CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Catastrophic Forgetting in Language Models

Master's thesis in Computer science and engineering

Tiantian Peng
Shakila Tayefeh

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Catastrophic Forgetting in Language Models

Tiantian Peng
Shakila Tayefeh

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Catastrophic Forgetting in Language Models
Tiantian Peng
Shakila Tayefeh

Supervisor: Marwa Naili, Department of Computer Science and Engineering
Examiner: Richard Johansson, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2025

Catastrophic Forgetting in Language Models
Tiantian Peng
Shakila Tayefeh
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Catastrophic forgetting remains a persistent challenge in the continual learning paradigm of neural networks, particularly in the context of pre-trained language models. This thesis investigates the phenomenon of catastrophic forgetting in large language models (LLMs), with a focus on BERT, through a series of benchmark evaluations. Specifically, we explore the effects of fine-tuning BERT on a vision-and-language dataset and subsequently evaluate its performance on GLUE and SuperGLUE tasks to assess the retention of previously learned knowledge. A brute-force approach was employed in an attempt to mitigate forgetting, involving standard fine-tuning without regularization or memory replay mechanisms. Contrary to expectations, empirical results demonstrate that the fine-tuned models exhibit degraded performance on benchmark tasks compared to the original pre-trained models, highlighting the severity of catastrophic forgetting. These findings emphasize the need for more sophisticated mitigation strategies and contribute to a deeper understanding of transfer learning limitations in current NLP systems.

Keywords: Catastrophic Forgetting, Continual Learning, BERT, Fine-Tuning, Transfer Learning, GLUE, SuperGLUE, Natural Language Processing (NLP)

Acknowledgments

During the journey, I would like to express my deepest gratitude to those who have silently supported and encouraged me.

First and foremost, I want to thank my girlfriend. Throughout this journey, she has been my steadfast pillar and selfless supporter. Her understanding, patience, and encouragement have been my greatest motivation. Whenever I faced challenges, she was always by my side, giving me strength and confidence. Her support made me believe in myself, overcoming many difficulties. Thank you for the invaluable companionship.

Secondly, I want to express my gratitude to my teammates. You are my most reliable partners in this thesis, and together we worked collaboratively to create this achievement. Your expertise, team spirit, and hard work made the entire process smoother and more enjoyable. We overcame many challenges together, sharing the joy of success. Wish you all the best in the future.

I would also like to extend my heartfelt thanks to my supervisor, Richard. His guidance, insights, and unwavering support were instrumental throughout this journey. Richard's expertise and thoughtful feedback constantly challenged me to think critically and improve the quality of my work. His encouragement during difficult moments gave me the confidence to push forward, and his mentorship has been invaluable to my growth. Thank you for being a dedicated and inspiring supervisor.

Once again, thanks to everyone who has appeared in my life. Your support has made me stronger. May we face future challenges together and create even better times.

Tiantian Peng, Stockholm, 2025-07-14

Contents

List of Figures

List of Tables

1

Introduction

Large **L**anguage **M**odels, simply **LLM**s are reshaping various aspects of human life. Powered by the Transformer-model architecture, GPT-4 [1], Bard, and HuggingChat [2] are pioneering examples of AI-driven chat interfaces revolutionizing the field of natural language processing. Exposed to massive amounts of data during their pre-training phase, LLMs are capable of capturing enormous linguistic knowledge through their vast set of parameters, which marks them as promising contributors to come into effect in various natural language processing tasks.

Although LLMs possess the extraordinary ability to generate synthetically coherent and consistently fluent output statements, they lack factual knowledge. Recent studies attempt to solve this issue of concern by providing LLMs with complementary data modalities. That explained, [3] develops a strategy to maximize vision-to-text knowledge transfer and proposes a novel approach to evaluate the multi-modal model's performance successfully. At the core of this study, a multi-modal CLIP-BERT model [4] is trained on the vision-and-language dataset (see section 3.1.2) with an MLM pre-training task and evaluated on the Memory Colors dataset (see section 3.1.1). Despite the aforementioned endeavors to enrich the ability of LLMs, a more significant challenge must be addressed, affecting the entire arc of their development: **Catastrophic Forgetting**.

Catastrophic interference or catastrophic forgetting is a phenomenon discovered as a fundamental limitation of neural networks abruptly and drastically forgetting previously learned information as new information is learned [5]. Inspired by the human brain, neural networks suffer from forgetting catastrophically. The human brain is prone to gradually forgetting previously learned information as a natural cognitive system. As new information is acquired, human cognition exhibits gradual forgetting of old information. Nevertheless, complete disruption of learned information seldom occurs in natural cognition; significantly, they do not forget catastrophically [6].

In the case of natural language processing, LLMs often face the issue of catastrophic forgetting. The models are pre-trained on a specific pre-training task, and while they primarily might demonstrate exemplary performance in pre-training and fine-tuning on a downstream task, if exposed to previously learned pre-trained training corpus again, they show a poor performance, which is the recognition of catastrophic forgetting and how it is measured using benchmarks. While not inherent, the loss of valuable information affects the transfer learning process where models must adapt

to downstream tasks; therefore, it is necessary to address catastrophic forgetting, particularly in critical applications.

While the proposed approach in [3] achieves significant results, it is crucial to closely inspect its performance in the context of catastrophic forgetting, which lays the foundation for this work. This study extends the discussion to the implications of catastrophic forgetting and proposes mitigation strategies while exploring the impact of different benchmarks.

1.1 Aims

The purpose of this study is to investigate, measure, and mitigate catastrophic forgetting in the domain-specific task based on [3]. It stands out as a pioneering endeavor to lay the foundation for a deeper understanding of catastrophic forgetting in LLMs within the context of benchmarks. Contrastingly, it explores the phenomenon of catastrophic forgetting by fine-tuning BERT on captions of the vision-and-language dataset, aiming to answer the following research questions:

- How does the pre-trained BERT model perform on different benchmarks?
- How does the BERT model perform when fine-tuned on the captions of the vision-and-language dataset?
- Does the fine-tuned BERT model suffer from catastrophic forgetting?
- How does the choice of benchmarks impact the catastrophic forgetting phenomenon?

1.2 Limitations

This work is centralized on humans' potential to learn knowledge from non-linguistic modalities. Diving into the details, these procedures are the pure reflection of how humans encounter knowledge to be learned. For instance, individuals learn arithmetic operations sequentially. Sensationally, they learn addition prior to multiplication, forming a factual basis for them to act in accordance with the learning process [5]. Sensibly, narrowing the concept to this work's particular case, individuals learn Sign language expressed in the visual-gestural modality opposite to the auditory-vocal process [7]. These thoroughly imply that knowledge is acquired sequentially in different modals over time.

Accordingly, following the occurrence of poor performance by the model and, as a result, the phenomenon of catastrophic forgetting is noteworthy as it remarks an open question of how an output statement generated by an observed pattern would lead to an unconcealed question of whether this is an issue of concern. Disruption of learned knowledge by new learning is considered a natural behavior in humans and complex language models; however, as scientists, we are obliged to present truthful natural language processing systems for further applications, which makes this a critical issue. To carry out such a task, the most challenging part is to validate

the problem given the available resources, particularly the computational resources. The computational resources allocated to this study by the Chalmers University of Technology facilitated the process of identifying the claimed hypothesis and partial mitigation of the problem; however, the study can be extended to benefit from more advanced computational resources inclined with the nature of this problem to provide more conceptual and empirical tools necessary to pave the way for further experiments. Nevertheless, possible options for further extension are proposed as the following:

- Although the primary focus of this study is on uni-modal models (text only), it can be extended to multi-modal models.
- Inspired by humans' potential to learn knowledge from non-linguistic modalities, a similar study can be performed on auditory vocal-to-text knowledge transfer.
- A similar study can be performed on vision-to-text knowledge transfer for the particular case of sign language.
- Color perception is varied across languages and cultures. A similar study can be performed in different languages and color categorization [8].
- In addition to the proposed mitigation approaches in this study, trajectory ensembling can be used as an innovative approach.

1.3 Road map

Excluding the current chapter that provides an introduction to this thesis, including the research questions and limitations, this thesis is structured as follows:

- Chapter 2 provides the theory behind established techniques used in the thesis.
- Chapter 3 describes the methods for investigating, measuring, and mitigating the problem.
- Chapter 4 presents the results of each method.
- Chapter 5 concludes the thesis along with a discussion.

2

Theory

This chapter introduces and describes the theory and concepts used throughout this thesis.

2.1 Language

Communication is the fundamental basis of humans' daily life. Humans communicate by means of pointing, facial expressions, iconic gestures, and language. Although communication can be formed without the use of language, the biological evolution of human cognition has led to the enhancement of an epiphenomenon of the human capacity to share intentions in the use of communication [9]. Language allows individuals to express their thoughts, feelings, and ideas to others and to understand the messages of others. **Language** is a system for associating form and meaning consisting of grammar and vocabulary, which can be conveyed through speech, sign, or writing. Historically, languages evolve, diversify, and can be influenced by cultural and social factors, leading to a set of rules for grammar, vocabulary, pronunciation, regional variations, and dialects, which can vary widely from one language to another. Linguistics is the study of language and communication, which involves examining the structure, use, and development of language in all its aspects.

2.1.1 Natural Languages

A **natural language** is the cornerstone of the evolvement of a language through subconscious use and repetition in humans. English, Swedish, Persian, French, Dutch, Norwegian, and Japanese are the native speech of humans and, accordingly, natural languages.

2.2 Machine Learning

Machine learning is the field of facilitating computers by means of mathematical algorithms to learn, reason, and act on the basis of data. This is accomplished by using a learning algorithm that is capable of automatically adapting the parameters of a mathematical model to agree with processed domain-specific data, extracting useful information, making predictions regarding unknown properties, and suggesting actions by making data-driven decisions [10].

Data is a set of unprocessed facts, values, symbols, text, sound, or photos that form the core basis of machine learning. **Datasets** are collections of data instances that contain a set of features or attributes fed to machine learning models.
A **training set** is a set of data instances used to fit the parameters of the machine learning model.
A **validation set** is a set of data instances used to provide an unbiased evaluation of a model fit on the training data set while tuning the parameters of the machine learning model.
A **test set** is a set of data instances used to provide an unbiased evaluation of a final model fit on the training data set.
The strategies of dataset division, interference, and size depend on the domain-specific task.

2.2.1 Supervised Machine Learning

Consider the training set as $\tau = \{x_i, y_i\}_{i=1}^{n}$ where x is a p-dimensional vector, $x = [x_1\ x_2\ ...\ x_p]^T$ with ^{T} denoting the transpose and y is a scalar value. Each element of the input vector x represents information, either categorical or numerical, proven to be useful and is considered to be relevant for the domain-specific task. Additionally, every input x in the training set is in relationship with a corresponding output y.

Supervised machine learning is a machine learning application case that tries to establish the goal of learning labels or target values y or mathematically extract as much information as possible from τ and then predict them for the unseen test dataset.

The term "supervised" is followed by the fact that the training data contains not only input values x_i but also output data y_i. In other terms, every input x_i is accompanied by a label y_i, and labels are at hand [10].

2.2.1.1 Classification

Classification is a sub-type of supervised machine learning application cases where the output y is categorical, having a finite number of discrete values. It involves constructing a classifier or classification model with the aim of assigning each input x to M possible output values. M is denoted by integers $1, 2, ..., M$ where $M \geq 2$ is the output classes or labels. Moreover, the ordering of integers is arbitrary and does not infer the ordering of the classes or labels. Binary classification is a special case of classification where the problem is concerned with $M = 2$ classes or labels 1 and -1 [10].

2.2.1.2 Regression

In supervised machine learning application cases, **Regression** primarily involves predicting a numerical output y. The type of the output y makes a distinctive difference between supervised machine learning application cases, classification, and regression. Although the main focus of this study lies elsewhere, it is worthy to

include a brief introduction and leave further details to be explored by curious minds among the readers interested in the subject.

2.3 Natural Language Processing

Natural language processing is a multidisciplinary field that enables computers to understand, process, and manipulate natural languages.
Knowledge acquisition of computers through natural language necessitates a wide range of methods as natural languages are large, filled with ambiguities, and take different forms. Contemporary approaches to natural language processing depend heavily on machine learning, which makes the process of analyzing natural language smoother [11].

2.4 Tokenization

Natural languages can be conveyed through writing or, alternatively, text. Ultimately, a written text is composed of characters, including letters, digits, and punctuation, along with others, based on the language.
To analyze the structure of the text, the most important preliminary step is tokenization.
Tokenization refers to the technique or process of splitting a text, sentence, or document into smaller pieces or units called tokens. Consider the following sentence:

Steve pelted the squirrels with acorns.[1]

Performing tokenization on this sentence involves splitting it into comprehensible tokens that can be assigned meaning. Thus, the aforementioned sentence can be tokenized in multiple arrangements.

Tokenization Type	Tokenized Sentence
Character Tokenization	['S', 't', 'e', 'v', 'e', 'p', 'e', 'l', 't', 'e', 'd', 't', 'h', 'e', 's', 'q', 'u', 'i', 'r', 'r', 'e', 'l', 's', 'w', 'i', 't', 'h', 'a', 'c', 'o', 'r', 'n', 's', '.']
Word Tokenization	['Steve', 'pelted', 'the', 'squirrels', 'with', 'acorns', '.']
Sub-Word Tokenization	['Steve', 'pelt', 'ed', 'the', 'squirrel', 's', 'with', 'acorn', 's', '.']

Table 2.1: A randomly chosen sentence from the vision-and-language dataset tokenized using different tokenization approaches.

[1]The sentence is randomly chosen from the captions of the vision-and-language dataset (see Section 3.1.2)

The purpose of tokenization is to facilitate the interpretation of computers; nevertheless, such arrangements are followed by complications related to their own vocabulary.

Character tokenization is the process of splitting the text into characters. For character tokenization, the vocabulary size is as many characters as the language possesses. For instance, character tokenization will lead to a vocabulary of size 26 characters in English, while the same amounts for Swedish, French, Italian, Portuguese, and Ukrainian are 29, 26, 21, 23, and 33, respectively. Although the vocabulary size is limited, which guarantees the absence of out-of-vocabulary or unknown characters, it results in the exponential growth of the output. Character tokenization also requires another interpretation layer to establish a relationship between the characters and their meaning.

Word tokenization is splitting the text based on the delimiters that include different punctuation characters and spaces. As the simplest type of tokenization, the word tokenization is excessively regulated in its vocabulary. Handling a large vocabulary where every word is considered a token can be challenging. Additionally, for word tokenization, out-of-vocabulary words are considered an issue for work tokenization, and while they can be addressed by assigning a special token presenting them, their interpretation is exigent. Consequently, a vocabulary of the right size is necessary to deliver this method efficiently and accurately. A vocabulary of the most common words has less sparsity and less skewed word frequency and can be a suitable solution.

Eventually, **sub-word tokenization** addresses all the issues of concern. Sub-word tokenization is similar to word tokenization. Moreover, it splits the text into units learned from the text corpora instead of relying solely on linguistic rules. Seldom utilizing the competencies of morphemes in the language, the parts of speech, structure, meaning, and function are inherently altered within words. The combination expedites the creation of an efficient and accurate valuable vocabulary that can also assign meaning to an out-of-vocabulary word, which is the primary goal of such tokenization. For instance, 'pelted' in the aforementioned example is aligned with 'pelt' as a token. The reason is that the suffix 'ed' changes the form of the verb. Subsequently, the computer acknowledges that this specific suffix is only added to verbs to form the past tense and uses this valuable inference to add it to its associated token instead of passing it as an out-of-vocabulary word.

Tokenization is a necessary yet challenging process to implement in natural language processing that must communicate punctuation, contractions, and significant symbols that can change the meaning of words.

2.5 Language Modeling

Arising from the writings and views of the anthropologist-linguist Edward Sapir and the linguist Benjamin Lee Whorf, the linguistic relativity hypothesis, also known as the Sapir-Whorf hypothesis, suggests that our native language molds our perception of the world through its semantic structures [12].

The process for youth, as language learners, commences at an early stage of life when they begin to become familiar with the environment surrounding them. Following their own development trajectory, young language learners learn by observing, listening, exploring, experimenting, and asking questions, leaning deeply on their non-linguistic knowledge [12]. Explicitly, this proceeds towards prompt cognitive stimulation through engaging various cognitive processes, encompassing perception, memory, learning, attention, decision making, and language abilities contributing to cognitive growth [13].

With the ultimate goal of developing communication skills, language learners must enhance language modeling as a demonstration tool to acquire language. Originating as a strategy to use words, it emerges to making sentences, engaging in meaningful conversations, and as adults possessing effective communication skills to better understand individuals and situations for building trust and respect. Closely unveiling the intricate interplay between linguistic relativity and cognitive abilities, language modeling lies within this process.

Assuming that AI is inspired by the human brain, language and language modeling establish a new meaning; A **language model** is a function, or algorithm for learning that function, that captures the salient statistical characteristics of the distribution of sequences of words in a natural language [14].

Proposed by the father of information theory, Claude Shannon, it allows making probabilistic predictions of the next word given the preceding continuous sequence of words or n-gram.

Figure 2.1: A sunburst chart where each sector corresponds to a series of n-grams. Consider the sentence from the vision-and-language dataset (see Section 3.1.2) *"Steve pelted the squirrels with acorns."*. The inner circle of the chart includes the sentence itself as it is a 6-gram. As we move toward the outer circles, unigrams, bigrams, and a few trigrams are illustrated.

Emanating from his work during the Second World War, Shannon proposed the idea in his paper "A Mathematical Theory of Communication," which later laid the foundation of communication. In the paper, Shannon describes a communication system consisting of some essential parts, one of which is an information source that produces a message or a sequence of messages to be communicated to the receiving terminal [15]. As a rule-based system using language models and n-grams, this can be considered the first natural language processing system. It demonstrates the pure application of a language model that produces word sequences as outputs [11].

2.6 Word Embeddings

A **word embedding** is a fixed-length mathematical representation with which every word of the vocabulary is assigned a vector of real-valued numbers. The relative similarities of the vector-based representation correlate with semantic similarities. Expressly, words that have semantic similarities have similar word embeddings in the vector space. Various methods have emerged over the long haul to obtain word embeddings, which are commonly categorized into two types in accordance with the strategies used to induce them. Strategies that leverage local data or the contextual meaning of words are prediction-based models, whereas strategies that use global information, generally corpus-wide statistics, are statistical models [16]. Word2Vec [17] [18] is an archetype of statistical strategies, while GloVe (Global Vectors for Word Representation) [19] is a prediction-based word embedding which is generally invoked by the operation of artificial neural networks to induce language models.

2.7 Artificial Neural Networks

Distinct regions of the human brain enable them to perform different tasks. Considering verbal communication among a group of individuals, comprehending an utterance is achieved by utilizing the competencies of different brain regions to successfully map and integrate linguistic meaning into existing world knowledge. Neurons, or originally nerve cells, are the fundamental units that perform the computation task while being connected to process the information.

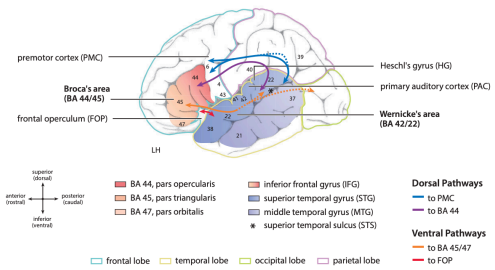


Figure 2.2: Anatomical and cytoarchitectonic details of the left hemisphere where the neuroanatomy of language is located [9]. The cerebral cortex is the outer layer lying on top. It is one of the largest and best-developed segments accommodating enormous amounts of neurons.

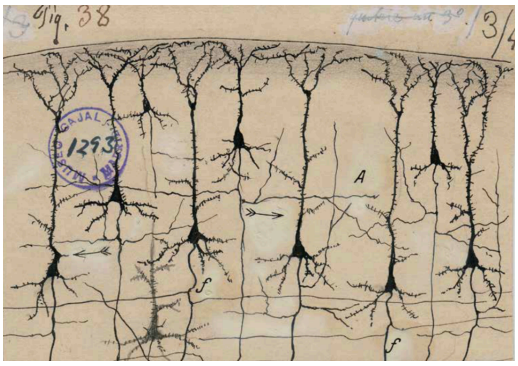
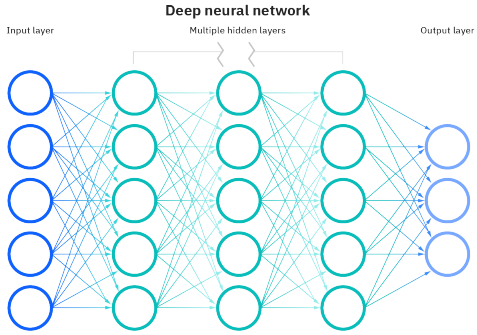


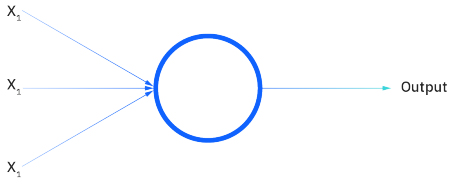
Figure 2.3: Drawing by Santiago Ramón y Cajal, the Spanish neuroscientist who received the Nobel Prize in Physiology and Medicine in 1906 together with Camillo Golgi 'in recognition of their work on the structure of the nervous system.' It shows neurons in the cerebral cortex. The cell bodies of neural cells, axons, and dendrites can be distinguished by connecting together to form a neural network responsible for processing visual, audio, and sensory data [20].

Conventionally, the network of neurons in the brain is identified by the term **neural networks**. Contrastingly, **artificial neural networks**, or simply neural networks, are a type of computational system that reflects the behavior of the human brain, allowing computers to solve different problems using the same mechanism.
Concurrently, in 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts portrayed the idea of a model of how neurons function in the human brain. Termed "connections," the model consisted of simple electrical circuits to simulate intelligent behavior. As a pioneering innovation representing a groundbreaking and revolutionary first attempt, it laid a crucial theoretical foundation and paved the way for decades of future endeavors.
Inspired by their work, Donald Hebb, a neuropsychologist, took the idea further and proposed fundamental concepts regarding the complex process of neural pathways between neurons in the brain. Translating the aforementioned networks into computational systems, the first network was successfully implemented in 1958 by Frank Rosenblatt [21].
1982 holds a crucial occurrence of the transition of statistical models to modern-day neural networks, as John Hopfield introduced his model accounting for associative memory after periods of challenges and stagnation in the years following Rosenblatt's breakthrough. Inspired by Hebb's work and the rise of computer power, his model was a recurrent neural network with dynamical trajectories converging to fixed point attractor states and described by an energy function. The model was then put to work for operation on sequence data when the computer scientist Geoffrey Hinton proposed the idea of representing words as vectors. Collectively, natural language processing gained an instantaneous resurgence in popularity.
Encountering a reconciliation with state-of-the-art research in 2003, neural language models were presented as a critical point for natural language processing systems [14]. Nourished by word embeddings [17] [18] and Glove [19], the system continued to grow to represent the attention mechanism [22], transformers [23], and pre-trained language models.

Dependent upon the domain-specific task, there are multiple strategies for the neural networks' architecture. Nevertheless, they hold a primary structure that is comprised of an input layer, neurons organized in the hidden layers, and an output layer. Connected to each other within the structure to allow the data flow, each neuron has an associated weight and threshold.



(a) The layout of a neural network with multiple hidden layers.



(b) A close-up view of a single neuron or a Perceptron. A Perceptron, created by Frank Rosenblatt in 1958, is the oldest neural network unit. The weights and thresholds are applied to inputs to produce the desired output using the activation function.

Figure 2.4: Neural Network Components [2]

Given p input patterns $x^{(\mu)}$ each possessing n components and l layers with input and output layers denoted as $l = 0$ and $l = L$, respectively, the state variables for neurons in layer l are $V_j^{(l)}$, the weights and thresholds connected to the neurons, and the activation function are noted in the following equation.

$$V_j^{(l)} = g(\sum_k w_{jk}^{(l)} V_k^{(l-1)} - \theta_j^{(l)}) \tag{2.1}$$

Each input pattern $x^{(\mu)}$ is fed to the input layer of the network. Upon propagating through the layers, the weights and thresholds are applied to the neurons to compute the local field and produce the desired output using an activation function, g iteratively [20].

An activation function is a mathematical transformation determining whether a neuron must be fired or activated based on the received input. The choice of an appropriate activation function for the downstream task is vital for the convergence rate of the neural network as each activation function has its own set of characteristics impacting the neural network's performance [24].

The output of the network with M components is indicated by an activation function called softmax outputs, which is described in the following equation where $b_i^{(L)} = \sum_j w_{ij}^{(L)} V_j^{(L-1)} - \theta_i^{(L)}$ is the local field in the output layer.

[2] IBM. (n.d.). What are neural networks?. IBM.

$$O_i = \frac{e^{\alpha b_i^{(L)}}}{\sum_{k=1}^{M} e^{\alpha b_k^{(L)}}} \tag{2.2}$$

Two significant properties of the softmax output (by construction; setting α to unity) are (1) $0 \leq O_i \leq 1$ and (2) $\sum_{i=1}^{M} O_i = 1$. These properties allow softmax output units to be interpreted as probabilities where the input pattern $x^{(\mu)}$ must be assigned to one of the M classes [20].

Leveraging supervised machine learning, a neural network adapts the weights and thresholds to approximate the desired outputs in relation to their associated inputs. This can be elucidated as an optimization problem of the model's parameters. Considering Θ as the set of all parameters, the optimization problem to find the suitable value for Θ is of the following form.

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, J(\Theta), \quad where \quad J(\Theta) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, \Theta) \tag{2.3}$$

$J(\Theta)$ indicates the cost function, whereas $L(x_i, y_i, \Theta)$ is the **loss function**. The loss function is a measure that helps to evaluate how well the model performs. However, the functional form of the loss function depends on the downstream task [10].

2.7.1 Feedforward Neural Network

Feedforward neural networks are the simplest form of artificial neural networks. The term "feedforward" indicates that the data flows and is processed forward through the network in one direction in the absence of any cycles.

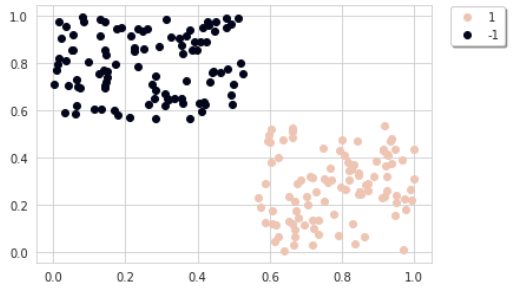
2.7.1.1 Single-Layer Perceptron

Possessing a simple form, a feedforward neural network has an architecture wherein it consists of a single neuron or perceptron. This architecture is composed of input and output layers termed a **single-layer perceptron** (see Figure 2.4b). Within the single-layer perceptron network, the input is fed to the output layer by means of designated weights. The weights are propagated through the inputs, computing the local field for each neuron. Utilizing an activation function applied on the local field, the neuron is either fired and activated or deactivated, allowing the control of data flow forward through the network. A single-layer perceptron is capable of classifying linearly separable patterns.

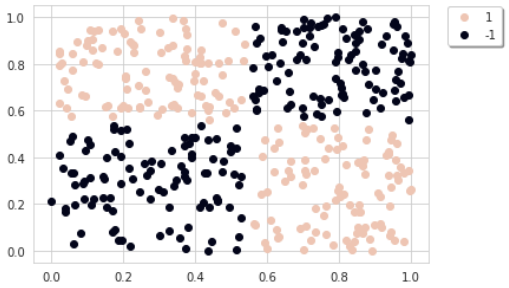
2.7.1.2 Multi-Layer Perceptron

Multi-Layer Perceptron is another class of feedforward neural networks where the architecture consists of interconnected feedforward layers of neurons. They comprehend an input layer, one or more hidden layers, and an output layer (see Figure 2.4a) across which the data flows, supplying them to classify non-linearly separable patterns. Each input pattern is fed to the input layer of the multi-layer network.

Upon propagating through the layers, the weights and thresholds are applied to the neurons to produce the desired output by using the activation function iteratively. The activation function is, for the most part, non-linear as the real-world problems are encapsulated thus.



(a) Binary classification problem with two-dimensional real-values uniformly distributed random inputs and classes equal to 1 and -1. Evidently, the binary classes can be separated into two neighborhoods corresponding to each class by a solid line, which is the decision boundary. A single-layer perceptron suffices for determining such decision boundary where the dataset is linearly separable [25].



(b) A special non-linear problem with two-dimensional real-values uniformly distributed random inputs and classes equal to 1 and -1. A multi-layer perception can be used to classify such a dataset.

Figure 2.5: Comparing the ability of single and multi-layer perception to classify linearly and non-linearly separable binary classes.

2.7.2 Recurrent Neural Network

The human brain enables them to actively engage in different tasks. Considering verbal communication among a group of individuals, comprehension is achieved once the linguistic meaning is mapped and integrated into existing knowledge. However, this process is not recurrent but persistent, conveying that it is built upon existing work, progress, or knowledge without constantly commencing from scratch.
Interchangeably, the feedforward layout of a feedforward neural network makes it incapable of capturing the dynamics of sequentially structured data.

Recurrent neural networks are a class of artificial neural networks used to process sequential data. The architecture of this type of neural network contains a directed cycle, allowing data flow in preceding layers within the cycle.

Unlike feedforward neural networks, recurrent neural networks are dynamical networks in which the iteration index t replaces the layer index l, resulting in the following equation with respect to weights, thresholds, and an activation function, g.

$$V_i(t) = g(\sum_j w_{ij}^{(vv)} V_j(t-1) + \sum_k w_{ik}^{(vx)} x_k - \theta_i^{(v)}) \quad for \quad t = 1, 2, ... \tag{2.4}$$

The training set $\tau = \{x(t), y(t)\}$ is fed to the recurrent neural network as a function of time t, where the dynamics of the network are unrolled across time steps with the same underlying weights and thresholds. In other words, weights and thresholds are shared within each layer of the network, giving them the ability to use hidden states or memory to process arbitrary sequences of input up to the current time frame t_c.

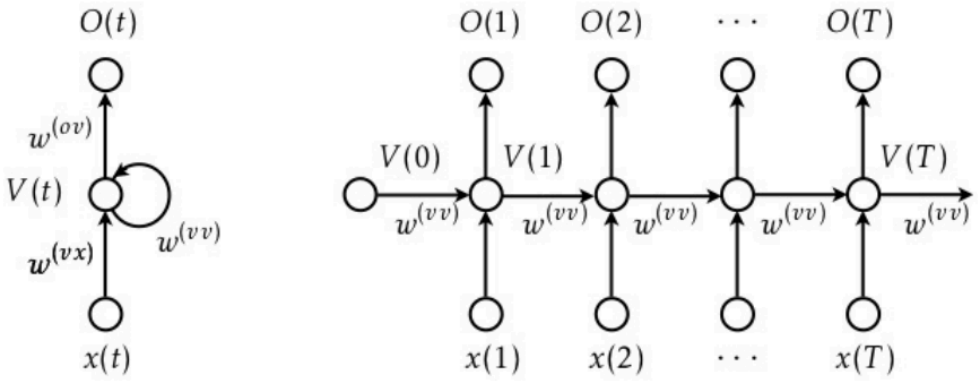


Figure 2.6: Schematic illustration of a recurrent neural network consisting of input, hidden, and output layers unfolded in time on the right-hand side [20]. Subject to the input and output size, there are four different variants of a recurrent neural network architecture. While the illustration is a many-to-many recurrent neural network, there are one-to-one, one-to-many, and many-to-one variations.

The characteristics of this architecture of artificial neural networks candidate them as an empowering toolkit in the field of natural language processing.

2.7.2.1 Bidirectional Recurrent Neural Network

The **bidirectional recurrent neural network**'s architecture was proposed to process arbitrary sequences of input in the past and future of a specific time frame. The aspiration emerged from splitting the state neurons of a standard recurrent neural network into two parts responsible for positive time direction (forward states) and negative time direction (backward states), respectively [26].

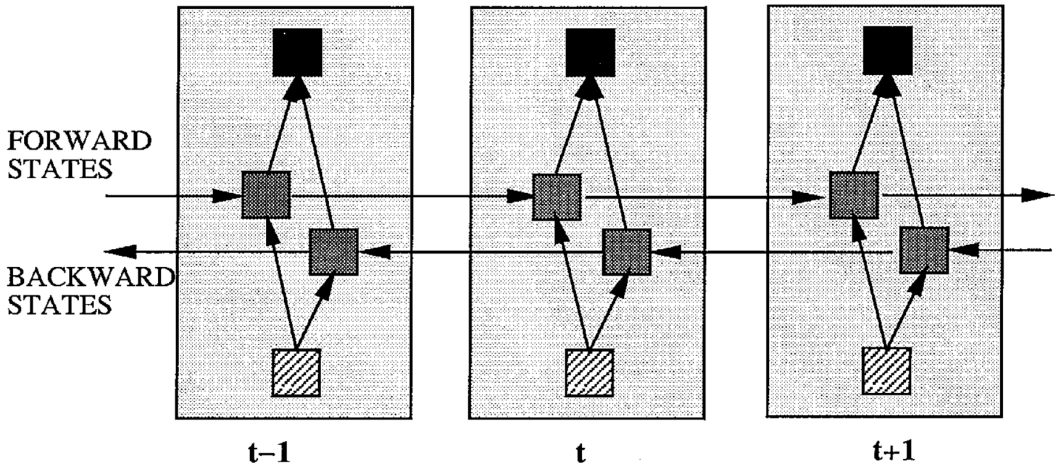


Figure 2.7: Bidirectional recurrent neural network architecture unfolded at time step t. Evidently, the outputs from forward states are not connected to inputs of backward states and vice versa, showing the general structure of such a class of artificial neural networks [26].

2.7.2.2 Long Short-Term Memory [3]

As a class of recurrent neural networks, **Long Short-Term Memory** networks or broadly recognized **LSTM** networks possess memory blocks in their hidden layers. As a basic unit, a memory block contains one or more memory cells and a pair of adaptive, multiplicative gating units that gate input and output to all cells in the block. Constant Error Carousel, or CEC, is a recurrently self-connected linear unit that lies at the core of each memory cell.

CEC's activation or Cell state s_c is determined by its current state, input to the cell itself net_c, input gate net_{in}, and output gate net_{out}. The gates are responsible for controlling the information flow, and in the case of gates being closed (activation around zero), irrelevant inputs and noise are prohibited from entering the cell.

Input gate activation y^{in} and output gate activation y^{out} read

$$net_{out_j}(t) = \sum_m w_{out_{jm}} y^m(t-1) \quad and \quad y^{out_j}(t) = f_{out_j}(net_{out_j}(t)) \tag{2.5}$$

$$net_{in_j}(t) = \sum_m w_{in_{jm}} y^m(t-1) \quad and \quad y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \tag{2.6}$$

where memory blocks are indexed j at discrete time step $t = 1, 2, ...$ with sigmoid function f in the range $[0, 1]$.

The input to the cell itself is

[3] Historically, the bidirectional neural network was introduced after the recognition of LSTMs in 1997.

$$net_{c_j^v}(t) = \sum_m w_{c_j^v m}\, y^m(t-1) \tag{2.7}$$

where c_j^v denotes the v-th cell of the j-th memory block.

The internal state of the memory cell is computed by

$$s_{c_j^v}(0) = 0 \quad and \quad s_{c_j^v}(t) = s_{c_j^v}(t-1) + y^{in_j}(t)\, g(net_{c_j^v}(t)) \quad for \quad t > 0 \tag{2.8}$$

with g, a centered sigmoid function in the range $[-2, 2]$.

The cell output y^c is computed by

$$y^{c_j^v}(t) = y^{out_j}(t)\, h(s_{c_j^v}(t)) \tag{2.9}$$

with h, a centered sigmoid function in the range $[-1, 1]$.

Last but not least, the output units k are

$$net_k(t) = \sum_m w_{km}\, y^m(t-1) \quad and \quad y^k(t) = f_k(net_k(t)) \tag{2.10}$$

with weights connecting from unit m to unit k and sigmoid function f in the range $[0, 1]$.

The described structure creates an additional module that enables the LSTM to learn when to remember and when to forget pertinent information [27], [28].

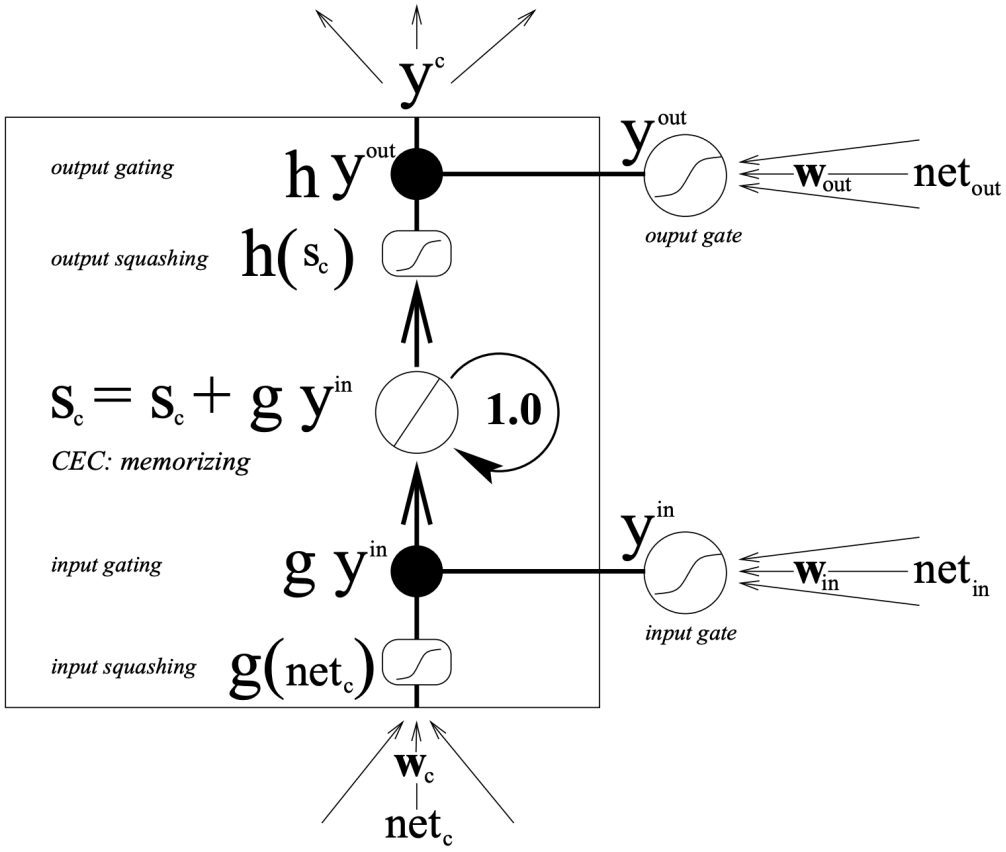


Figure 2.8: Internal architecture of the LSTM [28].

2.8 Theory of Neural Computation [4]

The diverse architectures of artificial neural networks are fundamentally geared towards their sole purpose of learning.
With the ultimate goal of learning, a neural network is a parametric model whose parameters, Θ, can be adjusted by solving an optimization problem. By way of explanation, a neural network is expected to determine a configuration that minimizes the cost function $J(\Theta)$ (see equation 2.3) [20], [29].

2.8.1 Gradient Descent and Backpropagation

The procedure of minimizing $J(\Theta)$ as an objective function, parameterized by a neural network parameters $\Theta \in \mathbb{R}^d$, involves updating the parameters in the opposite direction of the gradient of the objective function $\partial J(\Theta)$ with respect to the parameters [30], [31]. Primarily introduced by Augustine-Louis Cauchy, the French mathematician and engineer, this aims to establish the conceptual framework that

[4] The title is inspired by John Hertz, Andres Krogh, and Richard G. Palmer's dedication to the book "Introduction to the Theory of Neural Computation" [29].

underlies the most commonly used optimization algorithm in neural computation, **gradient descent** [32].

As an optimization algorithm, gradient descent intends to perceive a learning rule facilitating the exploration of finding a set of parameters, including weights and thresholds, minimizing the objective function by successive arbitrary improvement from an initiating state.

Pivotal factors central to this are, the cost function being only influenced by weights and input patterns, and the differentiability of the activation function. Nevertheless, the second factor is of no concern and can be handled by selecting a suitable activation function in regard to the downstream task, while the loss function only depends on the current state of the network, which evolves to be minimized by following the direction of the slope of the created surface downhill until a local minimum is reached [33].
Regardless, it is worth noting that a fascinating aspect of training artificial neural networks is their adaptability and versatility, making them unrestricted by rigid principles to develop an internal structure appropriate for any particular task domain. Contributing factors central to this freedom are activation and loss functions with an abundance to choose from, shaping the behavior of such models [31].

Drawing from the insights provided to indicate the outputs of the network with M components using the softmax activation function (see equation 2.2), softmax units are used analogously in conjunction with a loss function called cross-entropy loss function.

$$L = -\sum_{i\mu} t_i^{(\mu)} \log O_i^{(\mu)} \tag{2.11}$$

In this function, log stands for the natural logarithm where $t_i^{(\mu)}$ are the corresponding vectors in the training set and $O_i^{(\mu)}$ are the softmax units making it minimal when $O_i^{(\mu)} = t_i^{(\mu)}$.

Evaluating the learning rule for the gradient descent algorithm reads

$$\frac{\partial L}{\partial w_{mn}} = -\sum_{i\mu} \frac{t_i^{(\mu)}}{O_i^{(\mu)}} \frac{\partial O_i^{(\mu)}}{\partial w_{mn}}. \tag{2.12}$$

Computing

$$\frac{\partial O_i^{(\mu)}}{\partial w_{mn}} = \sum_l \frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} \frac{\partial b_l^{(\mu)}}{\partial w_{mn}}, \tag{2.13}$$

using

$$\frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} = O_i^{(\mu)}(\delta_{il} - O_l^{(\mu)}) \quad and \quad \frac{\partial b_l^{(\mu)}}{\partial w_{mn}} = \delta_{lm} V_n^{(\mu)} \tag{2.14}$$

where δ_{il} and δ_{lm} are the Kronecker delta where $\delta_{il} = 1$ if $i = l$, $\delta_{lm} = 1$ if $l = m$, and zero otherwise, respectively.

It can be obtained

$$\frac{\partial O_i^{(\mu)}}{\partial w_{mn}} = \sum_l \frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} \frac{\partial b_l^{(\mu)}}{\partial w_{mn}} = O_i^{(\mu)}(\delta_{im} - O_m^{(\mu)})V_n^{(\mu)}. \tag{2.15}$$

Subsequently

$$\delta w_{mn} = -\eta \frac{\partial L}{\partial w_{mn}} = \eta \sum_{i\mu} t_i^{(\mu)}(\delta_{im} - O_m^{(\mu)})V_n^{(\mu)} = \eta \sum_{\mu} (t_m^{(\mu)} - O_m^{(\mu)})V_n^{(\mu)}, \tag{2.16}$$

considering that $\sum_{i=1}^{M} t_i^{(\mu)} = 1$ for the type of classification where each input belongs to precisely one of M classes.

The corresponding learning rule for the thresholds derives

$$\frac{\partial L}{\partial \theta_m} = -\sum_{i\mu} \frac{t_i^{(\mu)}}{O_i^{(\mu)}} \frac{\partial O_i^{(\mu)}}{\partial \theta_m}. \tag{2.17}$$

Computing

$$\frac{\partial O_i^{(\mu)}}{\partial \theta_m} = \sum_l \frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} \frac{\partial b_l^{(\mu)}}{\partial \theta_m}, \tag{2.18}$$

using

$$\frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} = O_i^{(\mu)}(\delta_{il} - O_l^{(\mu)}) \quad and \quad \frac{\partial b_l^{(\mu)}}{\partial \theta_m} = -1 \tag{2.19}$$

where δ_{il} is the Kronecker delta where $\delta_{il} = 1$ if $i = l$ and zero otherwise.

It can be obtained

$$\frac{\partial O_i^{(\mu)}}{\partial \theta_m} = \sum_l \frac{\partial O_i^{(\mu)}}{\partial b_l^{(\mu)}} \frac{\partial b_l^{(\mu)}}{\partial \theta_m} = -O_i^{(\mu)}(\delta_{im} - O_m^{(\mu)}). \tag{2.20}$$

Subsequently

$$\delta \theta_m = -\eta \frac{\partial L}{\partial \theta_m} = -\eta \sum_{i\mu} t_i^{(\mu)}(\delta_{im} - O_m^{(\mu)}) = -\eta \sum_{\mu} (t_m^{(\mu)} - O_m^{(\mu)}). \tag{2.21}$$

Due to the feedforward layout of a feedforward neural network, which lays the foundation for proof of concept, the neurons are updated from left to right or alternately forward while the errors are updated backward, critically known as [backpropagation]backpropagation [5] [20], [31], [34].

In conclusion, the gradient descent algorithm works iteratively by random weights and zero thresholds initialization, re-evaluating them using the inferred learning rules resulting in updates to $w_{mn} = w_{mn} + \delta w_{mn}$ and $\theta_m = \theta_m + \delta\theta_m$ with the learning rate $\eta > 0$ as a tuning parameter determining the size of each iteration for the entire training set [20]. Commonly termed, **batch gradient descent**, it cycles through the entire training set per iteration or **epoch**, which affects its convergence rate, in addition to being pruned to getting trapped in shallow local minimums or saddle points [34].

In consequence, there are different adaptations of gradient descent following a trade-off between the accuracy and consistency of the parameter update and the computational time of performing an update.

2.8.1.1 Stochastic Gradient Descent

Stochastic gradient descent is one of such variations that contrastingly performs a parameter update for each training pattern. Ignoring redundant computations, stochastic gradient descent converges swiftly.

2.8.1.2 Mini Batch Gradient Descent

Eventually, taking the best of both worlds, **mini batch gradient descent** performs a parameter update for every p number of training patterns [20], [33].

2.8.2 Adaptation of the Learning Rate

In light of all these, convergence is still not guaranteed, and these optimization algorithms suffer from convergence rates due to misadjusted learning rates, which can be resolved using an adaptive learning rule.

$$\delta w_{mn}^{(t)} = -\eta \frac{\partial L}{\partial w_{mn}}\bigg|_{\{w_{ij}\}=\{w_{ij}^{(t)}\}} + \alpha \delta w_{mn}^{(t-1)} \quad for \quad t = 1, 2, ..., T \tag{2.22}$$

Using an adaptive learning rule, the increment at iteration t depends not only on the instantaneous gradient but also on the weight changes $\delta w_{mn}^{(t-1)}$ of the previous iteration $t-1$. This causes inertial dynamics and accelerates the algorithm by adding $0 \leq \alpha < 1$ as the **momentum** constant. Momentum increases for parameter

[5] If the derivatives are small, the gradients will decrease exponentially, and as they are propagated forward, lead to the vanishing gradient problem. On the other hand, if the derivatives are large, the gradients will increase exponentially, and as they are propagated backward lead to the exploding gradient problem. Historically, LSTM (see Section 2.7.2.2) was proposed to resolve those problems equivalently in conjunction with an appropriate gradient-based algorithm [27].

dimensions whose gradients point in the same direction, as simultaneously reducing updates for parameter dimensions whose gradients change directions [20], [33].

2.8.2.1 Adam

A commonly used adaptive learning rule is the **adaptive moment estimation** or **Adam**, an algorithm for first-order gradient-based optimization of stochastic scalar objective functions $f_1(\theta), ..., f_T(\theta)$ at subsequent time steps $1, ..., T$ differentiable with respect to parameter θ. Adam succeeds in helping accelerate the optimization algorithm by controlling not only the exponentially decaying moving averages of the gradients m_t but also the exponentially decaying moving averages of the squared gradients v_t by hyperparameters $\beta_1, \beta_2 \in [0, 1)$.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad and \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.23}$$

With $g_t = \nabla_\theta f_t(\theta)$ as the gradient of f_t evaluated at time step t with respect to θ.

The moving averages are estimates of the 1^{st} moment (the mean) and the 2^{nd} raw moment (the uncentered variance) of the gradients, which are initialized as vectors of zeros, resulting in bias-corrected estimates

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad and \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{2.24}$$

Adam's update rules read

$$\alpha_t = \alpha \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \quad and \quad \theta_t = \theta_{t-1} - \frac{\alpha_t}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.25}$$

with α as step size, making Adam a favorable algorithm in practice [33], [35].

2.9 Sequence-to-Sequence Model

A **sequence-to-sequence** model or simply **seq2seq** takes sequences from a source domain and transforms them to sequences in a target domain through an **encoder-decoder** architecture.
The encoder-decoder architecture consists of two pivotal components: (1) Encoder and (2) Decoder. The encoding stage produces a fixed-length vector representation of the input sequence $x = (x_1, ..., x_T)$, followed by the decoding stage, creating the sequence output $y = (y_1, ..., y_{T'})$ from the vector. Equally, the encoder and decoder can be equipped with various internal architectures [6]; however, primarily, they consist of multilayered recurrent neural networks due to their ability to learn

the data with long temporal dependencies successfully [36].

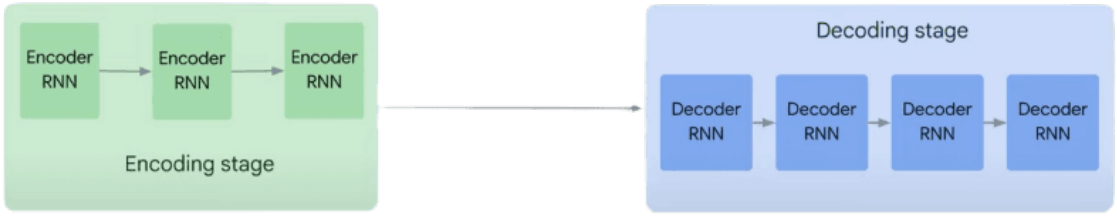


Figure 2.9: Encoder-decoder architecture equipped with multilayered recurrent neural networks. A great advantage of such a model is its ability to map uncorrelated sequence pairs of different lengths (T and T' may differ), which is illustrated above with different numbers of states in the stages.[7]

Taking a token of the input sequence at a time, the encoder assigns it to a recurrent neural network, which computes a hidden state h_t at time step t representing the token with respect to the previous hidden state h_{t-1} and all the previously ingested tokens and propagates it forward.

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \tag{2.26}$$

Eventually, encapsulating the information of all tokens of the input sequence, h_t lays a solid foundation for generating the fixed-dimensional vector representation c, which is passed to the decoder, acting as the initial hidden state. Following the same procedure

$$h_t = f(W^{(hh)}h_{t-1}) \tag{2.27}$$

$$y_t = softmax(W^{(s)}h_t) \tag{2.28}$$

are computed to predict output token y_t at time step t with the assistance of softmax units determining the output sequence.

The training is performed with a gradient-based algorithm and the ultimate goal of minimizing the cross-entropy loss function (or maximizing the conditional log-likelihood)

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, J(\Theta), \quad where \quad J(\Theta) = -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(y_n|x_n) \tag{2.29}$$

[6]Note that the concept emerges from "Sequence to Sequence Learning with Neural Networks" and "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" papers published in a small window of time from each other in the machine translation domain. While sharing similarities in the approach, they differ in the choice of internal architectures and enhance the use of LSTMs (see Section 2.7.2.2) and recurrent neural networks (see Section 2.7.2), respectively.

[7]Google Cloud Tech. (2023). Encoder-decoder architecture: Overview.

measuring the difference between the encoder-decoder architecture's predicted distribution and the true distribution of the output sequence in the target domain [37].

2.10 Attention is all you need [8]

Imagine arriving at the airport to catch a plane. Looking at the departure sign with an alert mind, you scroll through the list to find your flight without being distracted by the flow of travelers or all the surrounding advertisements. Heading straight to the designated check-in counter, you unexpectedly run into a friend in the crowd. Throughout this, the human brain goes through crucial states of attention, including vigilance and alertness, selection and distraction, orientation and filtering. While processing vast amounts of data, cognitive attention selectively considers a higher priority to certain received input or sensory information and invades consciousness whilst filtering out irrelevant information aiming to resolve information saturation [38].

The aforementioned example makes a great connection to the **attention** mechanism which comes to rescue the memory constraints of the encoder-decoder architecture, which are limited across long sequential lengths. The attention mechanism allows an encoder-decoder architecture to focus on specific tokens of an input sequence.

2.10.1 Additive Attention

Unlike the standard encoder-decoder architecture, in the **additive attention** mechanism, the encoder takes a token of the input sequence $x = (x_1, ..., x_{T_x})$ at a time and assigns it to a bidirectional neural network (see Section 2.7.2.1). Reading the input sequence in its natural order, the underlying forward recurrent neural network computes a sequence of forward hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_{T_x})$; however, the underlying backward recurrent neural network reads the input sequence in the reverse order and computes a sequence of backward hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_{T_x})$. Each annotation h_j representing a token x_j is derived from concatenating the corresponding forward and backward hidden states

$$h_j = [\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top]. \tag{2.30}$$

This allows the annotation h_j to summarize not only the preceding tokens but also the following tokens.

Contrary to the encoder-decoder architecture where the encoder is forced to encapsulate all the information of input sequence into a fixed-length context vector c, in the additive attention mechanism, the context vector is computed by the sequence of the encoder hidden states $(h_1, ..., h_{T_x})$

[8] The title is inspired by Ashish Vaswani, Noam Shazeer, Niki Parmer, Jacob Uszkoreit, Llion Jones, Aidan N. Gomez, Lucas Kaiser, and Illia Polosukhin's work and dedication at Google Brain and Google Research to "Attention is all you need" [23].

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.31}$$

where each annotation h_j contains the information of the entire input sequence with the focus on the surrounding tokens of x_j.

All hidden states are retained and utilized, and then passed to the decoder where the hidden state s_i at time step i is computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2.32}$$

to be fed to

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.33}$$

as an alignment model.

The alignment model a is a parametric model jointly trained as a feedforward neural network (see Section 2.4a) with respect to the gradient of the same loss function (see equation 2.29) allowing the weight α_{ij} of each annotation h_j to be computed by

$$\alpha_{ij} = softmax(e_{ij}). \tag{2.34}$$

Explicitly, α_{ij} is the probability of y_i being aligned to x_j concerning the conditional probability $p(y_i|y1, ..., y_{i-1}, x) = g(y_{i-1}, s_i, c_i)$ for each token y_i in the output sequence. Nevertheless, the decoder amplifies hidden states with higher alignment scores while downsizing hidden states with lower alignment scores by implementing the additive attention mechanism [39].

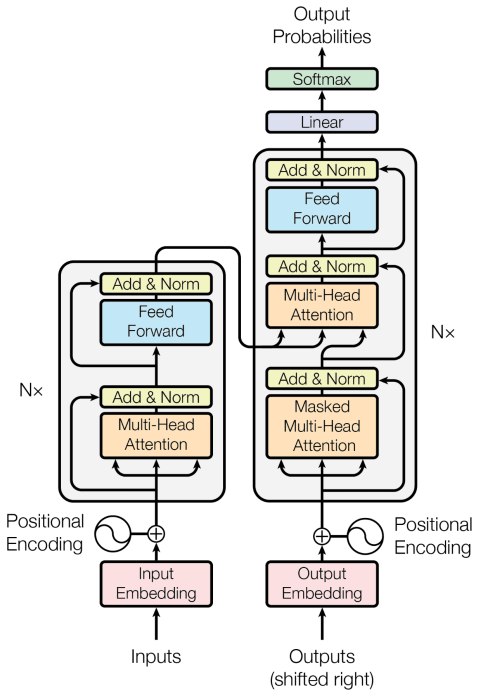


Figure 2.10: The Transformer-model architecture will be explained in detail as follows [23].

2.10.2 Scaled Dot-Product Attention

In the model architecture shown in Figure 2.10, the encoder maps an input sequence $x = (x_1, ..., x_n))$ to a sequence of continues representations $z = (z_1, ..., z_n)$. The decoder attempts to generate an output sequence $y = (y_1, ..., y_m)$ at a time given z. To possess such an establishment, the encoder and decoder are equipped with stacked self-attention and point-wise, fully connected layers.

Self-attention or intra-attention is an attention mechanism evolved to compute a representation of an input sequence by relating different positions of the same sequence, which captures long-term dependencies among the tokens.

On that account, the attention function can be described as mapping a query q and a set of key-value pairs (k, v) to an output:

- Q is a matrix where q is the d_k dimensional vector representation of the token for which self-attention is computed.
- K is a matrix where k is the d_k dimensional vector representation of a token that q's associated token is compared to.
- V is a matrix where v is the d_v dimensional contextualized vector representation of the token.

Matrices Q, K, and V are obtained by self-attention utilizing weight matrices $W^{(Q)}$, $W^{(K)}$, and $W^{(V)}$ as parameters adjusted through training and performing a matrix

multiplication with the embedded input sequence. Computing the attention function, the dot product of $Q\,K^\top$ is calculated and normalized by $\sqrt{d_k}$. A softmax function is applied to accumulate the weights on the values.

The **scaled dot-product attention** reads

$$Attention(Q, K, V) = softmax(\frac{Q\,K^\top}{\sqrt{d_k}})V. \tag{2.35}$$

The scaled dot-product attention is in many ways similar to additive attention, with the exception of the scaled dot-product being space-efficient and having a higher speed [23].

2.10.3 Multi-Head Attention

In **multi-head attention** mechanism, the queries, keys, and values are linearly projected h times with different learned linear projections to d_k, d_k, and d_v dimensions, respectively. Each of these projected versions is then utilized in performing the attention function in parallel, resulting in d_v dimensional output values. The final values are then obtained by

$$head_i = Attention(Q\,W_i^{(Q)}, K\,W_i^{(K)}, V\,W_i^{(V)}) \tag{2.36}$$

where the projections are parameter matrices

- $W_i^{(Q)} \in \mathbb{R}^{d_{model} \times d_k}$
- $W_i^{(K)} \in \mathbb{R}^{d_{model} \times d_k}$
- $W_i^{(V)} \in \mathbb{R}^{d_{model} \times d_v}$
- $W_i^{(O)} \in \mathbb{R}^{hd_v \times dmodel}$

used for the output transformation of size d_{model}.

The multi-head attention is obtained as

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^{(O)}. \tag{2.37}$$

This mechanism enables the model to jointly attend to information from different representation subspaces at different positions exhibiting a similar computational cost to a single attention head [23].

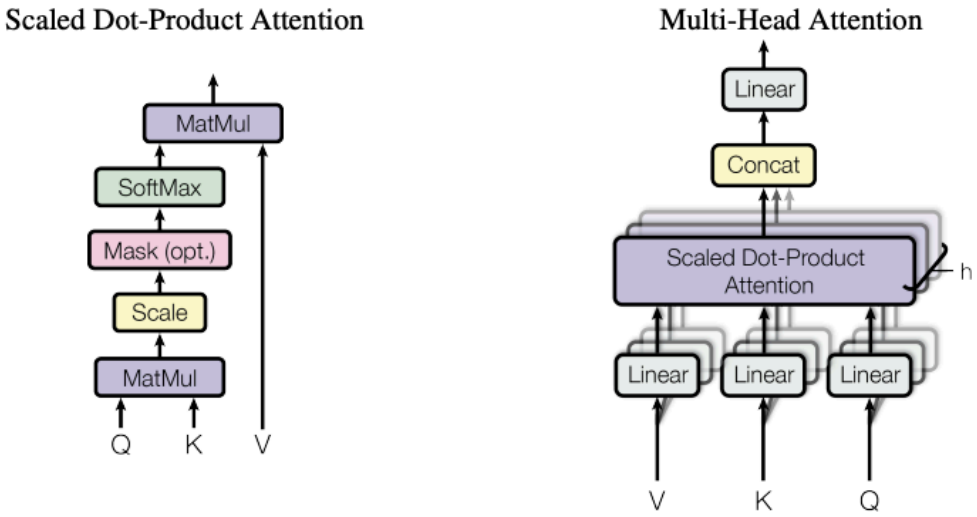


Figure 2.11: Scaled dot-product attention and multi-head attention consisting of several attention layers running in parallel [23].

2.10.4 Masked Attention

In the model architecture shown in Figure 2.10 and explicitly defined as an underlying argument in the context of scaled dot-product attention, the decoder is an auto-regressive model. In other words, the decoder generates the output token one at a time by consuming the previously generated tokens as additional input.

To preserve this auto-regressive property, tokens in a sequence can only attend to previous tokens and not future ones. Masking prevents attention to certain tokens, specifying which queries can attend to which keys. This is implemented within the structure of any aforementioned attention mechanisms (see Figure 2.11) by masking out (setting to $-\infty$) all values of the input of the softmax corresponding to illegal connections preventing leftward information flow [23].

2.11 Positional Encoding

As language is a system for associating form and meaning consisting of grammar and vocabulary, the location or position of words plays a crucial part in establishing the grammatical structure and delivering the semantics.

Positional encoding is the finite-dimensional representation of the location or position of tokens belonging to a sequence.

Various variants of positional encoding have emerged over the long haul, which is either a fixed statistical approach or can be learned as a positional embedding layer [40].

The fixed statistical approach is based on trigonometric functions wherein the most common case, sine, and cosine functions of different frequencies are employed. Considering this, as the frequencies of the sinusoidal functions are modified, the positional encoding will be assigned different values for every position.

With *pos* as the position and i as the dimension varying from 1 to d_{model}, where each dimension corresponds to a sinusoid, the positional encoding suggested by [23] read

$$PE_{(pos,2\mathrm{i})} = \sin(\frac{pos}{10000^{\frac{2\mathrm{i}}{d_{model}}}}) \tag{2.38}$$

and

$$PE_{(pos,2\mathrm{i}+1)} = \cos(\frac{pos}{10000^{\frac{2\mathrm{i}}{d_{model}}}}). \tag{2.39}$$

The wavelengths form a geometric progression from 2π to 10000.2π, allowing a model (see Section 2.12) to learn to attend to relative positions. This choice of positional encoding is concretely selected in [23] as the underlying model has no recurrence or convolution.

2.12 Transformer

Transformer is a model with an attention mechanism as an integral component, allowing it to draw global dependencies among input-output pairs without relying on recurrence.

Previously introduced in the concept of attention mechanism (see Section 2.10), the Transformer model architecture consists of an encoder and decoder equipped with stacked self-attention and point-wise fully connected layers.

A vital strength of the Transformer architecture compared to the standard encoder-decoder model or its connection through the attention mechanism is its ability to process input sequences in parallel, significantly accelerating training and inference.

As shown in Figure 2.10, the decoder-encoder architecture consists of $N\times$ identical layers each. There exist two sub-layers per layer in the encoder, which are multi-head attention mechanism and position-wise fully connected feedforward neural network built upon a residual connection [9] followed by layer normalization. The multi-head attention mechanism allows the model to capture global dependencies in parallel, while the position-wise feedforward network adds depth and importance to local features. This is also evident in the approach to how queries, keys, and values are employed in the self-attention sub-layers. Sharing a similar structure with the encoder, the decoder possesses an additional masked multi-head attention.

[9] Residual connections are connections that skip layers expediting the gradient flow and mitigating vanishing gradient problem

In the absence of any recurrence or convolution, the model adds the fixed positional encoding proposed in section 2.11 to the learned input embeddings. Although learned positional embeddings are also part of this experiment, it is claimed, apart from identical results, that PE_{pos+k} can be represented as a linear function of PE_{pos} for any fixed offset k allowing the model to extrapolate to longer sequential lengths.

The silver lining is that the positional encodings, input embeddings, and output embeddings all share the same dimension d_{model}. This not only facilitates the aforementioned residual connections the encoder-decoder stacks are built upon but also, in combination with output embeddings being offset by one position and masked multi-head attention, ensures decoder predictions at any position are dependent on preceding outputs.

As the input-output pairs are processed and provided to the encoder-decoder architecture, the queries come from the previous decoder layer. In contrast, the key-value pairs come from the encoder output all over the attention layers.

In the self-attention sub-layers of the encoder and decoder, queries, keys, and values come from the previous layer, which enables each position to attend to all positions except for masking in the decoder.

The fully connected feedforward network at each sub-layer is applied to each position separately and identically by

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.40}$$

where weights W_i and thresholds b_i as parameters differ from layer to layer [10].

Eventually, softmax units are operated on the decoder output using the same weight matrix for the embedding layers to predict the probability of following tokens [23].

2.13 Transfer Learning

Playing a musical instrument requires in-depth knowledge. Imagine a scenario where two individuals attempt to learn how to play the piano. While one person has significant musical knowledge through playing the violin, the other person completely lacks any musical knowledge. As the skilled violinist intend to learn how to play the piano, they can transfer previously acquired knowledge to be able to learn more smoothly.

The non-technical example of acquiring music knowledge can be expanded to a machine learning application where **transfer learning** involves transferring previously acquired knowledge from a related source domain D to a target domain T.

[10] ReLU activation function is adopted in this case. Introduced by Vinod Nair and Geoffrey E. Hinton at the University of Toronto in 2010 as $max(0, x)$, the rectified linear units revolutionized the training of deep neural networks as they add a non-linearity essence, allowing complex patterns to be captured.

Recall the task τ_S as $\tau_S = \{x_i, y_i\}_{i=1}^n$ defined for a given domain D with feature space χ and marginal probability distribution $P(X)$ (see Section 2.2.1). The source domain is defined as

$$D_S = \{(x_{S1}, y_{S1}), ..., (x_{Sn}, y_{Sn})\}. \tag{2.41}$$

Similarly, the target domain is defined as

$$D_T = \{(x_{T1}, y_{T1}), ..., (x_{Tn}, y_{Tn})\}. \tag{2.42}$$

The predictive functions for the source and target domain are f_S and f_T, respectively. According to that supposition, transfer learning is the process of improving the target predictive function f_T by utilizing the related information from D_S and T_S, where $D_S \neq D_T$ or $\tau_S \neq \tau_T$ [41].

A significant application of transfer learning is cases where training is computationally costly or sufficient quality data [11] is not at hand.

2.14 Pre-Trained Language Models

Transfer learning succeeds where traditional machine learning fails. Transfer learning's capability to address key challenges of traditional machine learning, including insufficient labeled data, incompatible computational power, and distributional mismatch, puts it forward as a great candidate to be applied to a wide range of problems such as natural language processing [42].

In the case of natural language processing, as the model architecture becomes more complex, the number of parameters that must be captured increases exponentially. Aside from the problems these models are pruned to, providing them with a massive amount of quality data is costly. Inspired by humans' capability to use prior knowledge to solve new problems, transfer learning aims to solve the target task by leveraging the knowledge learned from source tasks in different domains, which demonstrates significant performance in natural language processing applications. That is owing to the fact that a pre-trained model can learn universal language representations and provide a neat initialization leading to a better generalization performance and higher convergence rate on the downstream task. Typically, pre-trained language models are trained on a huge corpus of data to minimize the cross-entropy loss function with a probability distribution over the vocabulary given linguistic context for a pre-training task [43].

A common pre-training task is **M**asked **L**anguage **M**odeling, **MLM**, emerging from the theories proposed by Wilson L. Taylor in 1953 as Cloze Procedure. As a psychological tool for measuring the effectiveness of communication, the cloze procedure has a cloze unit at its heart. A cloze unit is any single occurrence of a successful

[11] Accuracy, completeness, validity, consistency, uniqueness, timeliness, and fitness for purpose are the criteria for data quality.

attempt to reproduce a deleted token from a sequence based on the context of the remaining tokens accurately [44]. Pre-trained language models with various encoder-decoder architectures are trained to predict randomly masked tokens given masked input sequences.

Primarily relying on the existing strategies for application on downstream tasks, language model pre-training has led to various language representation models. An existing strategy for applying pre-trained language representations to downstream tasks is fine-tuning, where task-specific parameters are trained on the downstream task to be adjusted or fine-tuned. However, there is a major drawback to this with standard language models being unidirectional where there is a limitation only allowing leftward information flow. While the Transformer (see Section 2.12), GPT-1 provides examples of such model architectures, BERT, RoBERTa, ALBERT, and CLIP rose to fame as they address this issue [4], [45]–[48].

2.15 BERT

Proposed as **B**idirectional **E**ncoder **R**epresentation from **T**ransformers, **BERT**, changed the game for natural language processing by alleviating the aforementioned unidirectionality constraint with an MLM pre-training objective.

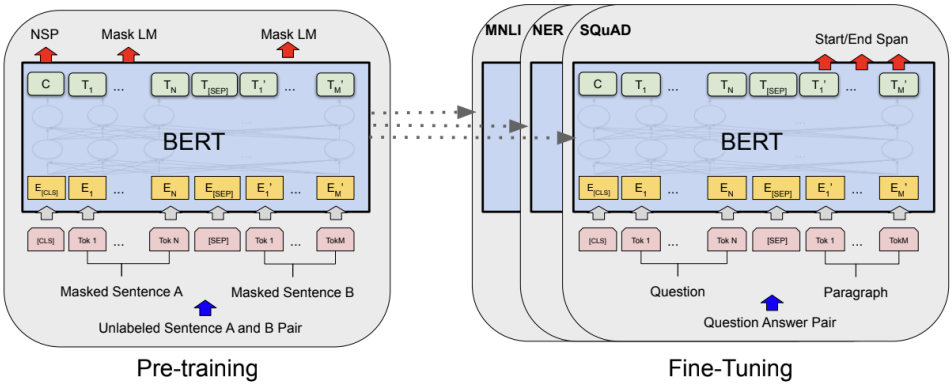


Figure 2.12: Although BERT had an MLM pre-training objective originally, the researchers, Jacob Delvin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova at Google AI Language, continue to demonstrate the model's ability by contributing to the "next sentence prediction" pre-training task additionally. "Next sentence prediction," or NSP, per the name suggests, trains the model to distinguish whether two input sentences are continuous segments. The overall pre-training and fine-tuning procedures, which share almost the same architecture except for the output layers, are shown [43], [46].

As a multi-layer bidirectional Transformer, BERT shares an identical model architecture to the Transformer (see Section 2.12). Primarily, the model was introduced in two sizes. Assuming L as the number of layers or Transformer blocks, H as the hidden size, and A as the number of self-attention heads, the two model sizes are:

- **BERT$_{\text{BASE}}$** consisting of $L = 12$, $H = 768$, $A = 12$ and 110M total parameters.

- **BERT$_{\text{LARGE}}$** consisting of $L = 24$, $H = 1024$, $A = 16$ and 340M total parameters.

Apart from the model architecture, BERT owes its pioneering success to the input-output representation and masking strategy.

To represent input-output sequences or pairs, WordPiece embeddings are used, where WordPiece is a sub-word tokenization model generated using a data-driven approach to maximize the language-model likelihood of the training corpus (equivalently, minimizing the cross-entropy loss), given an evolving word definition. Given a training corpus and a set of tokens D, the optimization problem is to select D wordpieces where the vocabulary is minimal in the number of wordpieces when segmented according to the chosen WordPiece model. Recall the sentence provided as an example in section 2.4. Applying the WordPiece on the sentence, the tokenized sentence reads

Steve pelted the squirrels with acorns.

```
[Steve, pe, _lt, _ed, the, squirrels, with, ac, _orn, _s, .]
```

Table 2.2: A randomly chosen sentence from the vision-and-language dataset (see Section 3.1.2) is tokenized using the WordPiece model. It can be observed that the words 'Steve', 'the', 'squirrels', 'with', and '.' are not wordpieces. While they are part of the model's vocabulary, the remaining words are not; hence, using a greedy optimization algorithm, they are tokenized into wordpieces that can be described by the vocabulary. Marked with a special word boundary system '_', the character indicates the boundaries between tokens ensuring the recovery of the original sequence without ambiguity. It is worth noting that the model was originally developed by Google to be applied to Japanese and Korean [49], [50].

Eventually, the input representation for a given token is constructed by summing the corresponding token, segment, and position embeddings.

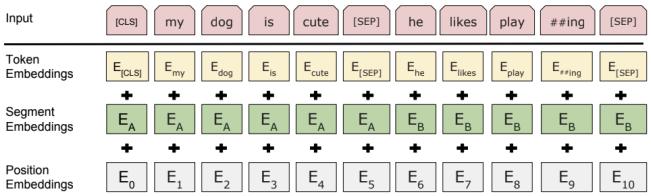


Figure 2.13: In [46], the researchers offer a practical illustration of their input representation pre-processing approach using the example sentence: "my dog is cute. he likes playing." It can be seen how the WordPiece model has tokenized the sentence. Moreover, special tokens [**CLS**] and [**SEP**] are introduced, playing a pivotal role in the model architecture. The [**CLS**] token marks the inception of a sequence, while [**SEP**] serves as a sentence separator. The final hidden state of the special [**CLS**] token is used as the aggregate sequence representation for classification tasks as $C \in \mathbb{R}^H$. Summarizing the notation, input embeddings are represented by E, and the final hidden vector for the i^{th} toke is represented by $T_i \in \mathbb{R}^H$. At last, segment embeddings come to rescue differentiating sentences.

When pre-training BERT with an MLM pre-training task, a percentage of the input tokens are randomly masked. Then, to obtain a bidirectional pre-trained model, only the masked token is predicted rather than reconstructing the entire input sequence. In spite of being an effective strategy, this creates a mismatch between the pre-training and fine-tuning phases, where the [**MASK**] token is absent during the fine-tuning. To resolve this, a procedure is suggested where 15% of the tokens are randomly masked; however,

- 80% of the time, the masked token is replaced with the actual [**MASK**] token
- 10% of the time, the masked token is replaced with a random token
- 10% of the time, the masked token is replaced with the unchanged token

allowing T_i to be used for prediction minimizing the cross-entropy loss.

After pre-training on unlabeled data over the pre-training task, BERT is initialized with the pre-trained parameters to be fine-tuned on the labeled data from the downstream task.

3

Methods

This chapter introduces and describes the methods used throughout this thesis.

3.1 Data

This study aims to investigate, measure, and mitigate catastrophic forgetting in the domain-specific task of visual-to-textual knowledge transfer, which necessitates adequate data for each method. Consequently, a detailed description of the data utilized throughout this process is provided.

3.1.1 Memory Colors Dataset

Human memory plays a significant role in shaping color perception. Human observers often consider characteristics for an object type, reflecting their knowledge of that object's typical color. This phenomenon, known as memory color, refers to the canonical or typical color that human observers associate with an object type depending on their experiences. Grass, cherry, snow, and cinnamon are always associated with the colors green, red, white, and brown, respectively [51].

In [3], the concept of memory color is endorsed and utilized to evaluate the transfer of visual knowledge to text. This is substantiated by empirical evidence demonstrating that human cognition allows retaining memory colors associated with specific objects. This knowledge enables individuals to respond to inquiries concerning these objects without direct visual cues.

The Memory Color dataset comprises 109 object types in the English language, where each object type in the dataset is paired with a descriptor, the associated memory color, and an illustrating picture. The object types in this experiment are real-world, well-known entities meticulously chosen and verified based on an annotator agreement by the authors to design a dataset tailored to the particular objective.

In Figure 3.1, an entry in the Memory Color dataset can contribute to a more profound understanding. The descriptors are chosen manually to satisfy grammatical accuracy and prevent item reference ambiguities, while the illustrating pictures are selected to describe the object type and the associated memory color visually.

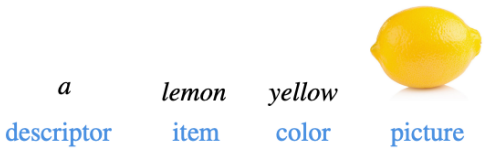


Figure 3.1: An entry in the Memory Color dataset.

The memory colors used for the object types in this experiment are black, blue, brown, green, grey, orange, pink, purple, red, white, and yellow (11 color categories), such that high requirements of admitting an agreement between human observers are fulfilled. It is worth noting that although this dataset is not used directly in this thesis, it provides greater clarity regarding the aims this thesis is trying to establish and possibly continue in further extensions (see section 1.2).

3.1.2 Vision-and-Language Dataset

The vision-and-language dataset lies at the core of this work. The dataset is constructed in [3] by combining four controlled public image+text datasets, including MS COCO [52], SBU Captions [53], visual Genome [54], and Conceptual Captions [55].

The preliminary investigations to identify the phenomenon of catastrophic forgetting in this study are limited to the captions of the vision-and-language dataset, and they can be extended to multi-modal models in later stages (see section 1.2). It contains 7,984,412 captions split into training (7,869,235) and validation (115,177). For demonstration, consider the following caption from the dataset:

"Steve pelted the squirrels with acorns."

This is a random entry from the dataset to provide a more profound understanding of the captions' structure and content in the vision-and-language dataset[1].

3.1.3 Pre-Training Data

Pre-training data resources are used to mitigate catastrophic forgetting in this domain-specific task. The pre-training data resources are based on the pre-training

[1]There are more captions than images in the dataset since several different captions may refer to the same image. For instance, the MS COCO large-scale dataset contains images of complex everyday scenes where each image is paired with five written caption descriptions [52]. Moreover, some image links in SBU Captions and Conceptual Captions have become broken; therefore, the total number of samples does not match what was originally reported. It is worth noting that the self-supervised training task at the core of [3] is unfortunately not at the center of focus in this study; however, in an attempt to ensure correct predictions with higher confidence to be originated from the visual modality, the text captions are filtered with different outlooks by the authors. Thus, they provide a detailed description of the total number of image and caption samples in each respective source dataset, which can be of interest to the curious minds of the readers.

procedure of BERT (see section 2.15) which follows the existing literature on language model pre-training including BookCorpus[2] [56] and English Wikipedia[3].

The BookCorpus dataset contains 74,004,228 plain text sentences from 11,038 books collected by the authors of [56]. For English Wikipedia, only the cleaned articles or text passages are used.

3.2 Metrics

Evaluation of language models is a sensitive task that requires employing suitable metrics based on the downstream task to assess the performance, providing quantitative and qualitative measures.

Human baseline is a qualitative measure involving human annotators to evaluate the quality of a language model's output in the domain-specific task. This assessment includes linguistic aspects such as fluency, coherence, and relevance, allowing the nuances of the language to be captured. It is essential to gauge how well a language model performs compared to human-generated language.

Confusion matrix is a common measure utilized to inspect the performance of such a model quantitatively. Considering a binary classification problem, the dataset can be separated into four groups based on the actual y and predicted $\hat{y}(x)$ outputs. The confusion matrix for such a model is shown in Table 3.1 [10].

TP, TN, FP, and FN are central components of the confusion matrix, providing an informative overview of the model's performance. TP and TN represent the number of positive and negative instances that are correctly classified, whereas FP and FN represent the number of negative and positive misclassified instances [57].

The relevance of each of these four quantities depends on the problem the downstream task is aiming to solve and motivates the choice of metrics; however, it is crucial to distinguish the trade-off between FP (Type I error) and FN (Type II error). Primarily, this lays a foundation for metrics to be calculated from the tabulated information, allowing a more detailed analysis [58], which can also be generalized to multi-class cases.

[2]https://huggingface.co/datasets/bookcorpus
[3]https://huggingface.co/datasets/wikipedia

	$y = 1$	$y = -1$	total
$\hat{y}(x) = 1$	**T**rue **P**ositive (TP)	**F**alse **N**egative (FN)	P^*
$\hat{y}(x) = -1$	**F**alse **P**ositive (FP)	**T**rue **N**egative (TN)	N^*
total	P	N	n

Table 3.1: The confusion matrix shows the number of true and false predictions obtained by the model given the dataset. P and N indicate the number of positive and negative instances in the dataset, respectively, whereas P^* and N^* indicate the number of positive and negative predictions by the model [10].

3.2.1 Accuracy

Accuracy measures the proportion of correctly predicted instances TP and TN to the total population n.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{n} \tag{3.1}$$

3.2.2 Precision

Precision measures the proportions of correctly identified positive instances TP to the total number of instances predicted as positive P^*.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{P^*} \tag{3.2}$$

Accuracy and precision are distinct concepts in measurement. Accuracy quantifies how close a measurement is to the true label. On the other hand, precision assesses the consistency of measurements, emphasizing how close individual predictions are together.

3.2.3 Recall

Recall measure the proportions of positive instances that are correctly predicted as positive.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{3.3}$$

Precision and recall are quantitative values between 0 and 1, where a higher value indicates a better model.

3.2.4 F1-Score

F1-score is the harmonic mean of precision and recall.

$$F1 = \frac{2.precision.recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3.4}$$

Similarly, it is a quantitative value between 0 and 1, where a higher value indicates a better model.

3.2.5 Matthew's Correlation Coefficient

Matthew's Correlation Coefficient is a metric used to measure the quality of a binary classifier. It provides a balanced assessment of a model's performance by considering correct and incorrect predictions in a single value.

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.5}$$

The MCC score ranges from -1 to 1, where 1 signifies a perfect model, 0 represents a random model, and -1 implies a perfect disagreement between the model's predictions and the true values.

3.2.6 Pearson Correlation Coefficient

Pearson Correlation Coefficient measures the linear correlation between two sets of data. Given random variables X and Y, it represents a normalized measurement of the covariance as it calculates the ratio between the covariance and the product of their standard deviations.

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{3.6}$$

The correlation coefficient ranges from -1 to 1, where 1 implies a perfect positive linear correlation, 0 represents no linear dependency, and -1 indicates a negative linear correlation between the variables.

3.2.7 Spearman's Rank Correlation Coefficient

The **Spearman's Rank Correlation Coefficient** is a non-parametric measure of rank correlation. It estimates how well a monotonic function can describe the relationship between two variables X and Y can be described using a monotonic function. That said, the Spearman correlation between the two variables is equal to the Pearson correlation between the ranked variables $R(X)$ and $R(Y)$.

$$r_s = \rho_{R(X),R(Y)} = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}} \tag{3.7}$$

The correlation coefficient ranges from -1 to 1, where 1 indicates a perfect positive correlation, 0 implies no correlation, and -1 specifies a perfect negative correlation.

It is worth noting that in the case of natural language processing tasks, the Pearson Correlation Coefficient and Spearman's Rank Correlation Coefficient must be interpreted as quantitative measures showing the correlation between the human baseline and the model's prediction.

3.3 Benchmark

Benchmark refers to standardized datasets, measures, and baselines providing a means to evaluate the performance of language models objectively. For language models to be effective, they must be able to process language in a way that is not exclusive to a single task or dataset. Hence, benchmarks are necessary tools to track progress and set a standard for assessing the performance of different language models agreed upon by the communities.

3.3.1 Glue

To pursue the aforementioned objective [59] presented **G**eneral **L**anguage **U**nderstanding **E**valuation, **GLUE** benchmark. GLUE is a collection of nine English natural language understanding tasks favoring language models that facilitate sample-efficient learning and effective knowledge transfer to push their abilities beyond their architecture and represent meaningful linguistic solution strategies.

GLUE relies on existing datasets agreed upon by the natural language processing community, which, in several cases, contain private test sets to ensure a fair assessment of the models' performance. Moreover, it provides an associated online platform[4] for model evaluation, comparison, and analysis, built upon the designed datasets, challenging the models to execute linguistic tasks more robustly.

GLUE is designed with the ultimate goal of enhancing the development of generalized natural language understanding systems. Therefore, a model is required to share substantial knowledge across all tasks to showcase a good performance. The tasks are outlined as follows and are balanced across classes while being evaluated with the suitable choice of metric.

[4] https://gluebenchmark.com/

Corpus	|Train|	|Test|	Task	Metrics	Domain
Single-Sentence Tasks					
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews.
Similarity and Paraphrase Tasks					
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
Inference Tasks					
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	conference/NLI	acc.	fiction books

Table 3.2: An overview of GLUE's task descriptions and statistics. Test sets shown in bold contain private labels.

3.3.1.1 Single-Sentence Tasks

CoLA The Corpus of Linguistic Acceptability contains English acceptability judgments collected from books and journal articles on linguistic theory. As a binary classification task, the goal is to evaluate whether a given sentence is grammatically acceptable [60].

SST-2 The Stanford Sentiment Treebank contains sentences from movie reviews and human annotations of their sentiment. The task involves classifying the sentiment of a given sentence as either positive or negative [61].

3.3.1.2 Similarity and Paraphrase Tasks

MRPC The Microsoft Research Paraphrase Corpus is a corpus of sentence pairs automatically extracted from online news sources with human annotations. The task is to identify whether two sentences are paraphrased of each other or are semantically equivalent [62].

QQP The Quora Question Pairs dataset is a collection of question pairs. The task is determining whether a pair of questions from the community question-answering website Quora are semantically equivalent.

STS-B The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines, video and image captions, and natural language infer-

ence data where each pair has a human-annotated similarity score from 1 to 5. Differing from the other tasks, which are either single sentence or sentence pair classification, this is a regression task that involves estimating how similar each pair is in terms of semantic meaning [63].

3.3.1.3 Inference Tasks

MNLI Multi-Genre Natural Language Inference is a crowdsourced collection of sentence pairs, each annotated with textual entailment information. In this multi-class classification task, every pair includes a premise and a hypothesis sentence. The primary objective is to ascertain whether the hypothesis sentence entails, contradicts, or remains neutral in relation to the premise sentence [64]. The private test set of MNLI is divided into matched (in-domain instances) and mismatched (out-of-domain instances) subsets depending on whether the domain of each test instance matches or resembles the training set domain [65].

QNLI Drawing inspirations from [66] and [67], the authors attempt to recast the Stanford Question Answering dataset into natural language inference. The Stanford Question Answering dataset is a question-answering dataset containing question-paragraph pairs, where the paragraphs are derived from Wikipedia, and the associated question is written by an annotator. For each question-paragraph pair, the paragraph contains a sentence that responds to the question. The conversion leads to the creation of the Question-answering Natural Language Inference, which involves a sentence pair classification task. Forming a pair between each question and each sentence of the corresponding paragraph and filtering out pairs with low lexical overlap lies at the heart of this process. This ignores the reliability of lexical overlap and eliminates the requirements for the precise answer to be selected and the assumption that the paragraph encompasses the answer. The modified task, therefore, is to determine whether the context sentence contains the answer to the question [68].

RTE The Recognizing Textual Entailment (RTE) benchmark, initiated by [69] under the PASCAL Network of Excellence for Recognizing Textual Entailment Challenge, aims to investigate the nuance of semantic expression in natural language processing. The challenge is held annually, focusing on textual entailment [66]. Given two fragments, the task is defined as recognizing whether the meaning of one text can be inferred from the other. In their efforts [59], construct a two-class split dataset by combining the data from RTE1 [69], RTE2 [70], RTE3 [71], and RTE5 [72]. Notably, examples were drawn from diverse sources, including news and Wikipedia. In three-class datasets, neutral and contradiction classes are collapsed into *not_entailment* for consistency.

WNLI The Winograd Schema Challenge or WSC is a reading comprehension task. A Winograd schema is a pair of sentences that differ only in one or two words, introducing a referential ambiguity resolved in opposite directions within the two sentences. This task is inherently challenging as it necessitates prior world knowledge and commonsense reasoning to address linguistic ambiguity and coreference resolution. The Winograd Natural Language Inference dataset or WNLI is created

by constructing sentence pairs in which an ambiguous pronoun is replaced by each possible referent. The task involves predicting whether the sentence with the pronoun substituted is entailed by the original sentence or not[73].

Coarse-Grained Categories	**Fine-Grained Categories**
Lexical Semantics	Lexical Entailment, Morphological Negation, Factivity, Symmetry/Collectivity, Redundancy, Named Entities, Quantifiers
Predicate-Argument Structure	Core Arguments, Prepositional Phrases, Ellipsis/Implicits, Anaphora/Coreference Active/Passive, Nominalization, Genitives/Partitives, Datives, Relative Clauses, Coordination Scope, Intersectivity, Restrictivity
Logic	Negation, Double Negation, Intervals/Numbers, Conjunction, Disjunction, Conditionals, Universal, Existential, Temporal, Upward Monotone, Downward Monotone, Non-Monotone
Knowledge	Common Sense, World Knowledge

Table 3.3: An overview of various linguistic phenomena annotated in the diagnostic dataset, organized under four major categories.

The benchmark, as summarized previously, only reflects an application-driven distribution of examples. Consequently, the authors include a diagnostic dataset for the analysis of system performance. The diagnostic dataset is a diverse test set constructed by producing examples for a variety of linguistic phenomena (see Table 3.3) based on naturally occurring sentences from various sources, including news, Reddit, Wikipedia, and academic papers.

Tags	**Sentence 1**	**Sentence 2**	**Fwd**	**Bwd**
Lexical Entailment (Lexical Semantics), Downward Monotone (Logic)	The timing of the meeting has not been set, according to a Starbucks spokesperson.	The timing of the meeting has not been considered, according to a Starbucks spokesperson.	N	E
Universal Quantifiers (Logic)	Our deepest sympathies are with all those affected by this accident.	Our deepest sympathies are with a victim who was affected by this accident.	E	N
Quantifiers (Lexical Semantics), Double Negation (Logic)	I have never seen a hummingbird not flying.	I have never seen a hummingbird.	N	E

Table 3.4: Examples from the diagnostic set. Fwd (resp. Bwd) denotes the label when sentence 1 (resp. sentence 2) is the premise. Annotated labels include entailment (E), neutral (N), or contradiction (C). Examples are tagged with the phenomena they demonstrate, and each phenomenon belongs to one of four major categories (in parentheses).

A sample of the diagnostic dataset is shown in Table 3.4. This dataset is evaluated

using R_3, a three-class generalization of Matthew's correlation coefficient. Similar to the benchmark, human baseline performance is established on the set with an average R_3 score among several annotators [74]. The performance can be the basis of comparison between different models but not between categories. However, it is crucial to emphasize that it does not represent overall performance or generalization in downstream applications. Finally, the diagnostic dataset in GLUE is provided as an analysis tool for error analysis, qualitative model comparison, development of adversarial examples, and, importantly, it is not intended to serve as a benchmark [59].

As introduced at the beginning of this section, GLUE is primarily designed as a prominent evaluation framework of natural language understanding that covers a range of training data volumes, task genres, and task information. It has paved the way for models like OpenAI GPT and BERT, exhibiting the potential of transfer learning. Despite providing a general-purpose evaluation of language understanding, GLUE surpasses human performance and particularly exceeds the human performance estimate on CoLA, MRPC, QQP, and QNLI tasks. This suggests limitations on further research since the benchmark is no longer a suitable metric for quantifying progress. Nevertheless, GLUE remains a cornerstone in the evaluation of language models, and there is substantial scope for improvement toward high-level goals set upon it [75].

3.3.2 SuperGLUE

SuperGLUE was introduced by [75] as a successor to GLUE, representing a pivotal advancement in the evaluation of natural language understanding. It was created as a natural language processing benchmark, satisfying the demand for posing a more rigorous challenge of natural language understanding.

SuperGLUE is a collection of eight language understanding tasks styled after GLUE with the same high-level motivation and basic design. It is drawn on existing data as it retains the two most challenging tasks, RTE and WSC, in GLUE. The task formats are expanded to include coreference resolution and question answering as well as sentence and sentence-pair classification. This incorporation of diverse task formats and the provision of robust human baselines guarantees a straightforward and dependable evaluation metric for assessing the effectiveness of any approach applicable to a wide range of language understanding tasks. Moreover, an inclusive informative leaderboard[5]with improved code support is provided, which signifies a great contribution to the natural language processing community to advance research purposes.

[5]https://super.gluebenchmark.com/

Corpus	|Train|	|Dev|	|Test|	Task	Metrics	Text Sources
BoolQ	9427	3270	3245	QA	acc.	Google queries, Wikipedia
CB	250	57	250	NLI	acc./F1	various
COPA	400	100	500	QA	acc.	blogs, photography encyclopedia
MultiRC	5100	953	1800	QA	$F1_a$/EM	various
ReCoRD	101k	10k	10k	QA	F1/EM	news(CNN, Daily Mail)
RTE	2500	278	300	NLI	acc.	news, Wikipedia
WiC	6000	638	1400	WSD	acc.	WordNet, VerbNet, Wiktionary
WSC	554	104	146	coref.	acc.	fiction books

Table 3.5: An overview of SuperGLUE's task descriptions and statistics. *WSD*, *NLI*, *coref.* and *QA* stand for word sense disambiguation, natural language inference, coreference resolution, and question answering, respectively. The number of total answers for MultiRC is listed for 456/83/166 train/dev/test questions.

In the design process of SuperGLUE, several criteria played a significant role. These criteria include task substance, task difficulty, evaluability, public data, task format, and license. To that end, each of these contributing factors are identified as the following:

Task substance: Tasks should test the capability of a system to understand and reason about texts in English.

Task difficulty: Tasks should push the boundaries of the scope of existing state-of-the-art systems yet be solvable by most college-educated English speakers. Additionally, tasks that require domain-specific knowledge are excluded.

Evaluability: Tasks should feature an automatic performance metric that aligns closely with human assessments of output quality.

Public data: Tasks should be drawn from existing public training data, undermining the risks associated with newly created datasets. Moreover, test sets with private labels are favored.

BoolQ	**Passage:** *Barq's – Barq's is an American soft drink. Its brand of root beer is notable for having caffeine. Barq's, created by Edward Barq and bottled since the turn of the 20th century, is owned by the Barq family but bottled by the Coca-Cola Company. It was known as Barq's Famous Olde Tyme Root Beer until 2012.* **Question:** *is barq's root beer a pepsi product* **Answer:** `No`
CB	**Text:** *B: And yet, uh, I we-, I hope to see employer based, you know, helping out. You know, child, uh, care centers at the place of employment and things like that, that will help out. A: Uh-huh. B: What do you think, do you think we are, setting a trend?* **Hypothesis:** *they are setting a trend* **Entailment:** `Unknown`
COPA	**Premise:** *My body cast a shadow over the grass.* **Question:** *What's the CAUSE for this?* **Alternative 1:** *The sun was rising.* **Alternative 2:** *The grass was cut.* **Correct Alternative:** `1`
MultiRC	**Paragraph:** *Susan wanted to have a birthday party. She called all of her friends. She has five friends. Her mom said that Susan can invite them all to the party. Her first friend could not go to the party because she was sick. Her second friend was going out of town. Her third friend was not so sure if her parents would let her. The fourth friend said maybe. The fifth friend could go to the party for sure. Susan was a little sad. On the day of the party, all five friends showed up. Each friend had a present for Susan. Susan was happy and sent each friend a thank you card the next week* **Question:** *Did Susan's sick friend recover?* **Candidate answers:** *Yes, she recovered* (`T`), *No* (`F`), *Yes* (`T`), *No, she didn't recover* (`F`), *Yes, she was at Susan's party* (`T`)
ReCoRD	**Paragraph:** *(CNN) Puerto Rico on Sunday overwhelmingly voted for statehood. But Congress, the only body that can approve new states, will ultimately decide whether the status of the US commonwealth changes. Ninety-seven percent of the votes in the nonbinding referendum favored statehood, an increase over the results of a 2012 referendum, official results from the State Electorcal Commission show. It was the fifth such vote on statehood. "Today, we the people of Puerto Rico are sending a strong and clear message to the US Congress ... and to the world ... claiming our equal rights as American citizens, Puerto Rico Gov. Ricardo Rossello said in a news release. @highlight Puerto Rico voted Sunday in favor of US statehood* **Query** For one, they can truthfully say, "Don't blame me, I didn't vote for them, " when discussing the <placeholder> presidency **Correct Entities:** `US`
RTE	**Text:** *Dana Reeve, the widow of the actor Christopher Reeve, has died of lung cancer at age 44, according to the Christopher Reeve Foundation.* **Hypothesis:** *Christopher Reeve had an accident.* **Entailment:** `False`
WiC	**Context 1:** *Room and board.* **Context 2:** *He nailed boards across the windows.* **Sense match:** `False`
WSC	**Text:** *Mark told Pete many lies about himself, which Pete included in his book. He should have been more truthful.* **Coreference:** `False`

Table 3.6: Specific examples from the development set of each task in SuperGLUE. **Bold** text represents part of the example format for each task. Text in *italics* is part of the model input. Underlined text is specially marked in the input. Text in a `monospaced font` represents the expected model output

Task format: To eliminate any constraints on the model architecture, tasks with relatively simple input-output format are prioritized. As previously highlighted, the scope of task formats is expanded to include coreference resolution and question answering, alongside single sentence and sentence-pair classification, which yields a set of tasks with longer inputs that require understanding individual tokens in context, complete sentences, inter-sentence relations, and entire paragraphs.

License: Task data should be available under licenses, allowing the use and redistribution for research purposes.

Taking these into account, the following eight tasks were selected for SuperGLUE, which can be seen in detail in Tables 3.5 and 3.6. The tasks are outlined below and

are balanced across classes while being evaluated with the suitable choice of metric.

BoolQ The Boolean Questions is a dataset containing a question, a paragraph from a Wikipedia article, the title of the article, and an answer, either yes or no. The questions are collected by heuristically identifying yes/no questions from anonymized, unsolicited queries to the Google search engine. The selected questions are subjected to a second filtering, sustained if only a Wikipedia page is returned as one of the first five results. Finally, the questions and Wikipedia pages are passed to human annotators for a strategic annotation process, where a passage from the article is paired with a question in response to yes or no. This reduces ambiguity and yields a dataset suitable for model applications [76].

CB The CommitmentBank comprises a sentence-pair classification task to assess a model's ability to understand and process commitment. The corpus contains short text examples, where at least one sentence contains an embedded clause. Each embedded clause is annotated with the degree to which it indicates the speaker's commitment to the truth of the clause. An example is shown in Table 3.6, which consists of a premise containing an embedded clause, and the corresponding hypothesis is the extraction of the clause. The task involves classifying the naturally occurring discourses drawn from several sources, including the Wall Street Journal, fiction from the British National Corpus, and Switchboard, into a three-class textual entailment [77].

COPA The Choice of Plausible Alternatives or COPA is a dataset designed to evaluate a model's ability to understand and process commonsense causal reasoning. The examples are manually selected from various resources, including blogs and a photography-related encyclopedia. The task is to determine the cause or effect of a given premise sentence from two possible choices [78].

MultiRC The Multi-Sentence Reading Comprehension is a question-answering dataset in which the question-paragraph pairs are crowdsourced across different domains to ensure linguistic diversity to the texts and the questions' wordings. According to Table 3.6, each example comprises a context paragraph, a question regarding the paragraph, and a set of possible answers. The underlying task involves evaluating a system's ability to predict whether a question is true or false based on reasoning over multiple context sentences. As a question-answering task, the selected evaluation metrics are $F1_a$ and EM. $F1_a$ is obtained by calculating the F1 score across all answer options, whereas EM determines the exact match for each question's set of answers [79].

ReCoRD The Reading Comprehension with Commonsense Reasoning Dataset is a multiple-choice question-answering task. Each example consists of a paragraph-question pair, where the paragraph is a news article drawn from CNN and the Daily Mail, and the question is a Cloze-style query with a masked-out entity. A model is expected to predict the masked-out entity from a list of entities marked in the paragraph utilizing commonsense knowledge. As a question-answering task, the chosen evaluation metrics are F1 and EM, with F1 being computed at the maximum token-level [80].

RTE RTE is included in SuperGLUE with the same data and format as GLUE. As a binary classification task, the goal is to determine whether a hypothesis is entailed by a premise or not. In Table 3.6, an example of the dataset is presented, which can enhance the understanding of the task given a tangible instance.

WiC Word-in-Context is a large-scale dataset designed for the generic evaluation of context-sensitive representations. The task is framed as a binary classification of sentence pairs, where sentences are drawn from WordNet [81], VerbNet [82], and Wiktionary. Each example consists of two text snippets and a polysemous target word w, either a verb or a noun, for which two contexts, c_1 and c_2, are provided. The task is to determine if the occurrence of w in c_1 and c_2 correspond to the same sense or not [83].

WSC Previously introduced within the GLUE benchmark, WSC is a challenging yet vital task assessing a model's ability to understand and make inferences from nuanced linguistic contexts. It involves resolving pronoun references, requiring syntactic and semantic comprehension as well as real-world knowledge. Nevertheless, it was recast as WNLI to be included in GLUE. Considering the nature of the task, a version of WSC is cast as a binary classification in SuperGLUE. As shown in Table 3.6, each example consists of a sentence with a marked pronoun and noun. The task is to determine whether the pronoun refers to the noun or not [73].

Similar to GLUE, SuperGLUE includes an expert-constructed diagnostic dataset designed to automatically test models for a broad range of linguistic, commonsense, and world knowledge. It is recast as a two-class dataset by collapsing the *contradiction* and *neutral* labels into a single *not_entailment* label. The dataset is evaluated using Matthew's correlation coefficient, and human performance is estimated following the same procedure for the benchmark tasks [75].

As an additional diagnostic set, Winogender is featured to facilitate the detection of social biases in natural language processing models. Winogender is a dataset designed in the style of Winograd Schema to assess gender bias in coreference resolution systems [84]. Included in SuperGLUE is the **D**iverse **N**atural **L**anguage **I**nference **C**ollection or **DNC**, which strategically incorporates a version of Winogender that is recast as a textual entailment task. In recasting, each example consists of minimal pairs: a premise or the unmodified Winogender sentence with a male or female pronoun and a short manually constructed hypothesis having a correct (*Entailed*) or incorrect (*Not-entailed*) pronoun resolution [85]. The selected evaluation metric is the gender parity score accompanied by accuracy. To evaluate the performance on Winogender, the gender parity score is defined as the percentage of minimal pairs for which the model's predictions are the same. Human performance is obtained through collecting non-expert annotations with accuracy and gender parity score.

To encapsulate, SuperGLUE succeeds GLUE by presenting a diverse set of eight challenging natural language understanding tasks, as measured by the difference between human and model baselines [75].

3.4 PyTorch

PyTorch is an open-source dynamic machine learning framework. As a Python-based library, it is designed to expedite research on machine learning models. Furthermore, it provides a high-performance environment for automatic differentiation of models executed on different devices (CPU and GPU) [86].

3.5 Hugging Face

Hugging Face, an open-source data science and machine learning platform, plays a significant role in advancing research in the field of natural language processing. At its core is the Transformer library, a comprehensive, robust collection of pre-trained models, including BERT and GPT-2. The package is dedicated to supporting Transformer-based architectures and facilitating the distribution of pre-trained models. As a pioneering platform, Hugging Face leverages libraries such as Transformers to promote and encourage collaboration among the community to build and share state-of-the-art models for various natural language processing tasks [87].

3.6 AllenNLP

AllenNLP[6]is an open-source platform designed for natural language processing research in PyTorch. It offers tutorials, API documentation, source code[7], and a broad collection of existing model implementations with high-level configuration language for seamless exploration [88].

3.7 Jiant

Jiant[8]is an open-source modular software toolkit for conducting transfer learning experiments on English natural language understanding tasks built on PyTorch [86], AllenNLP [88], and the Transformers package in HuggingFace[9]. This toolkit is used in this study as it streamlines the use of GLUE and SuperGLUE benchmarks. It offers built-in support for data preparation, model configuration, training, and evaluation, thus simplifying the experimental workflow.

3.8 WandB

Weights and **Biases**, commonly known as **WandB**, is a comprehensive platform that provides interoperable tools to track, visualize, and manage machine learning experiments in real-time. WandB supports logging metrics, visualizing model

[7]https://allenai.org/allennlp
[7]https://github.com/allenai/allennlp
[9]https://jiant.info
[9]https://github.com/huggingface/transformers

performance, managing hyperparameters, and collaboration. Additionally, it integrates seamlessly with state-of-the-art machine learning libraries such as PyTorch and Hugging Face to facilitate development and accelerate research workflows.

3.9 Catastrophic Forgetting

Catastrophic forgetting or **catastrophic interference** refers to the loss or deterioration of previously acquired knowledge. This phenomenon poses a significant challenge for neural networks, hindering their capability to acquire and adapt to new knowledge without compromising previously learned information when being trained on new tasks.

Catastrophic interference was originally discovered by Michael McCloskey and Neal J. Cohen in 1989. As pioneers in establishing the concept of catastrophic forgetting, they observed that sequential learning in backpropagation networks leads to catastrophic forgetting, where newly obtained knowledge disrupts previously acquired information, with the degree of interference increasing proportionally to the extent of newly obtained knowledge. However, this issue was not persistent when tasks were learned concurrently, framing catastrophic forgetting as a fundamental limitation of neural networks. While they successfully pinpointed the problem of catastrophic forgetting in neural networks, their proposed approaches failed to achieve satisfactory results in reducing the catastrophic interference exhibited by the neural networks.

Although McCloskey and Cohen acknowledged the catastrophic interference as a limitation of neural networks, they concluded that determining the extent to which catastrophic interference impairs neural networks is associated with discovering the specific conditions under which catastrophic interference is more severe than expected from human learners and the conditions under which interference remains within reasonable proportions [5]. In 1999, Robert M. French built upon the foundational work of McCloskey and Cohen with an in-depth study of the analogy between catastrophic forgetting in neural networks and natural cognitive systems, particularly the human cognitive system.

French argued that while the human brain is prone to gradual forgetting, complete disruption of acquired knowledge seldom occurs. Contrarily, the interference observed in neural networks is catastrophic. Additionally, French asserted that catastrophic forgetting is a radical manifestation of the stability-plasticity dilemma, which involves designing a system that is able to integrate new knowledge without disrupting previously learned information. Inspired by the then-recent research, French underscored a possible solution derived from how the human brain evolved to deal with this problem. The solution consists of two separate, permanently interacting processing areas, one for the new knowledge and the other for the long-term storage of previously learned information [6].

3.9.1 Mitigating Catastrophic Forgetting: A Review of Methods and Approaches

Early foundational efforts played an instrumental role in defining and contextualizing catastrophic forgetting, distinguishing it from forgetting in natural cognitive systems, and highlighting its abrupt and disruptive effects on neural networks. These endeavors inspired researchers to explore diverse methods to address catastrophic interference. Consequently, five main approaches have emerged in multi-layer perceptron-like architectures.

3.9.1.1 Regularization Methods

Regularization methods add constraints to the neural network's weight updates. Consequently, when the neural network is trained on a new task, the acquired knowledge is less likely to interfere with previously learned information. Notable examples of this are [89] and [90], where the latter is one of the primary approaches used in this research.

EWC Elastic Weight Consolidation, also known as EWC, is an algorithm developed to sequentially train a neural network on multiple tasks. Inspired by the mammalian brain's ability to avoid catastrophic forgetting, EWC is developed to address catastrophic forgetting in neural networks.

Given a neural network consisting of multiple layers, learning a task involves adjusting the set of weights and biases θ through linear projections followed by element-wise nonlinearities to optimize performance. Many configurations of θ result in the same performance, which implies that overparameterization enhances the probability of existing a solution for task B, θ^*_B close to the previously found solution for task A, θ^*_A. When learning task B, the EWC algorithm protects the performance in task A by constraining the parameters to remain in a region of low risk for task A centered around θ^*_A*. Hence, in this context, elastic refers to the constraint implemented as a quadratic penalty, which can be imagined as a spring that anchors the parameters to the previous solution. It is noteworthy that the stiffness of this spring should not be the same for all parameters; alternatively, it should be greater for parameters that contribute the most to the performance of task A.
Accordingly, the loss function in EWC is of the following form.

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \Sigma_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i}) \tag{3.8}$$

$\mathcal{L}_B(\theta)$ is the loss for task B, and λ indicates how important the previous task is compared to the new one. F is the Fisher information matrix which is an approximation of the second derivative of the loss near a minimum. Moreover, computing F relies only on the first-order derivatives, and it is guaranteed to be positive semi-definite. To put it in another way, the Fisher matrix is employed to constrain the weights that are important to the previous task, ensuring they remain close to their original value.

Remarkably, the algorithm is applicable to multiple tasks. EWC ensures that network parameters remain close to the parameters of the previously learned tasks by enforcing either separate quadratic penalties for each task or a combined penalty [90], [91].

3.9.1.2 Ensemble Methods

Ensemble methods attempt to alleviate catastrophic forgetting by explicitly or implicitly training multiple classifiers together. The classifiers are then combined to generate the final prediction. Learn++ and TrAdaBoost are explicit methods, while PathNet is an implicit example of an ensemble method [91]–[93].

3.9.1.3 Rehearsal Methods

Rehearsal methods attempt to mitigate catastrophic forgetting by mixing past data with the current data that is being learned. These methods are not resource-efficient as they require storing past data. Examples of rehearsal methods include [94] and [95], which have been shown to reduce catastrophic interference [91].

3.9.1.4 Dual-Memory Models

Memory consolidation in the mammalian brain is believed to store memories in two separate neural networks. Initially, new memories are stored in the hippocampus, a major brain region. Over time, they are gradually transferred to the pre-frontal cortex during sleep, where they are stored for long-term retention. Influenced by this mechanism, several algorithms have been developed, including dual-memory models. The aforementioned solution proposed by French, involving dual-memory systems inspired by the human cognitive process, aligns closely with this broader class of algorithms [91].

3.9.1.5 Sparse-Coding Methods

When new internal representations interfere with previously learned ones during the learning process, this overlap in representation can lead to catastrophic forgetting. Sparse representations help reduce the chance of this interference. Nevertheless, sparsity can diminish generalization and the ability to learn new tasks. Arguably. some of the most prominent examples of these methods are CALM, ALCOVE, sparse distributed memory, and fixed expansion layer models [91].

While these five approaches have shaped the trajectory of research to mitigate catastrophic interference, they are predominantly designed for perception-like architectures. The rapid advancement of pre-trained language models, in particular, the Transformer-based architectures, has significantly challenged the effectiveness of these approaches in mitigating catastrophic forgetting. As a result, they can not be directly applied to pre-trained language models. This stems from the fact that they require pre-training data, which is often inaccessible or massive in size, to be stored and used constructively [96].

Alternatively, it is essential to develop and adapt new strategies accordingly to overcome the catastrophic forgetting problem of fine-tuning pre-trained language models. Among recently developed strategies, adapters represent a key innovation.

3.9.1.6 Adapters

Adapters are neural modules integrated into pre-trained language models to enable task-specific fine-tuning in a parameter-efficient manner. These modules are lightweight and consist of a small set of trainable parameters Φ, while the pre-trained language model parameters Θ remain fixed during fine-tuning. This design allows adapters to encode task-specific representations in intermediate layers of the model, preventing catastrophic forgetting by isolating new knowledge in the adapter layers.

Originally proposed by [97], adapters were introduced as a strategy to fine-tune pre-trained language models without updating the full parameter set. Adapters are bottleneck layers with parameters Φ_l added to each transformer layer l. These bottleneck layers consist of a down-projection to a lower-dimensional latent space, followed by a non-linearity and an up-projection back to the original dimension. This enables task-specific learning while keeping the overall model structure intact. The adapter parameters are task-specific Φ, while the main model parameters Θ remain shared across tasks. Additionally, new layer normalization parameters or new LayerNorms are trained per task alongside the layers within the adapter modules. These new LayerNorms help stabilize task-specific representations, ensuring that the task embeddings are properly normalized and scaled within the adapter modules.

The placement and architecture of adapters within the pre-trained language model are important and may affect their efficacy. While [97] experiment with different adapter architectures empirically validates placing adapters after the feed-forward network within each transformer block, other works have explored various placements.

To address these issues and facilitate transfer learning with adapters in a range of settings [98] AdapterHub, a framework built on top of the HuggingFace Transformers library. AdapterHub provides a seamless framework for training and sharing adapters while offering scalability, modularity, and composition. Notably, the framework includes a configuration file where the architecture setting can be defined dynamically, supporting standard adapter architecture from the existing literature and extensibility.

Overall, the encapsulation of adapters enforces them to store task-specific knowledge within their own parameters. In other words, during training, each adapter learns output representations that are compatible across tasks while isolating task-specific parameters. This modularity allows multiple adapters to be combined, for instance, with the attention mechanism. Since the respective adapters are trained separately, challenges like skewed dataset sizes and sampling heuristics are avoided. By separating the processes of knowledge extraction and composition, adapters effectively mitigate catastrophic forgetting [98].

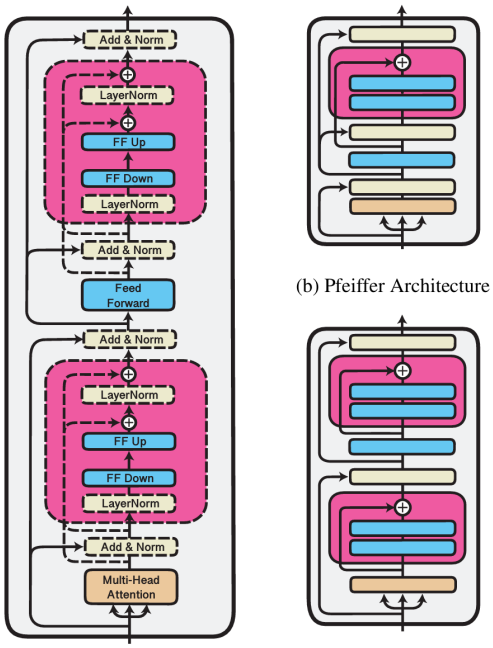


Figure 3.2: AdapterHub provides a unified interface to different adapter architectures and composition techniques, offering dynamic customization possibilities. The dashed lines in (a) show the current configuration options, which include the placement of new weights Φ (including down and up projections as well as new LayerNorms), residual connections, bottleneck sizes, and activation functions. All new eights Φ are illustrated within the pink boxes, whereas everything outside belongs to the pre-trained weights Θ. Pre-set configuration files are also provided in the literature, including architectures for [97] and [98].

3.9.1.7 Brute-Force Approach

The brute-force approach involves exposing the fine-tuned language model to its pre-training data. This straightforward approach essentially revisits the previously acquired knowledge to avoid catastrophic forgetting. Despite its high computational costs, the brute-force approach remains an available solution as a last resort to mitigate catastrophic forgetting.

This study explores three prominent approaches to mitigating catastrophic forgetting in pre-trained language models: EWC, adapter-based methods, and brute-force retraining. While these methods have demonstrated varying degrees of efficacy, emerging advancements such as Low-Rank Adaptation or LoRA [99] offer promising prospects for future research, which can be of interest to the curious minds of the study.

3.10 Experiments

The experimental setting presents a structured investigation into catastrophic forgetting in the domain-specific task of visual-to-textual knowledge transfer. Initial

investigations focus on text data instead of multi-modal models as they are extensively costly to fine-tune and evaluate. Additionally, text data provides a regulated environment for studying catastrophic forgetting, supported by benchmarks such as GLUE and SuperGLUE, ensuring clarity in interpreting results before extending the research to multi-modal contexts.

3.10.1 Baselines

The main baselines are built around BERT, in particular, the `bert-base-uncased` variant, using the jiant toolkit with its core dependencies PyTorch, AllenNLP, and HuggingFace Transformers library.

3.10.1.1 GLUE

Data for the GLUE tasks are available for download through the GLUE website[10]. jiant includes a download script to facilitate downloading each task. This script ensures that the data for each task, which comes with a standardized training set, development set, and an *unlabeled* test set, is structured in a format compatible with jiant's processing pipeline.

To fine-tune on GLUE, jiant loads the selected pre-trained model, in this case, the `bert-base-uncased`, from the Hugging Face Transformers library. The data for each task is tokenized with the corresponding tokenizer and processed to be fed to the model. The configuration files define vital experiment settings, including the model name, the corresponding tokenizer, and training hyperparameters.

The training and evaluation process is executed through jiants pipeline with the specified configuration files. During training, the model is fine-tuned on GLUE using the pre-processed data. After training is complete, jiant outputs task-specific metrics depending on the specific task requirements [46], [59], [100].

3.10.1.2 SuperGLUE

The process for fine-tuning on SuperGLUE is identical to that of GLUE, using jiant for data preparation, model configuration, training, and evaluation. Data for the SuperGLUE tasks are available for download through the SuperGLUE website[11][46], [75], [100].

3.10.2 Fine-tuning Procedure

The `bert-base-uncased` model is fine-tuned on the captions of the vision-and-language dataset (see Section 3.1.2) using an MLM objective following the practice recommended in [3]. The captions are tokenized using the model's corresponding built-in tokenizer, WordPiece. A masking procedure is applied to 15% of all WordPiece tokens in each sequence at random. Then, the model is trained to predict the

[10]https://gluebenchmark.com/tasks
[11]https://super.gluebenchmark.com/tasks

original token with cross-entropy loss. This process is logged and visualized using WandB for real-time monitoring [46].

Training is based on two approaches: (1) training for a fixed number of epochs and (2) training on gradient steps. While the hyperparameters are consistent with the study on which this work is based [3], the second approach distinctively allows saving the best-performing model during training by monitoring performance at specific checkpoints. This methodological choice reflects a trade-off between practical constraints and the need for thorough experimentation. The first approach is computationally efficient, significantly accelerating the training process, which makes it more suitable for the resource-limited settings of this study. However, the second approach, while computationally expensive and time-intensive, provides more insights into how the model evolves during training, allowing a thorough investigation of the catastrophic forgetting hypothesis. Considerably, the first approach is used for the evaluation on benchmarks, and the second approach provides a foundation for further investigation and future work. Both WandB and HuggingFace offer invaluable tools for saving optimal models at specific checkpoints; however, their resource limitations prevent their application in this study.

3.10.3 Benchmark Evaluation

The performance of the fine-tuned `bert-base-uncased` model is evaluated on the GLUE and SuperGLUE benchmarks. These benchmarks are prominently used to measure the model's performance on general language understanding tasks after fine-tuning, providing a direct comparison with baseline results. This comparison facilitates the assessment of the impact of domain-specific training and the extent of catastrophic forgetting. The evaluation is performed using the fine-tuned model trained on epochs to offer a practical balance between training efficiency and reliable results.

3.10.3.1 Evaluation on GLUE

The evaluation process on GLUE follows a procedure similar to the baselines with the exception of the fine-tuned `bert-base-uncased` model being used. jiant facilitates this process in the same manner as the baselines, leveraging its configuration-driven pipeline to streamline evaluation. Configurations serve as jiants primary user interface, ensuring that tasks and modeling components remain modular while enabling a seamless integration of workflows. This modularity, as well as the pipeline structure, simplifies the execution of evaluations across the multiple tasks in GLUE, providing consistency and reproducibility throughout the experimental process [100].

3.10.3.2 Evaluation on SuperGLUE

The evaluation process on SuperGLUE follows the same approach as that of GLUE, utilizing the fine-tuned `bert-base-uncased` model within jiant's pipeline, which can be specified through configuration files [100]. The task-specific metrics calculated during this process provide a detailed analysis of the models performance, further supporting the investigation of catastrophic forgetting.

3.10.4 Mitigating Catastrophic Forgetting

This study aimed to explore three prominent approaches to mitigating catastrophic forgetting in pre-trained language models: EWC, adapter-based methods, and brute-force retraining. Each of these methods offers distinct mechanisms to address forgetting, ranging from penalizing parameter updates based on their importance to modularizing task-specific knowledge or completely retraining the model across tasks. However, practical constraints such as computational power and time significantly influenced the scope of this exploration.

Elastic Weight Consolidation, or EWC, is a theoretically promising approach for mitigating catastrophic forgetting by regularizing parameter updates using the Fisher information matrix to penalize changes to parameters deemed critical for previously learned tasks. While EWC is computationally feasible for smaller models, scaling it to large pre-trained models, in this case, the `bert-base-uncased` model with 110M total parameters, presents significant challenges. Calculating the Fisher information matrix for 110 million parameters involves extensive memory and computational resources, as it requires estimating gradients and second-order derivatives for each parameter. Due to these constraints, implementing EWC is not practical within the scope of this study.

Adapter-based methods, which introduce lightweight, task-specific modules while freezing the pre-trained models parameters, are designed to address catastrophic forgetting efficiently. However, implementing these methods across multiple tasks and benchmarks requires sufficient memory and training resources, which exceeds the computational capacities available for this study.

Brute-force retraining is the only approach partially implemented in this research. This method involves retraining the model on both the pre-training data and the captions of the vision-and-language dataset (see Section 3.1.2), allowing the model to recall the pre-trained knowledge while adapting to new tasks. Although this approach is computationally demanding, it is explored as a baseline to understand the trade-offs between resource requirements and mitigation efficacy. Due to resource limitations, the model could only be trained for a small number of epochs before the process had to be terminated.

While the full implementation of mitigation strategies is not feasible, this study prioritized fine-tuning and evaluation to establish a foundational understanding of catastrophic forgetting in domain-specific tasks. The findings provide a basis for future work to implement and evaluate mitigation strategies under improved computational conditions. The preliminary results are discussed in the following chapter and offer insights into the practical challenges of these methods within constrained environments.

4

Results

This chapter presents the empirical findings obtained through this thesis.

4.1 Baselines

Pre-trained `bert-base-uncased` establishes strong baseline performance across both benchmarks. The process follows the recommended task-specific configurations outlined in the jiant framework [100].

4.1.1 Evaluating Pre-Trained BERT on GLUE

Table 4.1 presents the baseline performance of the original pre-trained `bert-base-uncased` model. It achieves an average score of 76.0, with strong performance in tasks such as SST-2 (93.8) and QNLI (90.6). These results reflect BERTs robust general-purpose language understanding capabilities, particularly in sentiment analysis and question inference. However, the model shows lower performance on syntactically challenging tasks such as CoLA (36.8).

		Single Sentence		Similarity and Paraphrase			Natural Language Inference					
Model	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
Metrics		Matthews corr.	acc.	acc./F1	Pearson/Spearman corr.	acc./F1	matched acc.	mismatched acc.	acc.	acc.	acc.	Matthews corr.
$\text{BERT}_{\text{BASE}}$ (uncased)	76.0	36.8	93.8	78.1/83.8	81.4/80.8	88.2/70.7	83.6	83.0	90.6	72.6	65.1	33.9

Table 4.1: Performance on GLUE test sets and diagnostic. All values are scaled by 100.

4.1.2 Evaluating Pre-Trained BERT on SuperGLUE

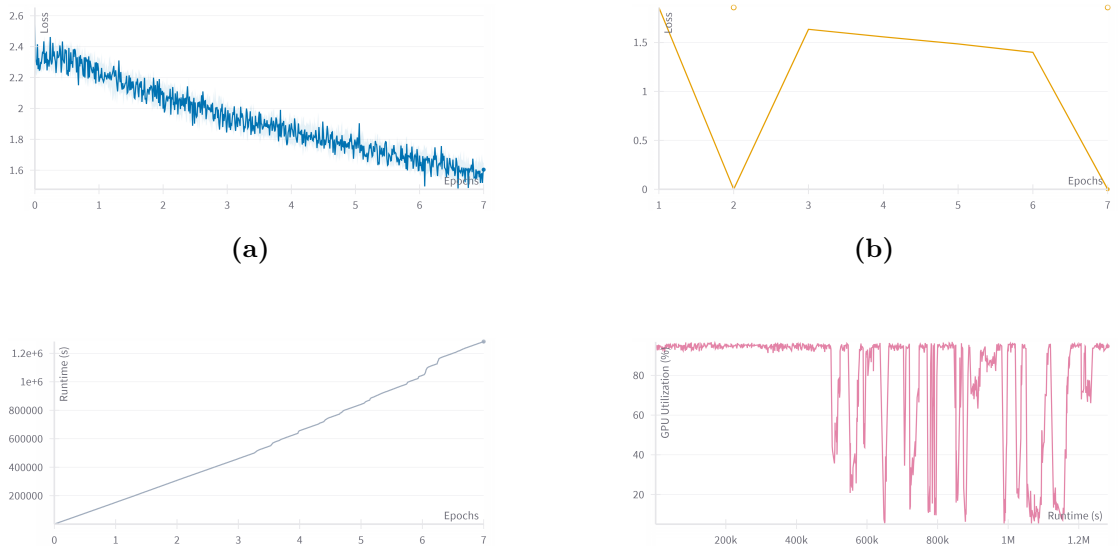
Table4.2 reports the performance of the pre-trained BERT model on SuperGLUE tasks. The model achieves an average score of 55.9, with balanced results across tasks such as BoolQ (62.3), CB (30.7), and ReCoRD (56.5). While these results are below the state-of-the-art, they reflect a consistent level of competence across a variety of reasoning and understanding tasks.

Model	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX_b	AX_g
Metrics		acc.	F1/acc.	acc.	$F1_\alpha$/EM	F1/EM	acc.	acc.	acc.	Matthews corr.	GPS/acc.
$BERT_{BASE}$ (uncased)	55.9	62.3	30.7/45.2	67.6	57.6/13.2	56.5/55.8	61.3	64.4	62.3	6.1	99.4/52.5

Table 4.2: Performance on SuperGLUE test sets and diagnostics. All values are scaled by 100. The score is the overall score on non-AX_* tasks.

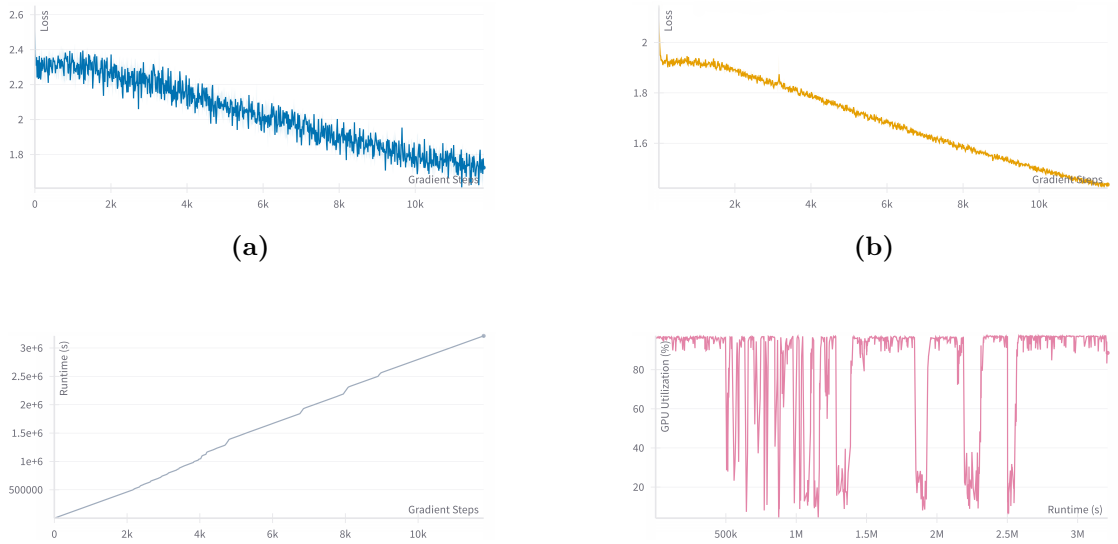
4.1.3 Fine-Tuning BERT on Vision-and-Language Dataset

The pre-trained `bert-base-uncased` model was fine-tuned on the captions of the Vision-and-Language dataset using two distinctive approaches with Adam optimizer and a learning rate of $5e-5$.



(a) **(b)** **(c)** **(d)**

Figure 4.1



(a) (b) (c) (d)

Figure 4.2

4.1.4 Benchmark Evaluation

To assess the extent of catastrophic forgetting, we evaluate the model on the GLUE and SuperGLUE benchmarks after fine-tuning.

4.1.4.1 Evaluating Fine-Tuned BERT on GLUE

Table 4.3 shows the performance after brute-force fine-tuning on the vision-and-language dataset. The average GLUE score drops from 76.0 to 63.9, indicating a clear loss of generalization ability. The most severe declines occur in CoLA (from 36.8 to 6.4) and WNLI (from 65.1 to 34.9). These tasks rely heavily on grammaticality and coreference understanding, which are underrepresented in the fine-tuning data. The consistent drop across tasks indicates that catastrophic forgetting occurred, particularly affecting language understanding abilities not reinforced during fine-tuning.

		Single Sentence		Similarity and Paraphrase			Natural Language Inference					
Model	**Score**	**CoLA**	**SST-2**	**MRPC**	**STS-B**	**QQP**	**MNLI-m**	**MNLI-mm**	**QNLI**	**RTE**	**WNLI**	**AX**
Metrics		**Matthews corr.**	**acc.**	**acc./F1**	**Pearson/Spearman corr.**	**acc./F1**	**matched acc.**	**mismatched acc.**	**acc.**	**acc.**	**acc.**	**Matthews corr.**
BERT++	63.9	6.4	87.1	73.9/82.5	77.7/76.8	84.4/64.4	72.7	72.6	81.5	62.6	34.9	22.3

Table 4.3: Performance on GLUE test sets and diagnostic. All values are scaled by 100.

4.1.4.2 Evaluating Fine-Tuned BERT on SuperGLUE

Table 4.4 presents the results of the fine-tuned model on the same SuperGLUE benchmark. The average score drops slightly from 55.9 to 53.3. Although the overall decline is modest, task-specific performance reveals sharper degradation. The precision of the CB drops significantly (from 30.7 to 16.7), and AX (from 6.1 to 2.9),

indicating diminished abilities in the inference of natural languages and the linguistic acceptability. RTE also decreases slightly (from 61.3 to 59.7), while WiC and WSC remain relatively stable (WiC: from 64.4 to 61.7; WSC: from 62.3 to 63.7).

Model	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX_b	AX_g
Metrics		acc.	F1/acc.	acc.	$F1_\alpha$/EM	F1/EM	acc.	acc.	acc.	Matthews corr.	GPS/acc.
BERT++	53.3	62.2	16.7/20.0	65.8	60.8/14.9	57.7/57	59.7	61.7	63.7	2.9	98.9/53.9

Table 4.4: Performance on SuperGLUE test sets and diagnostics. All values are scaled by 100. The score is the overall score on non-AX$_*$ tasks.

5

Conclusion

5.1 Discussion

This study explored the impact of fine-tuning pre-trained language models on downstream performance, with a specific focus on the phenomenon of catastrophic forgetting. Using BERT as the base model, we employed a brute-force fine-tuning strategy on a vision-and-language dataset and evaluated its performance on GLUE and SuperGLUE benchmarks. The motivation behind this approach was to test whether a straightforward adaptation could preserve prior knowledge while acquiring new task capabilities.

However, the experimental results indicated a clear performance drop: in the fine-tuned model compared to the original pre-trained model across multiple benchmark tasks. As reported in Section 4, the average GLUE score dropped from 76.0 to 63.9, with the most severe degradations in CoLA and WNLI tasks requiring syntactic and coreference understanding. On SuperGLUE, although the overall score decreased only slightly (55.9 to 53.3), task-specific declines were evident, particularly in CB and AX, which test natural language inference and linguistic acceptability. Only a few tasks, such as WSC and MultiRC, remained relatively stable. This suggests that catastrophic forgetting occurred despite the models extensive pre-training. One possible explanation lies in the lack of regularization or rehearsal mechanisms during fine-tuning, which are typically essential for retaining knowledge in continual learning contexts. Additionally, the nature of the vision-and-language data, which may differ significantly from the original training objectives of BERT, likely contributed to representational drift.

The results reflect a broader issue in transfer learning: the assumption that pre-trained models can be easily adapted to new tasks without substantial performance trade-offs does not always hold, particularly when the new task diverges in modality or domain. This highlights the need for more nuanced and adaptive learning strategies that account for task interference and knowledge retention.

5.2 Conclusion

In this thesis, we investigated catastrophic forgetting in large language models, focusing on BERT as a representative pre-trained transformer. A brute-force fine-tuning

method was applied to adapt the model to a vision-and-language task, followed by evaluations on established NLP benchmarks: GLUE and SuperGLUE to assess performance retention.

The findings clearly indicate that brute-force fine-tuning failed to mitigate catastrophic forgetting. The fine-tuned model exhibited inferior performance on both GLUE and SuperGLUE tasks compared to its pre-trained baseline, reinforcing the observation that naive fine-tuning can erode existing linguistic competencies.

This outcome highlights the limitations of simple adaptation strategies and underscores the importance of exploring more sophisticated continual learning methods.

Ultimately, this work contributes to a growing body of evidence that while pre-trained models offer powerful generalization capabilities, preserving their performance across tasks requires careful consideration of training dynamics and mitigation strategies against forgetting. Also, it provides provides empirical evidence that catastrophic forgetting remains a significant obstacle in transfer learning

5.3 Future Work

To address the shortcomings observed in this study, future research should explore more structured and memory-aware learning methods. Two promising directions include:

Adapter-Based Fine-Tuning [97]: This approach introduces small task-specific adapter modules between transformer layers while keeping the main pre-trained weights frozen. By limiting the scope of parameter updates, adapters help mitigate interference and enable modular fine-tuning across tasks.

Elastic Weight Consolidation (EWC) [90]: EWC adds a regularization term to the loss function that penalizes deviations from previously important weights. This strategy allows the model to learn new tasks while preserving critical knowledge learned during pre-training.

Ultimately, a more principled and controlled fine-tuning process is essential for ensuring that large pre-trained models maintain their capabilities across diverse and evolving task distributions

Bibliography

[1] OpenAI, *Gpt-4 technical report*, 2023. DOI: 10.48550/ARXIV.2303.08774. [Online]. Available: https://arxiv.org/abs/2303.08774.

[2] H. Touvron, L. Martin, K. Stone, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[3] T. Norlund, L. Hagström, and R. Johansson, "Transferring knowledge from vision to language: How to achieve it and how to measure it?" *arXiv preprint arXiv:2109.11321*, 2021.

[4] A. Radford, I. Sutskever, J. W. Kim, G. Krueger, and S. Agarwal, "Clip: Connecting text and images," *OpenAI. https://openai. com/blog/clip/,(accessed 2022-04-14)*, 2021.

[5] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, Elsevier, 1989, pp. 109–165.

[6] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[7] I. Holmström, "Teaching a language in another modality: A case study from swedish sign language l2 instruction," *Journal of Language Teaching and Research*, vol. 10, no. 4, pp. 659–672, 2019.

[8] Y. Kim, K. Thayer, G. S. Gorsky, and J. Heer, "Color names across languages: Salient colors and term translation in multilingual color naming models.," in *EuroVis (Short Papers)*, 2019, pp. 31–35.

[9] A. D. Friederici, *Language in our brain: The origins of a uniquely human capacity*. MIT Press, 2017.

[10] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön, *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[11] J. Eisenstein, "Natural language processing," *Jacob Eisenstein*, 2018.

[12] T. Regier and P. Kay, "Language, thought, and color: Whorf was half right," *Trends in cognitive sciences*, vol. 13, no. 10, pp. 439–446, 2009.

[13] K. M. Kiely, "Cognitive function," in *Encyclopedia of Quality of Life and Well-Being Research*, A. C. Michalos, Ed. Dordrecht: Springer Netherlands, 2014, pp. 974–978, ISBN: 978-94-007-0753-5. DOI: 10.1007/978-94-007-0753-5_426. [Online]. Available: https://doi.org/10.1007/978-94-007-0753-5_426.

[14] Y. Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, p. 3881, 2008.

[15] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[16] F. Almeida and G. Xexéo, "Word embeddings: A survey," *arXiv preprint arXiv:1901.09069*, 2019.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[20] B. Mehlig, "Machine learning with neural networks," *arXiv preprint arXiv:1901.05639*, 2019.

[21] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[22] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[23] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] D. Mayerich, R. Sun, and J. Guo, "Chapter fifteen - deep learning," in *Microscope Image Processing (Second Edition)*, F. A. Merchant and K. R. Castleman, Eds., Second Edition, Academic Press, 2023, pp. 431–456, ISBN: 978-0-12-821049-9. DOI: https://doi.org/10.1016/B978-0-12-821049-9.00015-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128210499000150.

[25] Y. Goldberg, *Neural network methods for natural language processing*. Springer Nature, 2022.

[26] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[29] J. A. Hertz, *Introduction to the theory of neural computation*. Crc Press, 2018.

[30] S. Ruder, "Neural transfer learning for natural language processing," Ph.D. dissertation, NUI Galway, 2019.

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

[32] C. Lemaréchal, "Cauchy and the gradient method," *Doc Math Extra*, vol. 251, no. 254, p. 10, 2012.

[33] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[34] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[37] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[38] S. Dehaene, *How we learn: Why brains learn better than any machine... for now*. Penguin, 2021.

[39] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[40] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *International conference on machine learning*, PMLR, 2017, pp. 1243–1252.

[41] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[42] S. Niu, Y. Liu, J. Wang, and H. Song, "A decade survey of transfer learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151–166, 2020.

[43] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020.

[44] W. L. Taylor, "cloze procedure: A new tool for measuring readability," *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[45] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," 2018.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[47] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[48] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[49] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[50] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152. DOI: 10.1109/ICASSP.2012.6289079.

[51] J. Pérez-Carpinell, M. De Fez, R. Baldoví, and J. C. Soriano, "Familiar objects and memory color," *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society*

for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur, vol. 23, no. 6, pp. 416–427, 1998.

[52] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[53] V. Ordonez, G. Kulkarni, and T. Berg, "Im2text: Describing images using 1 million captioned photographs," *Advances in neural information processing systems*, vol. 24, 2011.

[54] R. Krishna, Y. Zhu, O. Groth, *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International journal of computer vision*, vol. 123, no. 1, pp. 32–73, 2017.

[55] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 2556–2565.

[56] Y. Zhu, R. Kiros, R. Zemel, *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[57] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International journal of data mining & knowledge management process*, vol. 5, no. 2, p. 1, 2015.

[58] J. Lever, "Classification evaluation: It is important to understand both what a classification metric expresses and what it hides," *Nature methods*, vol. 13, no. 8, pp. 603–605, 2016.

[59] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[60] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, 2019.

[61] R. Socher, A. Perelygin, J. Wu, *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[62] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[63] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation," *arXiv preprint arXiv:1708.00055*, 2017.

[64] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[65] P. Kavumba, R. Takahashi, and Y. Oda, "Are prompt-based models clueless?" *Journal of Natural Language Processing*, vol. 29, no. 3, pp. 991–996, 2022.

[66] A. S. White, P. Rastogi, K. Duh, and B. Van Durme, "Inference is everything: Recasting semantic resources into a unified evaluation framework," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017, pp. 996–1005.

[67] D. Demszky, K. Guu, and P. Liang, "Transforming question answering datasets into natural language inference datasets," *arXiv preprint arXiv:1809.02922*, 2018.

[68] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[69] I. Dagan, O. Glickman, and B. Magnini, "The pascal recognising textual entailment challenge," in *Machine Learning Challenges Workshop*, Springer, 2005, pp. 177–190.

[70] R. B. Haim, I. Dagan, B. Dolan, *et al.*, "The second pascal recognising textual entailment challenge," in *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, vol. 7, 2006.

[71] D. Giampiccolo, B. Magnini, I. Dagan, and W. B. Dolan, "The third pascal recognizing textual entailment challenge," in *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007, pp. 1–9.

[72] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo, "The fifth pascal recognizing textual entailment challenge.," in *TAC*, 2009.

[73] H. Levesque, E. Davis, and L. Morgenstern, "The winograd schema challenge," in *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[74] J. Gorodkin, "Comparing two k-category assignments by a k-category correlation coefficient," *Computational biology and chemistry*, vol. 28, no. 5-6, pp. 367–374, 2004.

[75] A. Wang, Y. Pruksachatkun, N. Nangia, *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.

[76] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions," *arXiv preprint arXiv:1905.10044*, 2019.

[77] M.-C. De Marneffe, M. Simons, and J. Tonhauser, "The commitmentbank: Investigating projection in naturally occurring discourse," in *proceedings of Sinn und Bedeutung*, vol. 23, 2019, pp. 107–124.

[78] M. Roemmele, C. A. Bejan, and A. S. Gordon, "Choice of plausible alternatives: An evaluation of commonsense causal reasoning," in *2011 AAAI Spring Symposium Series*, 2011.

[79] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth, "Looking beyond the surface: A challenge set for reading comprehension over multiple sentences," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 252–262.

[80] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. Van Durme, "Record: Bridging the gap between human and machine commonsense reading comprehension," *arXiv preprint arXiv:1810.12885*, 2018.

[81] G. A. Miller, "WordNet: A lexical database for English," in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. [Online]. Available: `https://aclanthology.org/H94-1111`.

[82] K. K. Schuler, *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, 2005.

[83] M. T. Pilehvar and J. Camacho-Collados, "Wic: The word-in-context dataset for evaluating context-sensitive meaning representations," *arXiv preprint arXiv:1808.09121*, 2018.

[84] R. Rudinger, J. Naradowsky, B. Leonard, and B. Van Durme, "Gender bias in coreference resolution," *arXiv preprint arXiv:1804.09301*, 2018.

[85] A. Poliak, A. Haldar, R. Rudinger, *et al.*, "Collecting diverse natural language inference problems for sentence representation evaluation," *arXiv preprint arXiv:1804.08207*, 2018.

[86] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in pytorch," 2017.

[87] T. Wolf, L. Debut, V. Sanh, *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[88] M. Gardner, J. Grus, M. Neumann, *et al.*, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[89] G. E. Hinton and D. C. Plaut, "Using fast weights to deblur old memories," in *Proceedings of the ninth annual conference of the Cognitive Science Society*, 1987, pp. 177–186.

[90] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[91] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[92] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[93] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 193–200.

[94] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[95] T. J. Draelos, N. E. Miner, C. C. Lamb, *et al.*, "Neurogenesis deep learning: Extending deep networks to accommodate new classes," in *2017 international joint conference on neural networks (IJCNN)*, IEEE, 2017, pp. 526–533.

[96] S. Chen, Y. Hou, Y. Cui, W. Che, T. Liu, and X. Yu, "Recall and learn: Fine-tuning deep pretrained language models with less forgetting," *arXiv preprint arXiv:2004.12651*, 2020.

[97] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2790–2799.

[98] J. Pfeiffer, A. Rücklé, C. Poth, *et al.*, "Adapterhub: A framework for adapting transformers," *arXiv preprint arXiv:2007.07779*, 2020.

[99] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[100] Y. Pruksachatkun, P. Yeres, H. Liu, *et al.*, "Jiant: A software toolkit for research on general-purpose text understanding models," *arXiv preprint arXiv:2003.02249*, 2020.