abdul Jabbar, Xi Li, and Bourahla Omar. A Survey on Generative Adversarial Networks: Variants, Applications, and Training. *ACM Computing Surveys*, 54(8):1–49, November 2022.

[33] Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. A Challenge Dataset and Effective Models for Aspect-Based Sentiment Analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6280–6285, Hong Kong, China, November 2019. Association for Computational Linguistics.

[34] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up. *arXiv:2102.07074 [cs]*, December 2021. arXiv: 2102.07074.

[35] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Adversarial Training for Aspect-Based Sentiment Analysis with BERT. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8797–8803, January 2021. ISSN: 1051-4651.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[37] Matt J. Kusner and José Miguel Hernández-Lobato. GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv:1611.04051 [cs, stat]*, November 2016. arXiv: 1611.04051.

[38] Naveen Kumar Laskari and Suresh Kumar Sanampudi. Aspect Based Sentiment Analysis Survey. *IOSR Journal of Computer Engineering*, 18(2):24–28, 2016.

[39] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-ting Sun. Adversarial Ranking for Language Generation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[40] Bing Liu. *Sentiment Analysis and Opinion Mining*, volume 5 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers, 2012.

[41] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019. arXiv: 1907.11692.

[43] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[44] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv:1611.00712 [cs, stat]*, March 2017. arXiv: 1611.00712.

[45] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943.

[46] B. Mehlig. Machine learning with neural networks. *arXiv:1901.05639 [cond-mat, stat]*, October 2021. arXiv: 1901.05639.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013. arXiv: 1301.3781.

[48] S. Na, M. Do, K. Yu, and J. Kim. Realistic image generation from text by using bert-based embedding. *Electronics (Switzerland)*, 11(5), 2022. cited By 0.

[49] S. Naveen, M.S.S. Ram Kiran, M. Indupriya, T.V. Manikanta, and P.V. Sudeep. Transformer models for enhancing attngan based text to image generation. *Image and Vision Computing*, 115, 2021. cited By 0.

[50] Weili Nie, Nina Narodytska, and Ankit B Patel. RelGAN: Relational Generative Adversarial Networks for Text Generation, 2019.

[51] Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, January 2008.

[52] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[53] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[54] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, V. Hoste, M. Apidianaki, X. Tannier, N. Loukachevitch, E. Kotelnikov, N. Bel, S.M. Jiménez-Zafra, and G. Eryigit. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, pages 19–30, 2016.

[55] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, 2015.

[56] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.

[57] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training, 2018.

[58] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners, 2019.

[59] Russell Reed and Robert J. MarksII. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. A Bradford Book, Cambridge, MA, USA, February 1999.

[60] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. Place: US Publisher: American Psychological Association.

[61] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. Number: 6088 Publisher: Nature Publishing Group.

[62] R. M. S. T. Rachev. *Duality theorems for Kantorovich-Rubinstein and Wasserstein functionals*. Instytut Matematyczny Polskiej Akademi Nauk, 1990.

[63] Yunus Saatchi and Andrew Gordon Wilson. Bayesian GAN. *arXiv:1705.09558 [cs, stat]*, November 2017. arXiv: 1705.09558.

[64] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[65] Yu Shang, Xiaohui Su, Zhifeng Xiao, and Zidong Chen. Campus Sentiment Analysis with GAN-based Data Augmentation. In *2021 13th International Conference on Advanced Infocomm Technology (ICAIT)*, pages 209–214, October 2021. ISSN: 2770-1603.

[66] Jinsong Su, Jialong Tang, Hui Jiang, Ziyao Lu, Yubin Ge, Linfeng Song, Deyi Xiong, Le Sun, and Jiebo Luo. Enhanced aspect-based sentiment analysis models with progressive self-supervised attention learning. *Artificial Intelligence*, 296:103477, July 2021.

[67] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 380–385, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[69] Hu Xu, Bing Liu, Lei Shu, and Philip Yu. BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[70] Qiannan Xu, Li Zhu, Tao Dai, and Chengbing Yan. Aspect-based sentiment classification with multi-attention network. *Neurocomputing*, 388:135–143, May 2020.

[71] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[72] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2852–2858, San Francisco, California, USA, February 2017. AAAI Press.

[73] B. Zeng, H. Yang, R. Xu, W. Zhou, and X. Han. LCF: A Local context focus mechanism for aspect-based sentiment classification. *Applied Sciences (Switzerland)*, 9(16), 2019.

[74] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 4006–4015, Sydney, NSW, Australia, August 2017. JMLR.org.

Appendix

A Overall Performance

This section of the appendix contains additional results related to section 5.1, the overall results. Table A.1 contains the average overall performance for each configuration of the model, labelled data ratio r_L and unlabelled data ratio r_U. These results are averages over at least 3 runs.

		r_L = **10%**		r_L = **20%**		r_L = **50%**		r_L = **100%**	
Model	r_U	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**
Baseline		**0.707**	**0.554**	**0.758**	**0.614**	**0.834**	**0.727**	**0.869**	**0.770**
GAN$_{h=1}$	0.0	0.758	0.607	0.793	0.658	0.846	0.736	0.878	0.782
	1.0	0.750	0.601	0.800	0.678	0.854	**0.750**	0.878	0.782
	2.0	0.743	0.601	**0.819**	0.697	0.844	0.737	**0.881**	**0.784**
	4.0	0.769	0.635	0.815	**0.698**	**0.856**	0.747		
	8.0	**0.789**	**0.642**	**0.819**	0.695				
	16.0	0.755	0.613	0.821	0.693				
GAN$_{h=3}$	0.0	0.746	0.603	0.784	0.666	0.847	0.748	**0.882**	**0.791**
	1.0	0.762	0.619	0.813	0.693	**0.867**	**0.759**	0.878	0.785
	2.0	0.747	0.614	0.809	0.693	0.834	0.731	0.869	0.769
	4.0	0.765	0.638	**0.819**	0.692	0.864	0.758		
	8.0	**0.789**	**0.655**	0.814	**0.702**				
	16.0	0.778	0.644	0.812	0.682				
WGAN$_{h=1}$	0.0	0.729	0.576	0.782	0.641	0.823	0.715	0.864	0.761
	1.0	0.726	0.572	0.780	0.654	0.837	0.729	**0.873**	**0.778**
	2.0	0.753	0.606	0.787	0.663	**0.847**	0.734	0.869	0.771
	4.0	**0.757**	0.607	0.783	0.662	**0.847**	**0.738**		
	8.0	0.756	0.616	0.803	0.679				
	16.0	**0.757**	**0.622**	**0.819**	**0.696**				
WGAN$_{h=3}$	0.0	0.749	0.581	0.776	0.647	0.847	0.734	0.873	0.774
	1.0	0.741	0.581	0.810	0.680	0.856	0.744	**0.875**	**0.778**
	2.0	0.760	0.610	0.784	0.660	0.847	0.740	**0.875**	**0.778**
	4.0	0.740	0.610	0.825	0.706	**0.857**	**0.753**		
	8.0	0.748	0.633	**0.828**	**0.710**				
	16.0	**0.775**	**0.642**	0.812	0.700				

Table A.1: Overall performance. Best performing results per combination of model and labelled data ratio r_L are marked in bold.

Next, figures A.1 to A.4 visualise these results per model. Plots of the baseline performance, all models with unlabelled data and the best-performing models can be found in section 5.1.

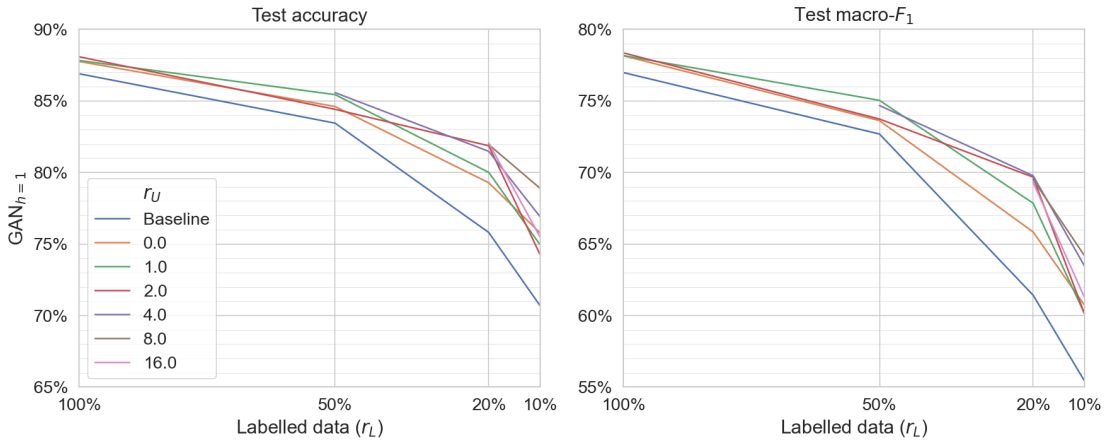

Figure A.1: Overall performance for $\text{GAN}_{h=1}$.

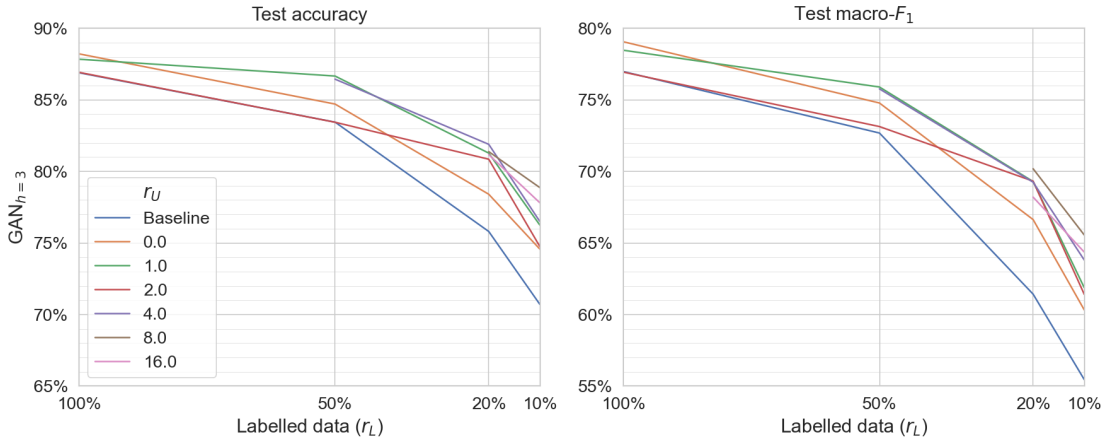

Figure A.2: Overall performance for $\text{GAN}_{h=3}$.

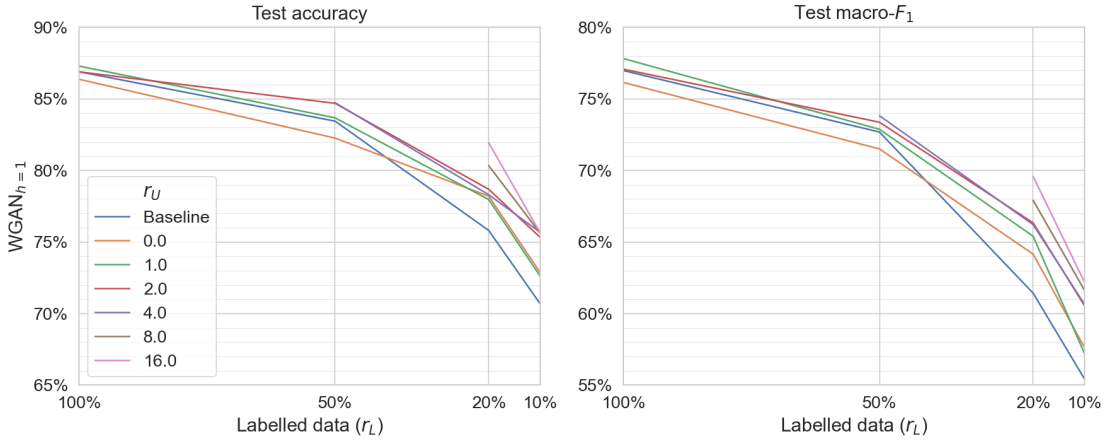

Figure A.3: Overall performance for WGAN$_{h=1}$.

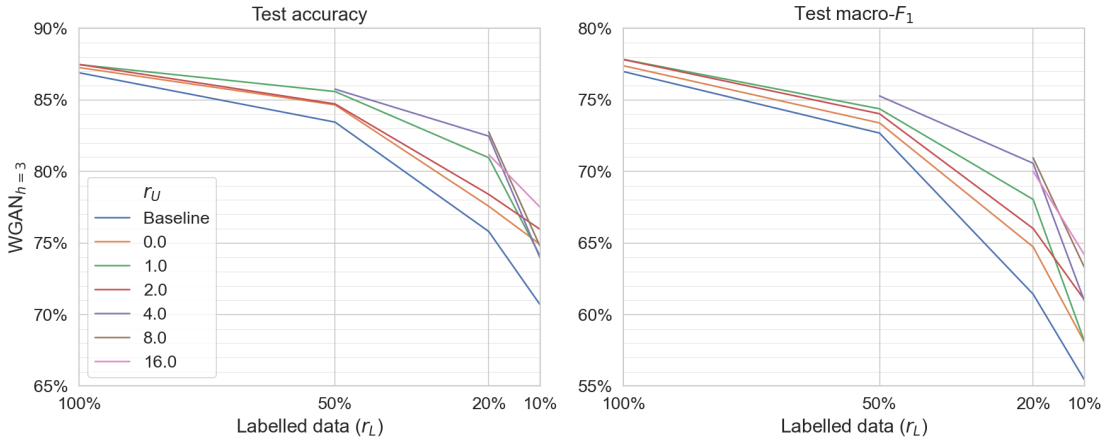

Figure A.4: Overall performance for WGAN$_{h=3}$.

B Sub-task performance

This section of the appendix contains additional results related to section 5.2, the sub-task results. Table B.1 contains the average detection performance for each configuration of the model, labelled data ratio r_L and unlabelled data ratio r_U. These results are averages over at least 3 runs.

		r_L = **10%**		r_L = **20%**		r_L = **50%**		r_L = **100%**	
Model	r_U	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**
Baseline		**0.814**	**0.792**	**0.845**	**0.827**	**0.904**	**0.889**	**0.922**	**0.908**
GAN$_{h=1}$	0.0	0.831	0.811	0.863	0.844	0.910	0.895	0.921	0.908
	1.0	0.823	0.805	0.869	0.851	0.906	0.891	**0.924**	**0.911**
	2.0	0.834	0.814	0.884	0.864	**0.920**	**0.905**	0.923	0.910
	4.0	0.855	0.836	**0.887**	**0.870**	0.916	0.901		
	8.0	**0.869**	**0.847**	0.873	0.855				
	16.0	0.835	0.815	0.878	0.856				
GAN$_{h=3}$	0.0	0.821	0.801	0.848	0.830	0.900	0.884	**0.927**	**0.914**
	1.0	0.836	0.817	0.876	0.859	**0.920**	**0.906**	0.922	0.909
	2.0	0.826	0.808	0.873	0.856	0.891	0.876	0.918	0.904
	4.0	0.841	0.824	**0.878**	**0.858**	0.914	0.900		
	8.0	**0.857**	**0.837**	0.872	0.856				
	16.0	0.844	0.824	0.869	0.850				
WGAN$_{h=1}$	0.0	0.819	0.801	0.860	0.840	0.896	0.880	**0.926**	**0.912**
	1.0	0.819	0.798	0.870	0.852	0.887	0.871	0.917	0.904
	2.0	0.814	0.794	0.856	0.837	0.903	0.887	0.925	0.911
	4.0	0.822	0.805	0.855	0.838	**0.914**	**0.899**		
	8.0	**0.828**	**0.811**	0.874	0.856				
	16.0	0.818	0.801	**0.892**	**0.872**				
WGAN$_{h=3}$	0.0	0.825	0.799	0.838	0.819	0.908	0.891	0.923	0.909
	1.0	0.806	0.785	0.867	0.847	**0.912**	**0.897**	**0.924**	**0.910**
	2.0	0.826	0.805	0.850	0.833	0.903	0.888	0.921	0.907
	4.0	0.813	0.796	**0.884**	**0.867**	0.908	0.892		
	8.0	0.815	0.798	**0.884**	**0.867**				
	16.0	**0.841**	**0.821**	0.872	0.856				

Table B.1: Detection performance. Best performing results per combination of model and labelled data ratio r_L are marked in bold.

Similarly, table B.2 contains the average classification performance for each configuration of the model, labelled data ratio r_L and unlabelled data ratio r_U. These results are also averages over at least 3 runs.

		r_L = **10%**		r_L = **20%**		r_L = **50%**		r_L = **100%**	
Model	r_U	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**	**Acc**	**mF$_1$**
Baseline		**0.669**	**0.652**	**0.701**	**0.685**	**0.783**	**0.776**	**0.804**	**0.799**
GAN$_{h=1}$	0.0	0.676	0.655	0.728	0.713	0.770	0.758	**0.814**	**0.809**
	1.0	0.676	0.657	0.748	0.735	0.793	**0.788**	0.812	0.808
	2.0	0.671	0.655	0.740	0.727	**0.795**	0.787	0.813	0.807
	4.0	0.699	0.680	0.744	0.734	0.791	0.784		
	8.0	0.696	0.680	**0.745**	**0.737**				
	16.0	**0.708**	**0.690**	0.728	0.716				
GAN$_{h=3}$	0.0	0.680	0.662	0.738	0.726	**0.792**	**0.787**	**0.821**	**0.817**
	1.0	0.686	0.664	0.746	0.739	0.785	0.779	0.817	0.813
	2.0	0.684	0.658	0.744	0.731	0.783	0.776	0.802	0.799
	4.0	0.703	0.693	0.735	0.726	0.788	0.781		
	8.0	**0.715**	**0.705**	**0.759**	**0.749**				
	16.0	0.700	0.684	0.713	0.700				
WGAN$_{h=1}$	0.0	0.657	0.637	0.716	0.703	0.768	0.762	0.802	0.799
	1.0	0.663	0.645	0.737	0.726	0.774	0.766	**0.823**	**0.819**
	2.0	0.682	0.666	**0.749**	**0.738**	0.772	0.765	0.822	0.817
	4.0	**0.705**	**0.686**	0.733	0.725	**0.805**	**0.800**		
	8.0	0.691	0.670	0.744	0.732				
	16.0	0.691	0.675	0.743	0.732				
WGAN$_{h=3}$	0.0	0.657	0.637	0.717	0.706	0.779	0.771	**0.808**	**0.804**
	1.0	0.667	0.645	0.745	0.735	0.778	0.771	**0.808**	0.803
	2.0	0.684	0.666	0.725	0.710	0.784	0.777	**0.808**	0.802
	4.0	0.690	0.669	0.756	0.745	**0.792**	**0.783**		
	8.0	**0.716**	**0.702**	**0.760**	**0.751**				
	16.0	0.708	0.685	0.757	0.748				

Table B.2: Classification performance. Best performing results per combination of model and labelled data ratio r_L are marked in bold.

Next, figures B.1 to B.4 visualise the sub-task performance per model. Plots of the baseline performance, models using only labelled data and the best-performing models can be found in section 5.2.

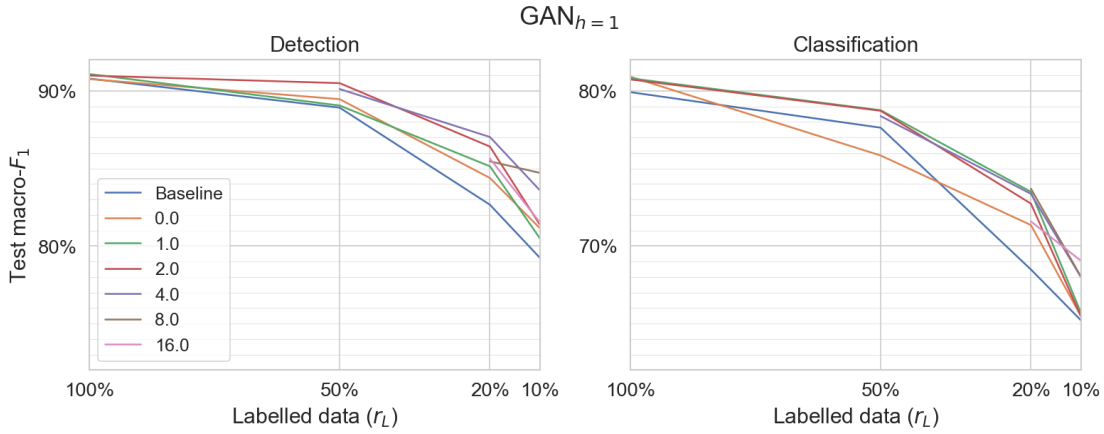

Figure B.1: Sub-task performance for GAN$_{h=1}$.

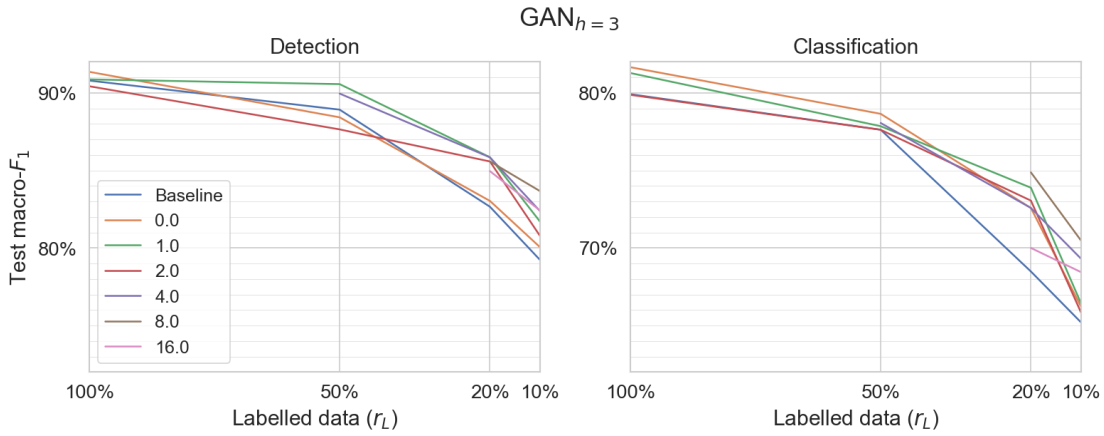

Figure B.2: Sub-task performance for GAN$_{h=3}$.

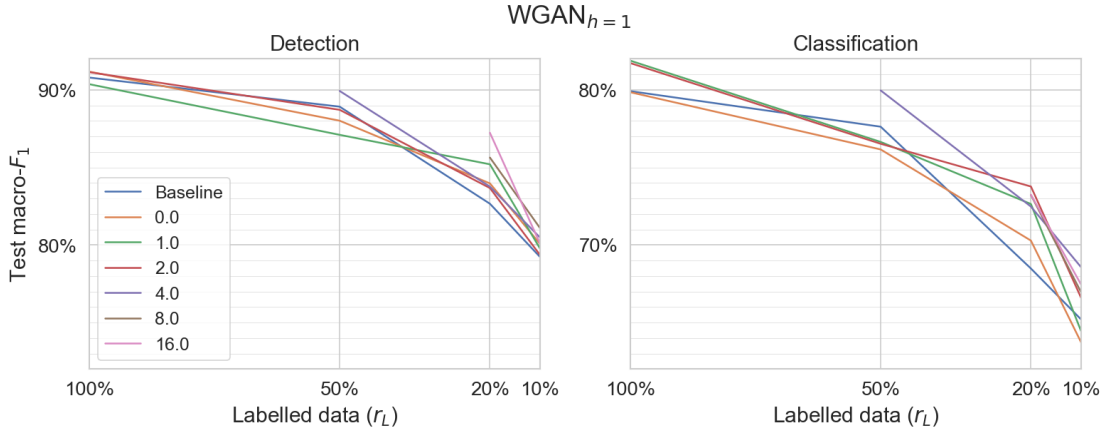

Figure B.3: Sub-task performance for WGAN$_{h=1}$.

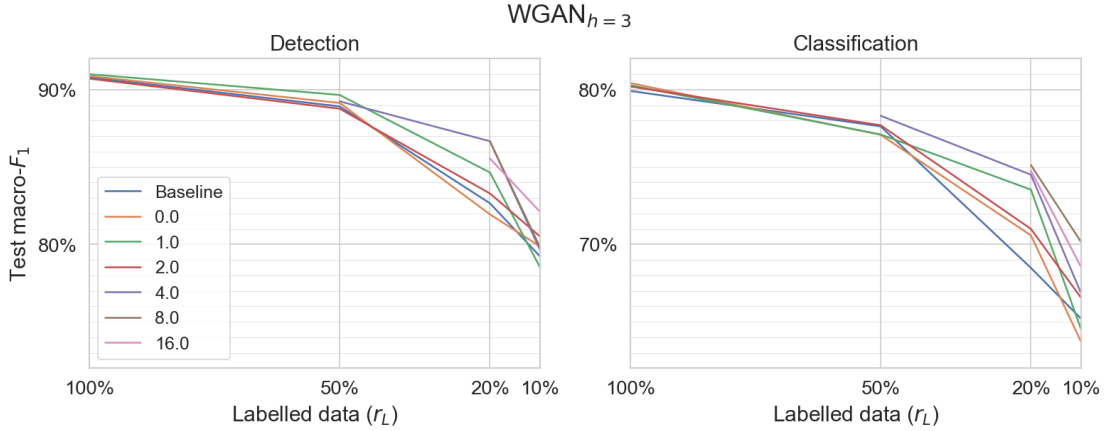

Figure B.4: Sub-task performance for WGAN$_{h=3}$.