# *Reformulation and Decomposition: Multitask learning approaches to Long Document Problems*

GEORGE THOMAS HUDSON

# Reformulation and Decomposition: Multitask learning approaches to Long Document Problems

**G Thomas Hudson**

A thesis presented for the degree of
Doctor of Philosophy at Durham University


Department of Computer Science
Durham University
United Kingdom

7th May 2024

# Abstract

Recent advances in Natural Language Processing (NLP) have led to success across a wide range of tasks including machine translation, summarization, and classification. Yet, the field still faces major challenges. This thesis addresses two key under-researched areas: the absence of general multitask learning capabilities, and the inability to scale to long, complex documents. Firstly, this thesis explores a form of multitasking where NLP tasks are reformulated as question answering problems. I examine existing models and measure their robustness to paraphrasing of their input. I contribute an annotated dataset which enables detailed analysis of model failures as well as evaluating methods for improving model robustness. Secondly, a set of long document tasks; MuLD, is introduced which forms a benchmark for evaluating the performance of models on large inputs with long-range dependencies. I show that this is a challenging task for baseline models. I then design an approach using task-decomposition to provide an interpretable solution which easily allows for multitask learning. I then explore how these themes of task reformulation for multitask learning, and task-decomposition for long inputs can be applied to other modalities. I show how visual modelling: a visual analogue of language modelling, can be used to predict missing frames from videos of simple physics simulations, and probe what knowledge about the physical world this induces in such models. Finally, I demonstrate how this task can be used to unite vision and NLP using the same framework, describing how task-reformulation and task-decomposition can be used for this purpose.

# Declaration

The work in this thesis is based on research carried out within the Innovative Computing Group at the Department of Computer Science at Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all the author's own work unless referenced to the contrary in the text.

# Acknowledgements

Firstly, I will be forever indebted to my supervisor Noura Al Moubayed for her guidance throughout this PhD. Your unwavering belief in me and tireless support has made this journey possible.

I would also like to thank everyone in Noura's research group. Your insights and company have made this thesis what it is. In particular, I wish to thank Tom Winterbottom for his friendship and support on this long journey, and for being all round good egg.

Also, gratitude goes to my family: Mum, Dad, Eleanor for their encouragement over all these years. You can stop asking if I'm still writing my thesis now - it really is over. Thanks to Evelyn for her patience, endless encouragement, and willingness to endure checking over 30,000 words of my writing.

This document would not exist without you all in my life.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**AI** Artificial Intelligence. 163

**BLEU** BiLingual Evaluation Understudy. 14, 101

**CNN/DM** Cable News Network/Daily Mail. 15, 16

**COMET** Crosslingual Optimized Metric for Evaluation of Translation. 14

**decaNLP** Natural Language Decathlon. 12, 15, 16

**GLUE** General Language Understanding Evaluation. 28

**IWSLT** International Conference on Spoken Language Translation. 13, 15

**LSTM** Long Short Term Memory. 21

**METEOR** Metric for Evaluation of Translation with Explicit ORdering. 14, 101

**MNLI** Multi-Genre NLI Corpus. 12

**MuLD** Multitask Long Document Benchmark. 5, 6, 15, 16

**MWSC** Modified Winigrad Schema Challenge. xi, 57

**NLI** Natural Language Inference. 11, 29, 44

**NLP** Natural Language Processing. 1–8, 10, 12, 18, 20, 27–29, 33, 40, 42, 44, 45, 49, 84, 85, 108, 110, 155, 157, 159, 160, 165, 166

**QA** Question Answering. 16–18, 32, 40, 105

**RNN** Recurrent Neural Network. 20, 21, 26

**ROUGE** Recall-Oriented Understudy for Gisting Evaluation. 14, 101

**SNLI** Stanford Natural Language Inference. 11, 12

**SQUaD** Standford Question Answering Dataset. 17

**SVM** Support Vector Machines. 18

**VLSP** Very Long Scientific Papers. 86

**WMT** Workshop on Machine Translation. 13, 14

**WRPA** Relational Paraphrase Acquisition from Wikipedia. 45

# Chapter 1

# Introduction

Human language is one of the most complex and versatile forms of communication, enabling us to express a wide range of concepts, thoughts, and emotions with often subtle variations in syntax and semantics. The ability to understand human language has been a long-standing challenge in Computer Science, and recent advances in this field of Natural Language Processing (NLP) seem to have made significant progress towards this goal.

Yet in many areas, the human still reigns supreme.

NLP approaches are usually based on deep learning, which require models to be trained on large datasets, often to perform a single specific task in isolation. In contrast, humans learn many tasks simultaneously and can transfer knowledge between them, making it possible for them to perform new tasks with relatively little training. This method of training, known as multitask learning, is effective in many areas of machine learning, including NLP (Radford et al., 2019a). There are numerous approaches to multitask learning, including modifying the model architecture and

training objectives. In this thesis, I focus on a task-reformulation approach and use it to explore the robustness of these models, providing analysis and potential improvements.

Another limitation of many of the approaches used is that they are unable to scale to long documents which contain complex structures, rich information, and long-range contextual dependencies. Whilst some work has allowed the processing of documents of a few thousand words in length, humans can successfully navigate and understand documents of tens and hundreds of thousands of words long (such as books, technical reports, and this thesis itself). For NLP to be truly useful, we need to be able to solve a wide range of tasks on documents of this scale.

This thesis explores these two problems, focusing initially on improving the robustness of a type of multitask learning framed around question answering. I then expand this to look at multitask NLP for long documents, first creating a benchmark of long document tasks, and then developing a model able to be used on this dataset. Finally, I apply these ideas to other modalities, suggesting ways that these techniques can be used as part of a multimodal future.

## 1.1 Motivation

During the undertaking of this thesis, the field of NLP has experienced a profound transformation.

The trend towards increasingly large models developed by organizations with near infinite resources place limits on what can be achieved by a single researcher working alone. Instead of competing to gain points on benchmarks, I instead focus on more fundamental research questions: How robust are current multitask models? Can they be broken by paraphrasing of their input? Can we develop ways of properly evaluating long document models? In a multitask way? And can these ideas be extended to other modalities?

The ability to process long documents is vital to solve the complex real-world use-cases we need models for. Many of our repositories of knowledge, including great works of fiction, transcripts

of important meetings, and cases that underpin our legal system, exist in forms unable to be fully understood and processed by models due to their length. A chatbot also requires the processing of long documents in order to remember the context of long-term conversations. This is especially important when creating digital companions where the model needs to recall events which occurred months or even years ago.

Multitask learning is also vital for the widespread adoption of machine learning. By solving multiple tasks simultaneously, we can improve the zero-shot ability of models to solve new tasks without specific training data. Users are then able to ask models to perform any task without being limited to a narrow set of allowable inputs. Multitask learning also promises to improve performance across all tasks by sharing knowledge, making better use of the resources available to us.

Lastly, the final work of this thesis is motivated by observing the transformative effect of large language models, and prompt-based multitask learning on NLP. Analysing a similar paradigm in computer vision leads me to explore how the ideas of this thesis can be applied in other modalities. The ultimate conclusion of this is to one day unite both NLP and computer vision in a single model, allowing a range of tasks not currently possible to be solved.

## 1.2 Thesis Contributions

The main contributions of this thesis are:

- An analysis of the robustness of multitask models to paraphrasing of their inputs. This involves creating my own dataset of paraphrases which have been annotated with individual paraphrase phenomena.
- Proposed solutions for improving the robustness of these models to paraphrasing.

- A multitask long document benchmark which can be used to evaluate the performance of models on a variety of real-world NLP tasks. This benchmark includes documents with a minimum length an order of magnitude longer than existing datasets, and includes a diverse set of tasks on documents such as novels and entire film scripts.
- A new model exploring a task-decomposition approach which can be applied to this dataset. This model requires no task-specific modification and can therefore be trained in a multitask way.
- Analysis of a range of models which are trained to solve the 'visual modelling' task on this benchmark.
- Exploration of how 'visual modelling' can be used as a 'super-task' uniting vision and NLP in a single model.

## 1.3 Publications

Many chapters of this thesis have been submitted for publication or published in journals/conference proceedings. The relevant chapters are indicated below:

- Chapter 3 contains work presented in G. Thomas Hudson, and Noura Al Moubayed, **"Ask me in your own words: paraphrasing for multitask question answering."** *PeerJ Computer Science 7* (2021): e759.
- Chapter 4 contains work presented in G. Thomas Hudson, and Noura Al Moubayed, **"MuLD: The Multitask Long Document Benchmark."** *Language Resources and Evaluation Conference. Marseille, The European Language Resources Association* (2022)
- Chapter 5 contains work presented in G. Thomas Hudson, and Noura Al Moubayed, **"A multitask task-decomposition approach to long document NLP tasks."** *Transactions of the Association for Computational Linguistics (TACL)* (Under Review)

- Chapter 6 contains work presented in G. Thomas Hudson, Thomas Winterbottom, Daniel Kluvanec, and Noura Al Moubayed, **"Visual Modelling: The Visual Parallel to Language Modelling Evaluated on Dynamic Simulations"** *Journal of Machine Learning Research (JMLR)* (Under Review)

## 1.4 Thesis Structure

This thesis explores both how approaches to multitask learning can be improved and suggests ways that these can be extended to long documents and other modalities.

Chapter 2 reviews the existing literature related to multitask learning, long document NLP, and paraphrasing, which is a prerequisite for understanding the rest of this thesis.

Chapter 3 explores a class of multitask models which reformulate NLP tasks as question answering. A crowdsourced dataset of paraphrased task-prompt questions, hand-annotated with paraphrase phenomena, is introduced. This is then used to analyse models trained on the decaNLP benchmark which expect a prompt question (E.g. What is the translation of this from English to German?) as input. I use the paraphrase phenomena annotations to gain a detailed insight into what types of paraphrasing cause these models to fail, and investigate why this might be. Chapter 3 also contains proposed solutions which aim to increase the robustness of these models, both by training the models on more varied inputs, and by making use of the multitask property of these models.

Chapter 4 proposes a new benchmark for long document NLP, the Multitask Long Document Benchmark (MuLD). I discuss the limitations of current datasets and argue that a true test of long document models requires a much longer and more varied dataset than currently available. I detail the creation of a new benchmark by extending and modifying data used for shorter tasks.

We also provide a simple baseline model, which serves as a reference point for evaluating the performance of other models.

In Chapter 5, I explore a task-decomposition model that can be applied to the MuLD benchmark proposed in Chapter 4. I describe my method for framing multiple long document tasks using this task-agnostic approach, and how this can be used to enable multitask long document NLP for any length of document. I also discuss the advantages and limitations of this approach, as well as potential areas for future research and improvement.

In Chapter 6, I take the ideas introduced in previous chapters - namely task reformulation and task decomposition - and explore how they can be applied to the vision domain. I introduce the concept of "visual modelling" as the visual parallel to language modelling. As training models on language modelling induces related knowledge about both linguistic concepts and real-world knowledge, I examine what knowledge visual modelling induces. To do this, I introduce datasets of simple physics simulations and associated 'test-tasks' which examine whether the models have knowledge of simple physical laws. I then discuss how this task can unite NLP and vision in a single framework, and how future work can apply the techniques of Chapter 5 to the visual modelling task.

Finally, in Chapter 7, I present a summary of the contributions made to the field and discuss potential future directions which expand upon this work and address its limitations.

# Chapter 2

# Literature Review

In this Chapter, I provide context to the later contributions by exploring and comparing the related work in a range of different areas. Firstly, I provide an overview of the fundamental tasks in NLP along with the approaches used to solve them. I focus on the basis of many recent NLP approaches - large language models based on the transformer model. I then discuss one of the major themes of this thesis - multitask learning, exploring how different methods of multitask learning can be categorized and their properties. I make special discussion of the task reformulating approach I explore, including the decaNLP challenge which forms much of the basis of Chapter 3. I discuss long document NLP, exploring both the datasets and models used to process these efficiently. I also review methods which have used similar task-decomposition approaches to my proposed solution in Chapter 5. Finally, I review the more theoretical topic of paraphrasing, including different typologies used to categorize specific paraphrase phenomena and how this applies to NLP techniques.

Note: As Chapter 6 is the only chapter which explores the extension of this thesis' ideas into the vision domain, I discuss related work relevant to that chapter within the chapter itself.

## 2.1 Natural Language Processing

Natural Language Processing (NLP) refers to techniques which enable machines to analyse, process, and create human language, both written and spoken. Alongside computer vision, it is one of the major subdomains of machine learning and includes a wide variety of tasks including translating from one type of text to another, generating entirely new text, as well as labelling existing text.

The following sections detail the types of NLP tasks relevant to this thesis, focusing on the datasets used to train and evaluate NLP models. Table 2.1 summarizes the datasets used throughout this thesis.

| Dataset Name | Task Type | Benchmark | Chapters Used | # Documents | Avg. Tokens/Doc |
|---|---|---|---|---|---|
| IWSLT | Translation | (PQ)decaNLP | Chapter 3 | 199,182 | 17.1 |
| CNN/DM | Summarization | (PQ)decaNLP | Chapter 3 | 312,085 | 646.6 |
| MNLI | Natural Language Inference | (PQ)decaNLP | Chapter 3 | 432,702 | 20.5 |
| SST | Sentiment Analysis | (PQ)decaNLP | Chapter 3 | 9,613 | 16.7 |
| WOZ | Dialogue State Tracking | (PQ)decaNLP | Chapter 3 | 5,012 | 8.6 |
| MWSC | Commonsense Reasoning | (PQ)decaNLP | Chapter 3 | 262 | 18.7 |
| WikiSQL | Semantic Parsing | (PQ)decaNLP | Chapter 3 | 80,654 | 51.2 |
| Open Subtitles | Translation | MuLD | Chapter 4, 5 | 5,637 | 13,952 |
| AO3 Style Change | Style Change Detection | MuLD | Chapter 4, 5 | 9,411 | 29,841 |
| Movie Character Type | Character Archetype Classification | MuLD | Chapter 4, 5 | 253 | 45,838 |
| Very Long Scientific Papers (VLSP) | Summarization | MuLD | Chapter 4, 5 | 482 | 57,473 |
| HotpotQA | Question Answering | MuLD | Chapter 4, 5 | 95,157 | 23,689 |
| NarrativeQA | Question Answering | MuLD | Chapter 4, 5 | 45,881 | 90,689 |
| Booksum/Goodreads2020 | Summarization | - | Chapter 5 | 405 | 110,000 |

Table 2.1: Summary of the datasets used in this thesis

### 2.1.1 Classification

Classification is the task of assigning some label to an input. This can be binary where there are two labels to choose from (e.g. classifying an email as 'spam' or 'not spam'), or multiclass where more options are available (e.g. classifying the opinions expressed in product reviews as 'positive', 'neutral', or 'negative'). There are also multi-label problems where each input can be assigned multiple labels.

There are many classification tasks in NLP. A simple multi-class dataset which is commonly used for evaluating basic models is 20 Newsgroups (Lang, 1995). This dataset consists of around 20,000 newsgroup posts evenly split into 20 classes based on the post topic (e.g. space, religion, hockey, etc). This is a simple task for many models, as each class contains unique terminology, allowing simple Bag-of-words techniques which ignore word order to be successful (Tong and Koller, 2001; Nigam et al., 2000; Zhang, Chen and Lee, 2005)

Beyond topic classification, more complex tasks require models which can perform more complex reasoning. In the RumourEval stance classification task for example, two sentences are provided, and the task is to predict whether the second sentence supports, denies, queries, or comments on the first (Derczynski et al., 2017). This requires models which solve this task to understand the relationship between the semantics of the two sentences.

As well as tasks which classify the content of the input, there are also those which explore classifying the stylistic content. In native language identification, models are tasked with identifying the writer's first (native) language from reading samples of their writings in English as a second language (Wong and Dras, 2011). Models are able to pick up on subtle differences in word choice, as well as spelling and grammar errors characteristic of each native language. In the PAN series of shared stylometric tasks, a variety of author profiling (Stamatatos et al., 2018; Rangel et al., 2020) and plagiarism detection (Stein et al., 2009) tasks are set, including style change detection

(Zangerle et al., 2020) which is expanded upon in Chapter 4 of this thesis – a task which involves identifying where the author changes in a document formed by combining the work of multiple authors.

**Natural Language Inference**

An example of a classification task which requires deep linguistic understanding is Natural Language Inference (NLI) where the objective is to label the logical relationship between a pair of sentences. This task is also known as Recognizing Textual Entailment (Cooper et al., 1996; Fyodorov, Winter and Francez, 2000; Bos and Markert, 2005; MacCartney and Manning, 2008). The first sentence of the pair is called the 'premise', and the second the 'hypothesis', and the model is tasked with classifying the relationship into one of 3 categories:

- *Entailment* - It is possible to infer the hypothesis from the premise.
- *Contraction* - The premise and the hypothesis are incompatible, i.e. the negation of the hypothesis would be labelled as *entailment*.
- *Neutral* - Everything else.

In order to succeed at NLI, the model must have a knowledge of linguistic phenomena including lexical entailment, coreference, tense, modality, quantification, belief, and lexical and syntactic ambiguity (Williams, Nangia and Bowman, 2017).

A key dataset used by many NLI approaches is the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) which consists of over 500,000 labelled sentence pairs in English - much larger than previous datasets used for this task. The data for this task includes image captions combined with ground-truth annotations generated by human crowd-sourced workers.

Its large size allows for many of the deep learning techniques commonly used in current NLP approaches.

Basing the dataset on the single genre of image captioning does however have its limitations. Image captions are short, simplistic and grounded in describing concrete visual ideas, with more abstract ideas such as referring to multiple time periods, and other modalities being extremely rare in this dataset. This limitation in dataset variety led to many models achieving scores similar to human performance, reducing the usefulness of the benchmark for evaluation.

To solve this, Williams, Nangia and Bowman (2017) introduced the Multi-Genre NLI Corpus (MNLI) which reproduces the scale and format of SNLI but expands the source data to include more domains. These domains from the Open American National Corpus (Ide, 2008) include spoken language transcriptions (two-person in-person dialogues, speeches, letters, press releases, reports, telephone conversations, non-fiction reports) as well as a collection of contemporary fiction from a variety of genres. As with SNLI, crowd-sourced workers are then asked to provide sentences for each label, as well as to label the sentence pairs created by other workers.

The MNLI dataset is used as one of the Natural Language Decathlon (decaNLP) benchmark tasks in Chapter 3.

### 2.1.2 Machine Translation

Machine translation is the task of translating a sentence from a source language into a target language.

This task makes extensive use of parallel text corpora - datasets where the same text is aligned in multiple different languages. Early datasets were based on parliamentary records, with the Canadian Hansard Corpus (Brown, Cocke et al., 1990) being one of the notable early examples.

With two official languages: English and French, the Canadian Parliament transcribes its debates in both languages. After alignment this provides a bilingual parallel corpus useful for training and evaluating machine translation models. Following from this, Europarl (Koehn, 2005), expanded this idea to the 11 languages of the European Parliament. Noting the need for wider language coverage, Ziemski, Junczys-Dowmunt and Pouliquen (2016) created the United Nations Parallel corpus which creates datasets of official records and other public-domain parliamentary documents in the six official languages of the United Nations. These languages - Arabic, Chinese, English, French, Russian, and Spanish allow evaluation of the ability of models to perform the more complex task of translating between languages which aren't part of the closely-related European language family.

With the increase in performance of models used to solve this task, there is a need for datasets which reflect the challenging, noisy, real-world use of language beyond sanitised transcripts of meetings and legal texts. To this end Lison, Tiedemann and Kouylekov (2018) introduced the OpenSubtitles dataset which is formed of aligned movie and TV subtitles from a large online repository. While still not completely natural language as this content is still scripted, it falls closer to colloquial language use than previous works.

With a wide variety of translation datasets available, a standardized benchmark was needed. The Workshop on Machine Translation (WMT), first introduced by Koehn and Monz (2006), runs shared tasks annually. It sets new challenges for teams to benchmark their models and frequently introduces new datasets - commonly extensions of existing popular datasets such as Europarl. Similarly, the International Conference on Spoken Language Translation (IWSLT) is related in scope with a focus on datasets which use spoken language, emphasising the unique properties of spoken language translation as distinct to written text translation (Akiba et al., 2004).

A complex problem for this and other sequence-to-sequence tasks is that of evaluation. Tra-

ditional metrics such a BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004), have been widely used for years to evaluate the quality of machine-generated translations. However, these and similar metrics such as Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005) have been criticized for their inability to capture semantic nuances and the quality of idiomatic expressions (Isozaki et al., 2010). These types of metrics commonly rely on n-gram matching, which can be insufficient for capturing the complexities of human language, resulting in a disconnect between the scores given by automatic metrics and human judges.

To address these limitations, researchers have proposed alternative evaluation metrics such as BERTScore (Zhang, Kishore et al., 2019) which leverages pre-trained contextual embeddings to evaluate the similarity between the generated and reference sentences. Rei et al. (2020) create a metric framework: Crosslingual Optimized Metric for Evaluation of Translation (COMET), which better aligns with human judgements by training a neural model. However, these metrics are not without their own limitations and biases, and the search for a universally accepted evaluation metric continues to be an open research question. Commonly, tasks such as WMT uses multiple automatic metrics alongside human evaluation (Kocmi et al., 2022) in order to mitigate some of these issues. In this thesis, I also use multiple metrics throughout, but due to the labour-intensive nature of human evaluation, I only evaluate the summarization output of the model in chapter 5 with human evaluation (summarization is a task which shares many of the same problems with evaluation metrics as translation).

While previous approaches to machine translation focused on individual sentences, many recent transformer models have begun to consider context at the document-level, improving the accuracy of translations. This approach requires a change in both datasets and metrics. The OpenSubtitles dataset is well-suited for this, as the individual subtitle lines form part of an entire film or TV show and thus have long context. The ContraPro annotations (Müller et al., 2018) augment

OpenSubtitles sentences where the use of pronouns create multiple possible translations which can only be disambiguated via the surrounding context. This challenging metric requires models to correctly pick between these possible translations.

In this thesis, Chapter 3 uses the IWSLT dataset as part of the decaNLP challenge. In Chapter 4, the opensubtitles dataset is expanded upon to form longer texts as part of the MuLD benchmark. This is then used to both train and evaluate our task decomposition model in Chapter 5, with the ContraPro annotations providing additional insights.

### 2.1.3 Summarization

Summarization is the task of taking an input and outputting a shorter version of the same document, keeping the main ideas intact (Radev, Hovy and McKeown, 2002) while creating a fluent output with little repetition (Moratanch and Chitrakala, 2017). This is an important task which enables users to find information quickly, especially in an age of rapidly growing internet use. Humans find summarization difficult and time-consuming, underlining the need for this to be performed automatically (Vilca and Cabezudo, 2017). There are 2 broad categories of summarization: extractive and abstractive. In extractive summarization, models construct their summaries by copying the most relevant phrases and sentences from the input text verbatim, while abstractive summarization involves models producing new text, distinct from the original document (Dernoncourt, Ghassemi and Chang, 2018).

Abstractive summarization is a more challenging task for models to solve. However, as abstractive summaries often occur naturally in many domains, the gold-standard targets for datasets are commonly abstractive. Early abstractive summarization datasets such as DUC (Over, Dang and Harman, 2007) and Gigaword (Rush, Chopra and Weston, 2015) consist of short snippets from news articles paired with single sentence summaries - commonly headlines. As models have grown more capable, datasets with more complex summaries have been created. The Cable

News Network/Daily Mail (CNN/DM) dataset (Hermann et al., 2015; Nallapati, Zhou, Gulcehre et al., 2016) consists of news articles scraped from the CNN and Dailymail websites, paired with summaries generated by human authors. In contrast to the single sentence summaries required by Gigaword, CNN/DM contains multiple bullet-pointed sentences, challenging models to produce fluent summaries keeping track of the current context while generating.

Before generative neural methods were widely used, extractive summarization was the dominant approach. It is possible to convert abstractive summaries into extractive ones. Various methods, including text-comparison metrics such as ROUGE (Nallapati, Zhou and Ma, 2016), or neural networks (Svore, Vanderwende and Burges, 2007), can be used to select the most representative sentences compared to the abstractive summary. Additionally, it is possible to compare the extracted summary produced by the model directly to the abstractive one via text-comparison metrics. However, it is likely that no combination of the sentences in the original document will match those in an abstractive summary (Jing, 2002).

Summarization is included as one of the decaNLP benchmark tasks in Chapter 3 by using the CNN/DM dataset. Summarization also forms part of the MuLD benchmark created in Chapter 4. The task decomposition model in Chapter 5 is evaluated on the BookSum dataset which is further discussed within the chapter itself.

### 2.1.4 Question Answering

Question Answering (QA) involves the model producing an answer to an input question. This can either be based on a reference document (Open QA) or completely generated using the model's learned knowledge (Closed QA). Like in summarization, Open QA systems can be further divided into those which simply quote the relevant parts of the reference document (Extractive QA), and those which generate free text based on the reference document in its own words (Abstractive QA).

Many QA datasets are created by starting with a high-quality dataset of reference texts and using crowd workers to pose and answer questions on these texts. The Standford Question Answering Dataset (SQUaD) (Rajpurkar, Zhang et al., 2016), is a commonly used open extractive QA dataset based on this concept. It is formed of 107,785 questions based on Wikipedia articles spanning a variety of topics with concrete entities such as people or places, as well as abstract concepts represented. The questions and answers were generated by providing crowd workers with paragraphs from the most popular Wikipedia articles and asking them to highlight spans of the paragraph which form the answer. This is in contrast to the earlier WikiQA dataset which only required selecting the correct sentence (Yang, Yih and Meek, 2015). The same questions were then answered by another worker to ensure consistency.

The SQUaD dataset was extended with SQUaD 2.0 which expands the original SQUaD dataset with unanswerable questions (Rajpurkar, Jia and Liang, 2018). Crowd workers were again used to generate these, and were specifically encouraged to pick unanswerable questions which look similar to the answerable ones. Workers were also encouraged to pick questions which have a plausible (but incorrect) answer in the reference paragraph, of the same type as required by the question.

SQUaD 2.0 is largely solved with multiple methods achieving scores above human performance on this task (Zhang, Yang and Zhao, 2020). Other datasets have extended the core methodology, increasing the difficulty. Kwiatkowski et al. (2019) argue that because of the artificial nature of questions generated by crowd workers, the unanswerable questions in SQUaD 2.0 can easily be answered without the required reasoning. They create a dataset based on real queries to the Google search engine to address this (an approach also explored with the WikiQA dataset from the Bing search engine on a smaller scale). Yang, Qi et al. (2018) explore making the QA problem more challenging via the introduction of multi-hop questions. These questions require the comprehension of multiple Wikipedia articles in order to produce the correct output to

questions such as "Nikolay Mitrofanovich Krylov and Anatoly Fomenko both held what academic title?" which requires reading the biographies of both Nikolay Mitrofanovich Krylov and Anatoly Fomenko and reasoning between them.

QA is also used as a frame for evaluating complex forms of reasoning. An example of this is the Winograd schema challenge (Levesque, Davis and Morgenstern, 2011), which serves as a successor to the famous Turing test (Turing, 1950) for evaluating intelligence. The WSC consists of questions such as "The trophy does not fit into the brown suitcase because it's too [small/large]. What is too [small/large]?". While trivial for humans, this task requires common-sense reasoning and world knowledge - in this case the purpose of suitcases and spatial reasoning.

QA features prominently in this thesis, as it is used both as a multitask learning 'supertask' in Chapter 3 as well as one of the standard NLP tasks used as a benchmark throughout this work.

## 2.2 NLP Methods

For many years, classical machine learning techniques were the dominant approach employed to solve language processing problems. Traditional approaches often relied on feature engineering, where domain-specific linguistic knowledge was used to extract relevant features from text data. These engineered features were then fed into algorithms such as Support Vector Machines (SVM) (Cristianini and Shawe-Taylor, 2000; Vapnik, 1995), Naive Bayes (Maron, 1961), and Decision Trees (Quinlan, 1993), among others. While these methods showed success in certain NLP tasks (Haffner, Tur and Wright, 2003; Gomes et al., 2017; Giorgos et al., 1999), they were limited by their inability to effectively capture complex patterns and the long-range dependencies present within language data. Additionally, hand-crafting features for each specific task is time-consuming and requires expert linguistic and domain specific knowledge, which limits the wide-spread adoption of these techniques across many areas.

Deep learning solves this problem with the use of (artificial) neural networks. Inspired by the way biological neurons are able to effectively learn new tasks in the human brain, neural networks model this process as a series of interconnected artificial neurons. Each neuron receives input signals, processes them, and generates an output. By composing these neurons into layers (typically an input layer, hidden layers, and a final output layer), we form an artificial neural network.

Mathematically, the operation of a single artificial perceptron (a type of artifical neuron) can be described as follows:

$$y = f(\sum_{i=0}^{n} W_i X_i + b)$$

Where

- $x_i$ represents the $i$-th input.
- $w_i$ denotes the weight of the input $x_i$, and represents how important that input is to calculating the neuron's output.
- $b$ represents a bias term which allows the neuron to shift its output, allowing it to learn to produce a specific output even when all inputs are 0. This is similar to how the intercept in linear regression is used.
- $f$ represents the activation function, which is applied to the previous terms and allows the neuron to represent non-linear relationships in the input – most non-trivial real-world problems contain such non-linear relationships. Typical examples of activation functions are sigmoid, tanh and ReLU (Glorot, Bordes and Bengio, 2011).

The weights and biases needed for a particular NLP problem are found in the training phase using a process called backpropagation (Rumelhart, Hinton and Williams, 1986a; LeCun, 1985).

In contrast to classical machine learning approaches, neural networks can avoid the need for manual feature engineering. By passing the raw input into the model, the neural network is able to both extract relevant features and produce a final output through the use of many layers.

## 2.3 Sequence-to-Sequence Models

Many tasks in NLP are sequence-to-sequence tasks – they take a string of tokens as input, and output a different string of tokens as the output. Examples of this kind of task include machine translation, automatic summarization, and question answering. Traditionally, these tasks have been solved using Recurrent Neural Network (RNN) encoder-decoder models (Graves, 2013; Bahdanau, Cho and Bengio, 2014b; Sutskever, Vinyals and Le, 2014; Nallapati, Zhou, Santos et al., 2016; Darapaneni et al., 2021).

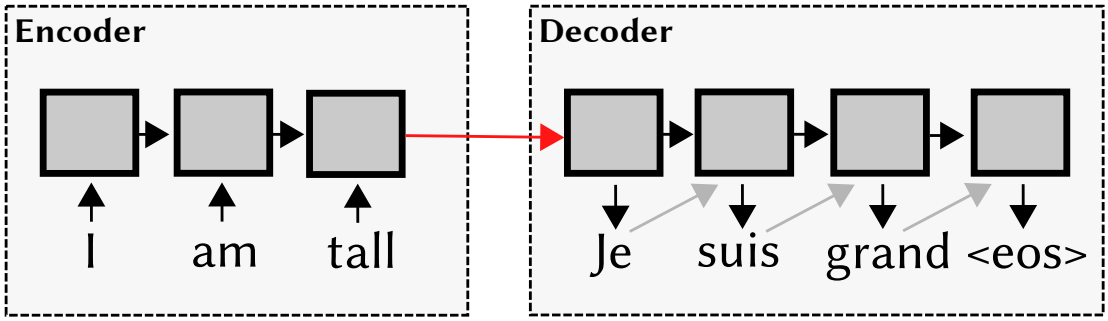



Figure 2.1: Example of an RNN performing English-French translation. The encoder and decoder each consist of a single recurrent unit, which are unrolled over time in this example. The context vector passing between the encoder and decoder is highlighted in red and is the key bottleneck of this approach.

RNNs are neural networks that include layers designed to process sequential data (e.g. sound, video, strings). They consist of recurrent units which have an internal state which is updated

with each input(Rumelhart, Hinton and Williams, 1986b). For processing language, this means that as every word of the input is passed in, the model updates its internal state. Varieties of RNN such as Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) differ in how they update and output this internal state.

As shown in Figure 2.1, for sequence-to-sequence tasks, an encoder RNN takes the input and compresses it into the internal state. This single state is then passed to the decoder which generates the output one token at a time (e.g. for translation this would be the same sentence in another language).

A problem with this approach is that all the important information from the input sequence must be represented solely in the fixed-size final state of the encoder (also known as the context vector). This harms the performance of the model when dealing with long input sequences, as information which is important at one step of the decoding may be irrelevant to later steps. To solve this, Bahdanau, Cho and Bengio (2014a) proposed the idea of attention. Instead of only using the final internal state of the encoder, we use a combination of all states. Attention allows us to weight the relative importance of these states (the amount of 'attention' to pay) for each output word (Figure 2.2).

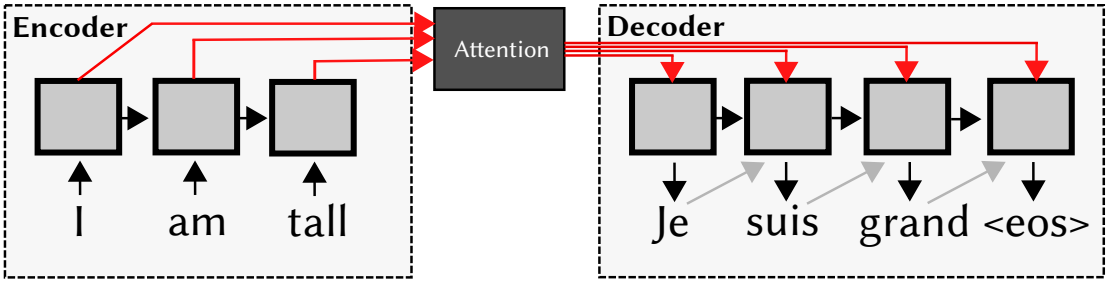



Figure 2.2: Example of an RNN using attention. The internal states from each timestep of the encoder are combined to produce a weighted average for each timestep of the decoder.

## 2.4 Transformers

While attention was initially used to augment existing recurrent networks, Vaswani et al. (2017), found that using attention alone can produce impressive results. A new model - the transformer, keeps the encoder-decoder structure, but uses a combination of simple feed-forward layers and self-attention in order to process the input.

The core innovation of the Transformer model lies in its use of a self-attention mechanism, which enables the model to focus on different parts of the input sequence during processing. Self-attention computes the weighted sum of the input embeddings, where the weights are determined based on the relevance or importance of each input position to other positions. The calculation of the self-attention mechanism is expressed mathematically as follows:

Given a sequence of input embeddings $X = \{x_1, x_2, ..., x_n\}$, the self-attention mechanism produces an output matrix $Y = \{y_1, y_2, ..., y_n\}$ through the following steps:

1. **Query, Key, and Value:** For each input position $i$, the self-attention mechanism generates three linear projections: query ($Q$), key ($K$), and value ($V$). These projections are obtained by multiplying the input embeddings $X$ with learned weight matrices $W_Q$, $W_K$, and $W_V$, respectively:

$$Q_i = X_i W_Q,$$
$$K_i = X_i W_K,$$
$$V_i = X_i W_V.$$

2. **Attention Scores:** The attention scores between position $i$ and position $j$ are computed by taking the dot product between the query vector of position $i$ and the key vector of

position $j$:

$$A_{ij} = \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right),$$

where $d_k$ represents the dimensionality of the key vectors. The softmax function normalizes the attention scores across all positions in the input sequence.

3. **Weighted Sum:** The final output embedding $y_i$ for position $i$ is calculated as the weighted sum of the value vectors of all positions, with attention scores serving as the weights:

$$y_i = \sum_j A_{ij} V_j$$

To allow the model to create multiple representations for each token, this self-attention procedure is performed multiple times in parallel - each instance is referred to as a self-attention head. These heads use their own weight matrices to allow each to focus on different parts and representations of the input sequence. The outputs of each self-attention head are then concatenated and merged with the help of another learnt weight matrix $W_O$. This forms the multi-headed attention block, which is the key component of the transformer model.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W_O$$

Unlike in recurrent models, there is no explicit order associated with the input tokens - we must add this as an extra time signal vector to the embedding ourselves with the use of a positional encoding. Vaswani et al. (2017) interweave sine and cosine functions:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Although there are other choices for positional embeddings (including the model learning its own embeddings), the advantage of building a positional embedding on sine and cosine functions is that they can scale to sequences of arbitrary length - even sequences longer than those that the model was originally trained upon. This is an important property for the efficient transformer models discussed in section 2.6.2 which need to process long text.

The model is built as an encoder decoder structure, with alternating multi-head attention and regular feed forward (fully connected) layers. Between these layers there are residual connections followed by layer normalisation which helps mitigate the vanishing gradient problem (Hochreiter, Bengio et al., 2001; Pascanu, Mikolov and Bengio, 2013), where the signal can get lost during back-propagation. This problem is especially noticeable when we have a very deep architecture with many layers, as is the case with the transformer.

Figure 2.3 shows how these layers are composed. The final output of the encoder is passed into a similarly structured decoder before the final output, which is a commonly simple linear layer.

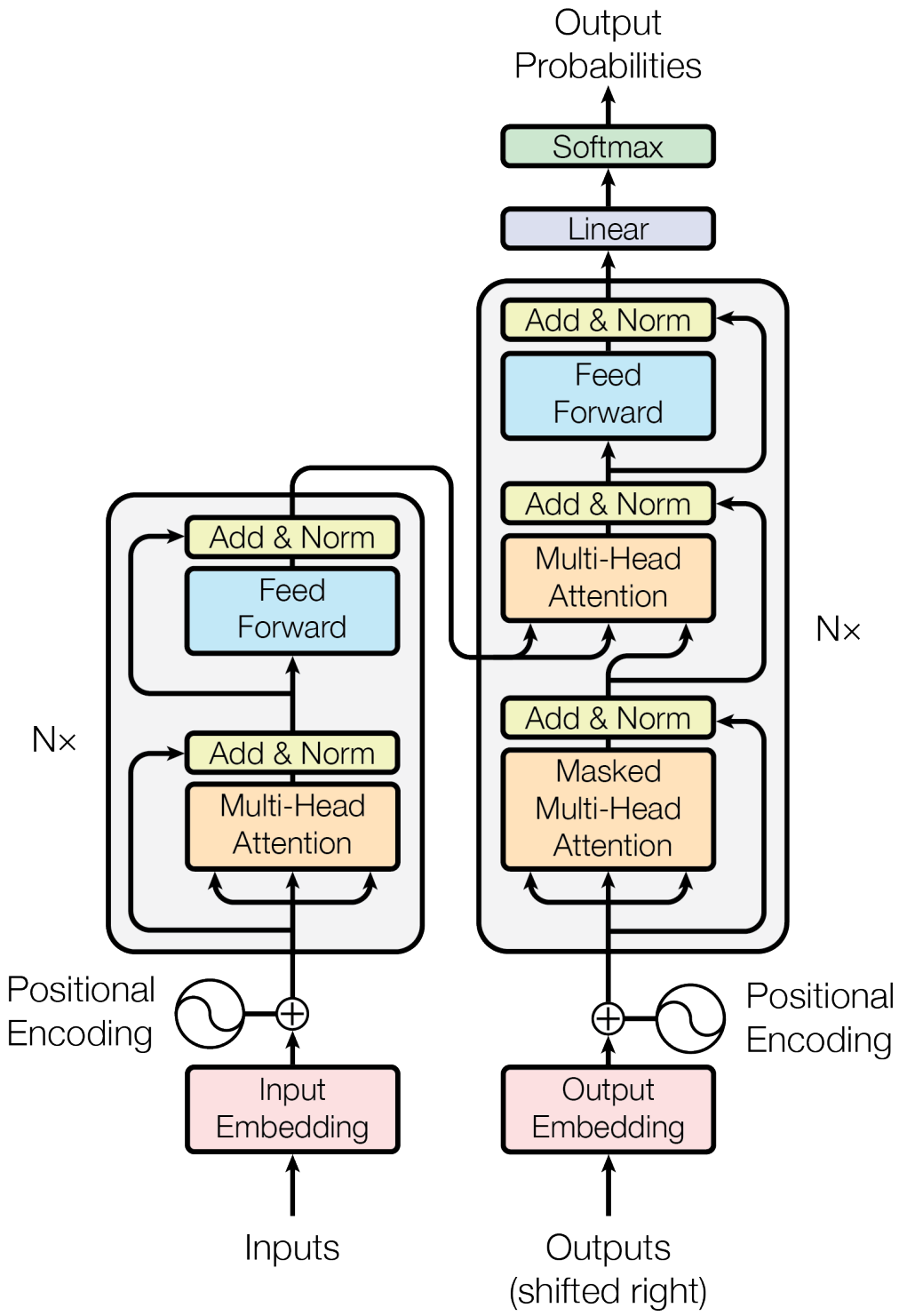



Figure 2.3: Overview of the transformer model. Diagram from Vaswani et al. (2017)

## 2.5 Language Models

James was a lonely Sailor. I spoke with the sailor because **[MASK]** was needing a friend.

Figure 2.4: Example of the language modelling task. Notice how in order to correctly fill in the masked token '[MASK]', the model must have knowledge of complex concepts such as co-referencing.

A key advantage of the transformer is that unlike the RNN, it can be easily parallelized as it stores no internal state. This allows the model to be made much deeper, and to be trained on much larger datasets. One task which has large datasets available is masked language modelling, as no labels are needed - just a large source of diverse text. There are multiple variations of this task (Devlin et al., 2019; Peters, Neumann, Iyyer et al., 2018), but in general we take some percentage of the words in the input (e.g. 15%) at random, and replace them with a [MASK] token. The language model is then trained to predict these missing tokens. Although it seems simple, a model which performs well at this task must have an understanding of grammar as well as more complex concepts such as co-referencing and named entities (Figure 2.4). With a scalable model such as the transformer and a large dataset of text, we can achieve strong performance on the language modelling task (Devlin et al., 2019).

While the language modelling task itself is of limited use, the knowledge it captures can be transferred to other tasks. When trying to solve another NLP task, we can begin with a pretrained language model and then fine-tune it for our new task. This allows us to transfer the knowledge from the language model trained on a large dataset, to a new task where a much smaller dataset is available. Using language models like this has led to dramatic performance increases across a wide range of NLP tasks[1]. In fact, in the GLUE benchmark containing 11 tasks representative of NLP, all the competitors use some kind of pretrained language model. As explored in Section

[1] https://gluebenchmark.com/

2.7.5, the new model prompting paradigm is based on language modelling.

The T5 model (Raffel et al., 2020) used in Chapter 3, Longformer model (Beltagy, Peters and Cohan, 2020) used in Chapter 4, and GPT-J-6B (Wang, 2021) model used in Chapter 5 are all pretrained on language modelling tasks.

In Chapter 6, inspired by the success of language modelling in NLP, I explore its analogue in computer vision, which I dub 'visual modelling'.

## 2.6 Long Documents and Efficient Transformers

The majority of work in NLP has focused on short documents (a few paragraphs or even single sentences). In fact many common NLP approaches can only accept limited length inputs - BERT for example, can only take 512 tokens as an input (Devlin et al., 2019).

How long a 'long document' has to be in order to be considered 'long' is rarely defined in the literature, however some consider any document longer than the small 512 token input length of models such as BERT to be 'long' - the Long Document Arena for example includes documents as short as 1024 tokens in length (Tay et al., 2021). In Chapter 4, I argue that this is much shorter than what would be considered a 'long document' in everyday speech, which motivates the need for benchmarks with much longer document lengths.

### 2.6.1 Benchmarks

Work on long-documents has focused on the question answering task, with a prominent dataset being NarrativeQA (Kočiský et al., 2018). This consists of film scripts and novels for which questions are crowdsourced, based on a short summary of the longer documents. The authors hope that questions based on a summary will force a model to capture long-term dependencies

in the full document. Other work has used summarization itself as a long document task, with Cohan et al. (2018) using summarization of academic papers into their abstracts as a dataset.

However, to fully validate models for long document NLP, we need a variety of datasets, the lack of which is a problem identified by numerous works (Kitaev, Kaiser and Levskaya, 2020; Kočiský et al., 2018). This is a problem I seek to address in Chapter 4.

The prototype for many NLP benchmarks is the success of the General Language Understanding Evaluation (GLUE) Benchmark which challenged models to solve 9 language understanding tasks including question answering, co-reference resolution, and sentiment analysis (Wang, Singh et al., 2018). The success of GLUE resulted in advancement to the point where a successor with more challenging tasks was required: SuperGLUE (Wang, Pruksachatkun et al., 2019). This approach has been replicated both across other languages (Yao et al., 2021), and other task types such as Natural Language Generation (Liu, Yan et al., 2021).

Recently, there have been some attempts to design benchmarks which test the ability of models to understand long documents.

The Long Range Arena was designed as a benchmark for evaluating the performance of efficient transformers on long input sequences (Tay et al., 2021). This benchmark challenges models to perform 6 synthetic and real-world multitmodal tasks on documents with up to 16,000 tokens. However, these tasks are all forms of classification, and as noted by Shaham et al. (2022), one of only two NLP tasks; LRA, uses byte tokenization as a way of artificially increasing the token count while having a much smaller number of words.

Guan et al. (2021) introduced a benchmark of long document Chinese tasks based on short stories. Four tasks were set including cloze tests, sentence position prediction, plot completion,

and story generation. However, there is a need for a benchmark focusing on English text, more conventional 'real-world' tasks, and crucially on much longer documents.

The QuALITY benchmark (Pang et al., 2021), was designed as a multiple-choice question answering dataset which selects questions which can't be answered by briefly skimming the text. Again, the maximum document length used: 6,000 is quite short in comparison to what would be considered a "long document" (e.g. legal texts, technical reports) in colloquial speech.

The most notable recent development is SCROLLS: Standardized CompaRison Over Long Language Sequences (Shaham et al., 2022) which used 7 long document datasets to create a benchmark with a range of input lengths with the median for most tasks (excluding the very long NarrativeQA) falling between 1000 and 10,000 words - which I argue is still too short to be a reliable evaluation of the longer transformer models such as the LED Longformer variant. Additionally, SCROLLS only focuses on question answering, summarization and NLI. This limits the type and length of the expected outputs to only short sentences or paragraphs.

In Chapter 4, I introduce a benchmark which includes a diverse set of NLP tasks and crucially document lengths much longer than these existing benchmarks.

### 2.6.2 Efficient Transformers

Multiple works have attempted to modify the transformer model to reduce the effect of document length on the time and memory complexity of the original model.

**Reformer** Kitaev, Kaiser and Levskaya (2020) proposed multiple optimisations to the transformer model to allow it to process longer inputs. The first of these optimisations is Locality Sensitive Hashing (LSH). When training a transformer, the attention requires comparisons of

the vectors in the previous layer with those in the next, using a cosine similarity measure. For a network with n nodes, this requires $O(n^2)$ comparisons.

To reduce the cost of these comparisons, they use a binning approach. Vectors are grouped into buckets based on their angle (with some probability), a technique known as Locality-Sensitive Hashing (LSH). Kitaev, Kaiser and Levskaya (2020) use a type of LSH called random projections. In this method, hyperplanes are created at random, then the vectors are assigned a bit depending on which side of each hyperplane it falls. We can then group vectors into bins with the same bit pattern. When comparing against a query vector, it is sufficient to only search within the same bin, reducing the self-attention complexity to $O(nlog(n))$. However, as noted by Wang, Li et al. (2020), this hides a large constant ($128^2$), and therefore is only a benefit when the sequence is very long.

Another issue when running transformers on long documents is the memory required to hold all activations of the hidden layers. To reduce this, ideas inspired by the work of Gomez et al. (2017) on RevNets are used to create a reversible transformer. This allows the activation at any point to be calculated from the activation of the final layer, removing the need to store intermediate values.

These modifications to the transformer allow an increase in the size of the input from 512 tokens (BERT), to around 64,000 tokens. While the authors tested their model on simple datasets such as WikiText (text compression), they don't show results on other more realistic tasks such as machine translation, and question answering.

**Longformer** Beltagy, Peters and Cohan (2020) propose a modification to the full self-attention pattern which is the root cause of the transformer's $O(n^2)$ computational and memory requirements. By only considering neighbouring tokens within a fixed distance (sliding window attention), introducing gaps in this window (Dilated sliding window), as well as allowing only a small

number of tokens to attend to every other (global attention), the requirements can be reduced ($O(n\sqrt{n})$) while still retaining much of the ability of the model to capture long-range dependencies. Other work such as the Sparse Transformer (Child et al., 2019), have followed a similar strategy of devising a sparse attention pattern.

Unlike Kitaev, Kaiser and Levskaya (2020), the longformer model was tested on a variety of real-world tasks including question answering and summarization. I use longformer as part of the baseline model on my long document benchmark in chapter 4.

**Linformer** Wang, Li et al. (2020) use low-rank factorization as a way to approximate self-attention, reducing its complexity. They provide theory and run experiments on both language modelling and classification to justify that the self-attention mechanism is low-rank allowing its approximation in linear time and memory complexity. This is shown to be $O(n)$ with respect to sequence length.

### 2.6.3 Task Decomposition

I propose that an alternative to developing models which can process a long input in a single pass, is a family of techniques based on task decomposition. In task decomposition approaches, a complex task is broken down into simpler tasks, which are each solved and then recombined to form the final output.

This is related to the divide-and-conquer paradigm, which is the basis of many solutions to key problems in algorithm design such as sorting (Hoare, 1961), efficient multiplication (Karatsuba and Ofman, 1962), and computing the discrete Fourier transform (Cooley and Tukey, 1965).

Similarly, there have been a variety of attempts in the machine learning field to use task-decomposition as a framework for breaking down complex tasks (Singh, 1992; Dayan and Hinton,

1992). This decomposition may take many forms, and the choice of decomposition approach is often highly task-specific.

For QA tasks, for example, we commonly have a question and a reference document which contains the answer. A query such as "Who has more singles titles, Brian Gottfried or Peter Fleming?" might be deconstructed into two distinct sub-questions: "How many singles titles does Brian Gottfried have?" and "How many singles titles does Peter Fleming have?". These can then be evaluated individually, yielding answers that can be directly compared ("25" and "3") to provide a final answer to the original question.

Task decomposition has also been applied to other domains. For instance in summarization, documents can be decomposed into sections, paragraphs, and sentences, before a summary is generated. This allows for more coherent and informative summaries for long input documents (Gidiotis and Tsoumakas, 2020; Zhang, Ni et al., 2021).

Similarly, in machine translation, the concept of breaking down long sentences into simpler clauses has been used to tackle complex examples (Sulem, Abend and Rappoport, 2020). When the sentence is decomposed into useful sections, this approach can reduce context loss and therefore improve translation accuracy.

Task decomposition can also be applied to other machine learning paradigms such as reinforcement learning where Hierarchical Reinforcement Learning is used to break down complex tasks into simpler, previously-learned primitives (Dayan and Hinton, 1992; Barto and Mahadevan, 2003; Nachum et al., 2019; Ranchod, Rosman and Konidaris, 2015; Hutsebaut-Buysse, Mets and Latré, 2022).

In a generalized approach, we can perform one of two actions at each stage of processing:

1. Respond: Providing an answer to the current task if it is straightforward and unambiguous.
2. Decompose: Breaking down the current task into multiple smaller, simpler sub-tasks that can be tackled more easily.

In the case of summarization, Wu, Ouyang et al. (2021) apply this to long documents, showing that a deterministic algorithm can be used to perform the decompose function. This simplifies the procedure, only requiring a model which can provide the correct responses to the summarization subtasks. In Chapter 5, I generalize this concept, creating a model which can be applied to any long document NLP task.

## 2.7 Multitask Learning

Traditionally, machine learning models are trained to perform well on a single task in isolation. This differs greatly from how humans learn new tasks - we rely on prior experience to solve related problems. For instance, a baby first learning to recognize faces can also apply this knowledge to identifying other kinds of objects (Ruder, 2017). Multitask learning seeks to emulate this process by training models to solve multiple objectives simultaneously.

It has been shown that across multiple domains: computer vision (Girshick, 2015), speech recognition (Deng, Hinton and Kingsbury, 2013), and in natural language processing (Collobert and Weston, 2008), multitask learning is an effective way of combining a wide range of the knowledge relevant for solving tasks in a specific field.

Formally defined, multitask learning (MTL) involves a set of $n$ machine learning tasks $T_1, T_2, \ldots, T_n$ which contains subsets of related tasks. The challenge is to train a single model, $M$ which minimises the losses, $L$ on each of the tasks:

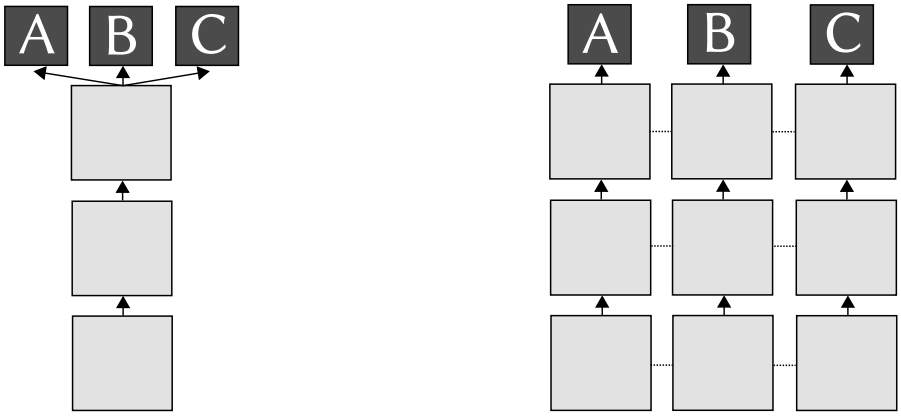
$$M = \arg\min \sum_{i=0}^{n} L(T_i, M)$$

The objective is to incorporate knowledge from all tasks with the aim of outperforming models trained on each task in isolation, leading to better generalisation and overall performance. However, there is a possibility of an effect known as 'negative transfer' or 'destructive interference' where if irrelevant information is shared between tasks, multitask learning can harm rather than improve performance. The choice of MTL method as well as careful task choice, training order and other optimisations is important in mitigating this phenomenon.

Multitask learning is closely related to transfer learning - another approach which aims to use the knowledge learned by one task to aid in learning others. Transfer learning is however a sequential process. First the model is trained on a secondary task which the model is able to learn well (either because it is easier, or has large datasets available). The model is then trained on the primary, target task with the aim of transferring the knowledge gained to this new task. In contrast, multitask learning trains on all tasks simultaneously and the aim is to improve performance on all tasks - not just a single primary task.

There are two broad multitask learning approaches: *soft parameter sharing* and *hard parameter sharing* as shown in Figure 2.5. Hard parameter sharing encompasses techniques where a subset of the model weights are directly reused across multiple tasks, such that they are trained to jointly minimise the loss functions of each of the tasks. In soft parameter sharing, each task uses its own parameters (akin to a separate model for each task), but these are constrained using techniques such as regularization (such as $l_2$ or trace norm) to favour similar values (Duong et al., 2015; Yang and Hospedales, 2017). This keeps the distance between the corresponding parameters small, encouraging each sub-model to learn a solution which is also useful for solving the other

tasks. Many of the approaches used for multitask deep learning fall somewhere in-between these categories, using ideas from both.




(a) Hard-Sharing (b) Soft-Sharing

Figure 2.5: Multitask sharing schemes

This thesis focuses on a specific form of hard-sharing referred to as 'Task Reformulating', a family of techniques where multiple tasks are incorporated into a single model by converting them into a joint 'supertask'. I devote Section 2.7.5 to discussion of this and the related paradigm of prompt-based learning.

### 2.7.1 Shared Backbone

By far the most common model design, the shared backbone (also known as Shared Trunk (Crawshaw, 2020), or Simple Hard Parameter Sharing (Ruder, 2017)) is the typical hard-sharing based approach as shown in Figure 2.5a. In this, the inputs from all tasks are fed into a single sequence of shared layers which extracts features common to all tasks. This is followed by separate output 'heads' for each task, each head learning how to make use of the extracted features to solve a specific task.

While this approach successfully forces the model to learn how to extract common features, this comes at the disadvantage of sensitivity to negative transfer if any of the tasks require substantially different features. This can be somewhat mitigated by the use of very large models which have sufficient capacity to learn features required for all tasks.

### 2.7.2 Cross-stitch Networks

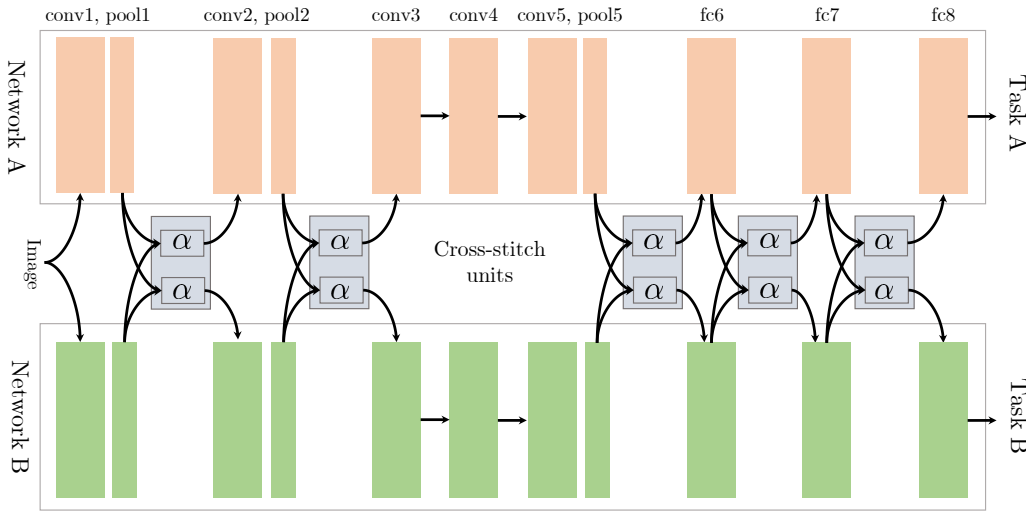



Figure 2.6: Structure of a Cross-Stitch Network. Figure from Misra et al. (2016)

While having a single shared feature extraction core can be successful, Misra et al. (2016) take an approach inspired by soft parameter sharing. In Cross-Stitch Networks, Misra et al. (2016) provide a mechanism for combining the representations learned by individual task-specific subnetworks, allowing the network to leverage the strengths of each task to improve overall performance.

In a cross-stitch network, the input to the network is processed separately by multiple subnetworks, each dedicated to a specific task. These subnetworks learn task-specific representations by

optimizing their respective objective functions. However, instead of keeping the representations isolated within each subnetwork, cross-stitch units are introduced to enable communication and collaboration between the subnetworks.

The cross-stitch units facilitate the exchange of information between the subnetworks at pre-specified layers of the network. This is achieved by learning a set of weights that determine the extent to which the representations from each subnetwork should be combined. The weights are learned during training using backpropagation, where the gradients flow through the cross-stitch connections.

Each cross-stitch unit (Figure 2.6) is placed in-between the layers of each task-specific network and learns the optimal linear combination of the previous layer outputs to pass as the inputs to the next layer. Formally, given the activations $x_{\mathrm{A}}, x_{\mathrm{B}}$ of tasks A and B from a layer in the network, the cross-stitch unit learns a matrix $\alpha$ which to produce the linear combinations: $\tilde{x}_{\mathrm{A}}, \tilde{x}_{\mathrm{B}}$ which are used as the inputs for the next layer in each network. Specifically, the cross-stitch unit learns,

$$\begin{bmatrix} \tilde{x}_{\mathrm{A}}^{ij} \\ \tilde{x}_{\mathrm{B}}^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{BA} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{A}}^{ij} \\ x_{\mathrm{B}}^{ij} \end{bmatrix}$$

for every $(i, j)$ in the output. The values of the $\alpha$ matrix dictate how much of the information from one task is passed to the other. For example, if $\alpha_{AB}$ is 0, then no outputs from B are used in the input to the next layer of A and the networks for each task are independent. If the model learns a higher value, a shared representation is used. In this way, the model is able to learn

when it is useful to share features between tasks (usually when the tasks are highly related), and when not to.

The cross-stitch connections can be seen as a way to jointly optimize the objectives of multiple tasks, promoting both task-specific learning and shared representation learning. By allowing information to flow across tasks, cross-stitch networks encourage the emergence of representations that capture both the commonalities and differences among the tasks, leading to enhanced overall performance.

One advantage of cross-stitch networks is their ability to effectively handle tasks with varying levels of relatedness. Tasks that share similar underlying structures or concepts can benefit from sharing representations, while tasks with distinct characteristics can maintain their task-specific representations. This flexibility makes cross-stitch networks suitable for multitask learning scenarios where tasks exhibit varying degrees of overlap.

There is however, a limit to this flexibility, as the location of the cross-stitch units must be specified in advance.

### 2.7.3 Hierarchical Approaches

Hierarchical approaches make explicit use of the theorized relationships between tasks, where some tasks (e.g. named entity recognition) require simple 'low-level' reasoning, and others (e.g. relation extraction) build on this to enable deeper, more complex understanding. These relationships are usually mirrored in hierarchical approaches, where layers close to the input are used to solve the low-level tasks, and high-level tasks are output after more layers (Sanh, Wolf and Ruder, 2018; Hashimoto et al., 2017). However, the hierarchical pipeline direction may not always be from low-level to high-level tasks. Alqahtani, Mishra and Diab (2020) for example, experiment with starting with the higher level tasks (POS tagging, word segmentation), and then doing the lower level diacritization task using later layers in the model.

### 2.7.4 Sluice Networks

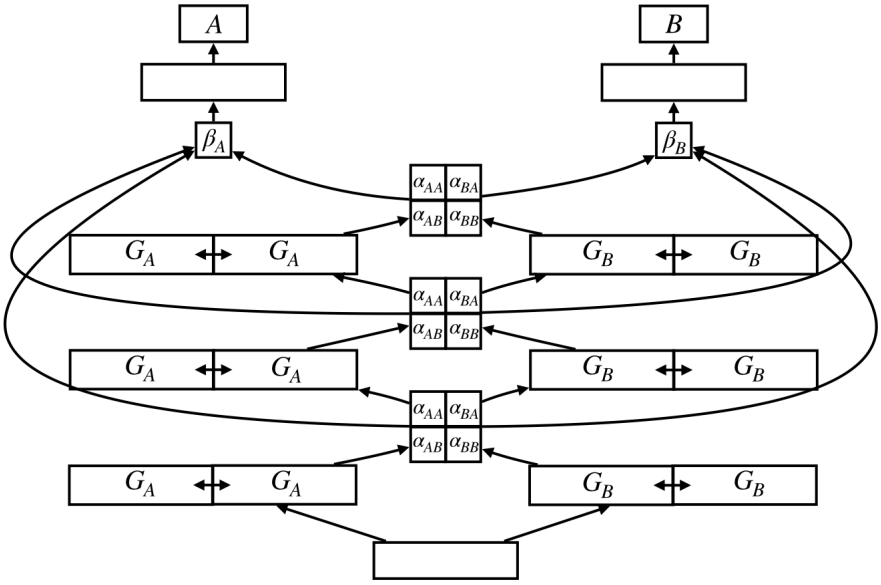



Figure 2.7: Structure of a Sluice Network. Figure from Ruder et al. (2017)

Ruder et al. (2017) generalize all these ideas with sluice networks which are able to learn which layers (or parts of layers) are useful for solving each task.

Figure 2.7 shows how this architecture is organised. As in cross-stitch networks, we have a subnetwork for each task. However, the parameters for each layer are divided into two subspaces, $G_1$ and $G_2$ which learn task-specific and shared features respectively. To keep the subspaces orthogonal (to prevent learning redundant features), the Frobenius norm of the product of the two subspaces is added to the loss function as an auxiliary term.

The inputs to the next layer are formed of a linear combination of the previous layer's task specific output and the shared outputs, mediated through a learnt $\alpha$ matrix which controls how much information is shared.

This general architecture can represent hard sharing (if all $\alpha$ values are set to a constant), and cross-stitch networks (if all the $\alpha$ values for the two subspaces are identical, and $\beta$ is 0 apart from the last layer - ignoring the skip connections). By varying the parameters at different depths, this model can also represent a hierarchical approach.

### 2.7.5 Task Reformulation

An alternative approach to modifying the model architecture involves modifying the format of the input data, reformulating each task in terms of a single "supertask". This allows for an extreme form of hard sharing by using a single model which only has to solve the supertask without any task-specific parameters.

Examples of commonly-used supertasks include question answering, span extraction, and more recently - language modelling. Table 2.2 shows how a range of NLP tasks can be reframed in terms of these supertasks.

Keskar et al. (2019) propose unifying NLP under the supertask of span-extraction. Many tasks already fit into this paradigm naturally, such as extractive QA and summarization. Tasks such as classification and regression can be incorporated by appending the possible output options to the input (See Table 2.2 for examples). However, there are limitations with this choice of supertask. Because the output tokens must appear somewhere in the input, tasks such as translation where the input and output have very different vocabularies are impossible. Additionally, tasks such as story generation based on an initial prompt cannot be solved using this span extraction framing.

The T5 model (Raffel et al., 2020) unified 8 tasks from the GLUE and SuperGLUE NLP benchmarks into a single text-to-text model. Each input is prepended with a short instruction such as “translate English to German:”, or “cola sentence:” which indicates to the model which task is

| Task | Supertask | Format |
|---|---|---|
| Sentiment Classification | Span Extraction | I love the product 0.0 0.35 0.5 0.75 **1.0** |
| | QA | What is the sentiment of the sentence, "I love the product."? |
| | Language modelling | [INST] «SYS»You are a sentiment classifier for product reviews.«/SYS»<br>review: I love the product.<br>[/INST] |
| Summarization | Span Extraction | **An American woman died** aboard a cruise ship... |
| | QA | What is the summary? An American woman died aboard a cruise ship... |
| | Language modelling | [INST] «SYS»You are a summarizer for news articles.«/SYS»<br>article: An American woman died aboard a cruise ship...<br>[/INST] |
| Translation | Span Extraction | – |
| | QA | What is the translation from English to French? An American woman died aboard a cruise ship... |
| | Language modelling | [INST] «SYS»You are an expert English to French translator«/SYS»<br>input: An American woman died aboard a cruise ship...<br>[/INST] |
| QA | Span Extraction | Nikola Tesla (10 July **1856** – 7 January 1943) was... [SEP] What year was Tesla born? |
| | QA | What year was Tesla born? Nikola Tesla (10 July 1856 – 7 January 1943) was... |
| | Language modelling | [INST] «SYS»Answer the question using the context«/SYS»<br>Question: What year was Tesla born?<br>Context: Nikola Tesla (10 July 1856 – 7 January 1943) was...<br>[/INST] |

Table 2.2: Examples of framing common NLP tasks as different supertasks. For span extraction, I illustrate spans which form part of the output in **bold**.

requested. However, these instructions are not fully grammatical sentences but short sequences of keywords, limiting the model's ability to respond to novel tasks.

The decaNLP challenge (McCann, Keskar et al., 2018) solves this by casting all tasks as open question answering. In contrast to short keyword instructions, each task is framed using fully grammatical questions such as "What is the summary?" or "Is this review positive or negative?", along with a context document and the model is tasked with providing an answer. McCann, Keskar et al. (2018) were amongst the first to propose demonstrate indicating the task to the model via natural language, and therefore I use decaNLP as a way to explore the robustness of these task reformulations to paraphrasing in Chapter 3.

Originally considered too challenging by McCann and Keskar (2019), language modelling has since been shown as a viable "supertask" for many NLP problems. Radford et al. (2019a) introduced gpt2, a 1.5B parameter transformer model which produced state-of-the-art results on 7 NLP tasks. The authors observe that by framing tasks in a similar way to how they are observed naturally in the training set, the model is able to solve such tasks without specific finetuning. For example, adding the text *TL;DR:* to the input causes the model to produce a summary. A key advantage of this choice of supertask is the lack of supervised data required - many of the capabilities of the model can be learnt via unsupervised language modelling.

Chapter 6 of this thesis proposes visual modelling: a vision parallel to language modelling as a multimodal "supertask" uniting both NLP and vision. This follows from the success of language modelling as a "supertask" for NLP.

In the last year, the idea of task reformulation has evolved into the recent explosion of prompt-based learning, which use language modelling on prompt-answer pairs as the "supertask". In this paradigm, the model is trained to follow an instruction describing the task and provide a response.

By reformulating the task into a form closer to examples found in the training data, the need for task-specific finetuning can be reduced, or even eliminated (Scao and Rush, 2021). The prompts can also be more detailed and nuanced than possible with the single phrase or question used by previous approaches, improving performance. Mishra et al. (2021) for example, construct detailed prompts which contain things to avoid, positive and negative examples, along with the task definition.

## 2.8 Paraphrasing

Paraphrasing is often defined as 'sameness of meaning' (Vila, Martí and Rodríguez, 2014). This however is ambiguous as there are many degrees of 'sameness', and the boundary between paraphrasing and other phenomena (e.g. co-reference, inference) is often unclear.

From a theoretical linguistics perspective, many frameworks explore the idea of paraphrasing. In meaning-text theory, language is understood as a mapping from the semantics (meaning) of a sentence to its written form (phonetics) (Mel'čuk, 1992). Intermediate mappings, such as syntactic and morphological representations, are joined by rules detailing the possible transformations from one level of representation to the next. The choice of transform at each layer of representation, as well as the equivalences between representations at the same level, give rise to paraphrasing.

In NLP, paraphrasing is generally studied from a machine learning perspective, with notable interest surrounding paraphrase identification for plagiarism detection (El Desouki and Gomaa, 2019; Hunt et al., 2019; Altheneyan and Menai, 2020). Recent advances in language models have shown state-of-the-art performance on this task and the related task Natural Language Inference (Yang, Zhu et al., 2019; Devlin et al., 2019). The standard corpus used for evaluation is the Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005), which consists of annotated

pairs extracted from news articles. Quora Question Pairs (QQP)[2] is a larger dataset, formed of questions submitted to the website Quora, and is often used for training models (Imtiaz et al., 2020; Tomar et al., 2017; Li, Jiang et al., 2019).

Additionally, various methods have been developed to make models more robust to paraphrasing their input (Ribeiro, Singh and Guestrin, 2018; Minervini and Riedel, 2018; Iyyer et al., 2018). Many of these methods consist of automatically generating variations of the input, feeding each into the model, then ensembling the answers. Dong et al. (2017) perform this via back-translation, while Buck et al. (2018) explore an approach based on an agent which has been trained using reinforcement learning to reformulate the input to maximise the performance of the final model. Iyyer et al. (2018) build an NLI model which uses a dataset augmented with adversarial paraphrase examples, an approach which is limited by the lack of large high-quality paraphrase datasets.

In Chapter 3, I explore methods of improving robustness via multitask learning as well as training models on more varied inputs.

### 2.8.1 Paraphrase Typologies

To better categorise paraphrase phenomena, typologies can be constructed based on the understanding of paraphrasing from different fields - primarily theoretical linguistics, discourse analysis, and computational linguistics.

Discourse analysis provides a view on paraphrasing which explores the user's intent rather than specific linguistic mechanisms. Cheung (2009) describes paraphrases of definitions via high level concepts such as "exemplification" (adding an example) and "refining" (narrowing the scope of the definition). For NLP applications, this high-level analysis doesn't provide meaningful insight

[2] www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

| | | |
|---|---|---|
| Morpholexicon-based changes | Morphology-based | Inflectional Changes<br>Modal-verb Changes |
| | Lexicon-based | Spelling Changes<br>Same-polarity Substitutions<br>Synthetic/analytic Substitutions |
| Structure-based changes | Syntax-based | Ellipsis<br>Coordination Changes |
| | Discourse-based | Punctuation Changes<br>Sentence Modality Changes |
| Miscellaneous changes | | Change of Order<br>Addition/Deletion |

Table 2.3: Summary of the Vila (2013) paraphrase typology.

into how small changes such as replacing words with synonyms affect the performance of trained models.

In computational linguistics, typologies are often formed as lists of very specific paraphrase mechanisms, grouped into general classes for use in a particular application. Defined at such a low level, these are incomplete descriptions of paraphrasing and cannot be easily transferred to other languages.

Vila Rigat et al. (2011) developed a typology specifically with NLP applications in mind. Their hierarchical approach has been used to tag plagiarism corpora (Barrón-Cedeño et al., 2013), and the influential Microsoft Research Paraphrase Corpus (MSRPC-A) (Vila, Bertran et al., 2015). The typology consists of 20 paraphrase types and is hierarchical, where paraphrase types are grouped by the level that the change occurs (e.g. morphological, lexical, semantics). Vila (2013) also use the typology to annotate the Spanish corpus - Relational Paraphrase Acquisition from Wikipedia (WRPA), showing that it can generalise to non-English paraphrases. Below is an overview of the types from the typology relevant to this thesis, illustrated with examples using the dataset created in Chapter 3. Paraphrases will often mix multiple of these types, which are summarized in Table 2.3.

**Inflectional Changes** - changing the inflectional affix of words.

(a) Is the review *positive* or *negative*?

(b) Does this review show *positivity* or *negativity*?

**Modal Verb Changes** - Changing modality via the use of modal verbs. In the example below, the modal verb *have to* in (a) is exchanged for *must* in (b).

(a) You *have to* find the summary.

(b) You *must* find the summary.

**Spelling Changes** - format and spelling changes of the lexical units. This includes: case changes, abbreviations, and digit/letter alternations. In the example below, the acronym SQL in (a) is changed to the full term, Structured Query Language in (b).

(a) What is the translation from English to *SQL*?

(b) What is the translation from English to *Structured Query Language*?

**Same-polarity Substitutions** - changing one lexical/functional unit with another of the same meaning. This includes phenomena such as synonyms and substituting general with specific terms.

(a) Is the *review* positive or negative?

(b) Is the *movie review* positive or negative?

**Synthetic/analytic substitutions** - swapping a synthetic for an analytic structure. This includes phenomena such as compounding, and lexically emptied specifier additions.

(a) Summarize the document.

(b) Provide a summary of the document.

**Ellipsis** is the removal of sentence elements which can be recovered through linguistic mechanisms. In the example, while 'the review' appears in both clauses or (b), it only appears the first time in (a).

(a) Is the review positive or negative?

(b) Is the review positive or *is the review* negative?

**Coordination changes** - changes where one of the paraphrases contains a coordination (syntax linking two or more elements together) and this is changed or removed in the other paraphrase. In the example below, the two sentences in (a) are joined via the disjunctive coordination *or* in (b).

(a) Is the review positive? negative?

(b) Is the review positive or negative?

**Punctuation changes** - any change in the punctuation (not lexical units) in a sentence. In the example below, the list of possible answers in (a) is numbered in (b). These numberings don't add further information, just emphasize that the sentence contains a list.

(a) Is the review positive or negative?

(b) Is the review (1) positive or (2) negative?

**Sentence modality changes** - Modifying the modality of a sentence, e.g. from a question (a) to an imperative (b):

(a) What is the summary?

(b) Find the summary.

**Change of order** — swapping the order of some sentence element, e.g. the order of items in a list

(a) Is this review *positive* or *negative*?

(b) Is this review *negative* or *positive*?

**Addition/deletion** - adding or removing lexical/functional units.

(a) Is this review positive or negative?

(b) *Hmm*, is this review positive or negative?

These paraphrase types are used to annotate the PQ-decaNLP dataset of paraphrased questions created in Chapter 3, which serves as a method of evaluating the robustness of models to different kinds of paraphrasing of their input.

## Epilogue

In this chapter I explored the tasks, datasets, and techniques which underpin the rest of this thesis. As I focus on multitask learning, the NLP tasks detailed in Section 2.1 are used throughout the subsequent chapters. Chapter 3 draws from the paraphrasing work discussed in Section 2.8, while Chapters 4 and 5 address the issues of long document datasets and models described in Section 2.6.

# Chapter 3

# Ask Me in Your Own Words: Investigating Paraphrasing for Multitask Question Answering

This chapter explores how multitask models which are trained on the decaNLP benchmark fail with simple paraphrasing of their question prompt. The contents are mostly drawn from my paper "Ask Me in Your Own Words: Investigating Paraphrasing for Multitask Question Answering" published in PeerJ.

The main contribution is: PQ-decaNLP, a crowdsourced corpus of paraphrase questions covering 7 of the decaNLP tasks (Figure 3.3b). Importantly, I annotate the PQ-decaNLP corpus using the paraphrase typology of Vila, Bertran et al. (2015), allowing analysis of the specific types of paraphrase phenomena which cause the model to fail. I find that simple transformations such

as changing the label order and altering the sentence modality harm the performance, which is a serious hurdle for any zero-shot learning applications.

I also explore methods for improving the robustness to paraphrasing of models, including the addition of paraphrase generation/paraphrase detection as tasks. Knowledge learnt from adding these new tasks does not transfer to increased robustness in other tasks, but training models using PQ-decaNLP questions greatly improves the robustness to paraphrasing.

## 3.1 Introduction

Recent progress in Natural Language Processing (NLP) has led to improved performance across a wide range of language understanding problems[1]. A key component of these advances is the use of knowledge transferred from other tasks, most prominently from language modelling (Peters, Neumann, Iyyer et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019)

McCann, Keskar et al. (2018) developed a new NLP benchmark: the Natural Language Decathlon (decaNLP). This challenges a single model to perform ten Natural Language Understanding (NLU) tasks by framing each task as question answering (Figure 3.1). For example, when solving a translation task, a model is asked to "Translate from English to German", given a paragraph on English text as the context, and is expected to output the translation of the context in German as the answer. The key appeal of this task design is that it favours models where all parameters are shared between all tasks and adding new tasks only requires additional training data, not redesigning the model. As well as decaNLP, McCann, Keskar et al. (2018) proposed the Multitask Question Answering Network (MQAN) as a neural network architecture for solving the ten decaNLP tasks. More recently, models such at T5 (Raffel et al., 2020) have explored a similar text-to-text paradigm.

[1] www.gluebenchmark.com/leaderboard

Is this review positive or negative?
What is the summary?
Translate from English to German
What is the translation from English to SQL?
What is the change in state?

Figure 3.1: Examples of fixed questions for decaNLP

It's important to note that these questions (some of which are in fact instructions) are not intended to be simple wake-up words (WUWs), but as fully grammatical sentences. A major motivation for decaNLP is the ability to support zero-shot learning. By forming a new task as question answering, MQAN can solve tasks which are variations of the ones it was trained on - For example, MQAN trained to detect sentiment using the question "Is the review positive or negative?" should also be able to answer similar questions for related tasks such as "Is the sentence happy or is it angry?". This requires models to understand the meaning contained within the questions rather than just treating them as simple WUWs. I suggest that robustness to paraphrasing is a key first step (necessary, but not sufficient) in making the idea of more general zero-shot learning feasible within the decaNLP paradigm.

I also envisage real-world applications where a user can interact with a system by asking questions about some context document in order to solve a task. Outside an academic context where the prompt can be tightly controlled, a user should to be able to express the task they want the system to solve naturally. This could also become part of a voice assistant, where a user asking "Translate this for me", or "What's the translation?" are both equally valid.

In this chapter, I systematically analyse the robustness of text-to-text models to paraphrasing, as well as explore techniques to improve it via adding paraphrase detection and paraphrase generation as additional tasks. For this purpose, I contribute a crowdsourced corpus[2] of paraphrased questions: PQ-decaNLP.

[2]Available at `www.github.com/ghomasHudson/paraphraseDecanlpCorpus`

More importantly, I annotate the PQ-decaNLP corpus using a paraphrase typology, allowing new analysis of the specific types of paraphrase phenomena which cause the model to fail. I find that the performance is significantly harmed by simple transformations such as exchanging the order of words and changing the sentence modality from questions to imperative commands, but that training using PQ-decaNLP questions greatly improves the robustness to paraphrasing.

## 3.2 DecaNLP

The decaNLP challenge (McCann, Keskar et al., 2018) frames multiple tasks as question answering, and is an extreme case of hard parameter sharing where all the parameters are shared (without any task-specific parameters). This approach has key advantages, primarily that new tasks can be added without any modification to the model architecture, only requiring changes to the dataset to frame the task as a question. Table 3.1 shows the 10 tasks included in decaNLP. Each one uses standard, publicly-available datasets and metrics. These metrics are summed to give an overall 'decaScore'.

| Task (Dataset) | Metrics | Question | Context | Answer |
|---|---|---|---|---|
| Question Answering (SQuAD) | nF1 | What causes precipitation to fall? | In meteorology, precipitation is any product of condensation that falls under gravity... | gravity |
| Translation (IWSLT) | BLEU | Translate from English to Geman | Most of the planet is ocean water. | Der Großteil der Erde ist Meerwasser. |
| Summarization (CNN/DM) | ROUGE | What is the summary? | Harry Potter star Daniel Radcliffe gains access to a reported £320 million fortune... | Harry Potter star Daniel Radcliffe gets £320M fortune... |
| NLI (MNLI) | EM | Hypothesis: Some men are playing a sport. | A soccer game with multiple males playing. | Entailment |
| Sentiment Analysis (SST) | EM | Is this sentence positive or negative? | The product was a pleasure to use. | Positive |
| Semantic Role Labelling (QA-SRL) | nF1 | What is equipped to do something? | Ballast tanks are equipped to change a ship's trim. | Ballast tanks |
| Relation Extraction (QA-ZRE) | cF1 | Who was in charge of Kraków? | The current President of Kraków, is Jacek Majchrowski. | Jacek Majchrowski |
| Dialogue State Tracking (WOZ) | dsEM | What is the change in dialogue state? | Are there any French restaurants? | food: French |
| Semantic Parsing (WikiSQL) | lfEM | What is the translation from English to SQL? | The table has column names... Who is the player that wears number 42? | SELECT Player FROM table WHERE No. = 42 |
| Commonsense Reasoning (MWSC) | EM | Who feared violence? councilmen or demonstrators? | The city councilmen refused the demonstrators a permit because they feared violence. | councilmen |

Table 3.1: Tasks included in decaNLP with example questions, contexts, and answers. EM=Exact Match, nF1=Normalised F1, cF1=corpus-level F1, dsEM=dialogue state EM, lfEM=logical form EM

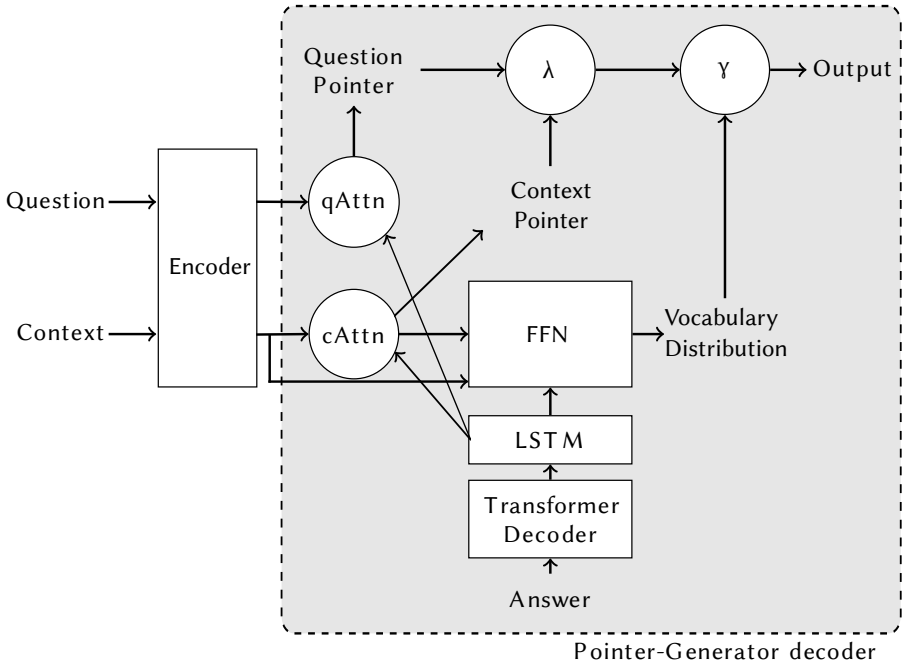



Figure 3.2: MQAN model overview. It is structured as an encoder-decoder model, with a pointer-generator decoder chooses between copying from the question, context, or an external vocabulary without explicit supervision.

Figure 3.2 shows the model proposed by McCann, Keskar et al. (2018) to solve decaNLP: the MQAN (Multitask Question-Answering Network). This network is based on encoder-decoder models for abstractive question answering, notably employing a pointer-generator mechanism for creating the output, a technique commonly applied to summarisation (See, Liu and Manning, 2017). Pointer-generator networks (See, Liu and Manning, 2017) allow a model to construct the answer from both words contained in the input, and words in an external vocabulary. The choice of when to use vocabulary words or when to point to words in the question is not explicitly supervised but is freely learnt by the model. This has been applied to the summarisation task, where it is useful for a model to copy quotes from the source material as well as vocabulary words to create a natural sounding summary. As decaNLP uses a separate question and context

sequence, McCann, Keskar et al. (2018) generalise the pointer-mechanism to copy from the question, context, and vocabulary. This is a modification particularly important for decaNLP tasks where the question may contain the class labels (E.g. "Is this review *positive*, or *negative*?"), the context can contain key phrases (as in summarisation), or words can only be in the vocabulary (as in translation). The encoder of the model uses a BiLSTM encoder, dual co-attention, and self-attention to encode the question and context sequences ensuring that long-term dependencies are captured and information is shared between the sequences. The full details of this model can be found in McCann, Keskar et al. (2018).

Raffel et al. (2020) build on decaNLP to explore a similar text-to-text paradigm using a transformer model trained using simple keyword prompts (e.g. "summarize:", "cola sentence:"). This model (named T5) was constructed after a series of experiments comparing different architectures, unsupervised objectives and multitask-learning strategies. The final model organizes its transformer blocks in an encoder-decoder structure, pretraining it using a BERT-style denoising objective. This model achieved state-of-the-art performance on 18 NLP tasks.

## 3.3 Methodology

I present the methodology for the two parts of my work:

1. A new dataset: PQ-decaNLP, which I use to analyse how the existing models perform when provided with paraphrased questions.

2. Proposed improvements to the model training to increase the performance on paraphrased questions.

Who was hurt? Bob or Charlie?
↓
{mainQuestion} {choice1} or {choice2}?

(a)

What is the synopsis?
Give me a condensed version
Sum up the article
What would be a good summary of the article?
Sum it up.

(b)

Figure 3.3: (a) Template transformation for the Modified Winigrad Schema Challenge (b) Examples from the paraphrase corpus for the summarisation task

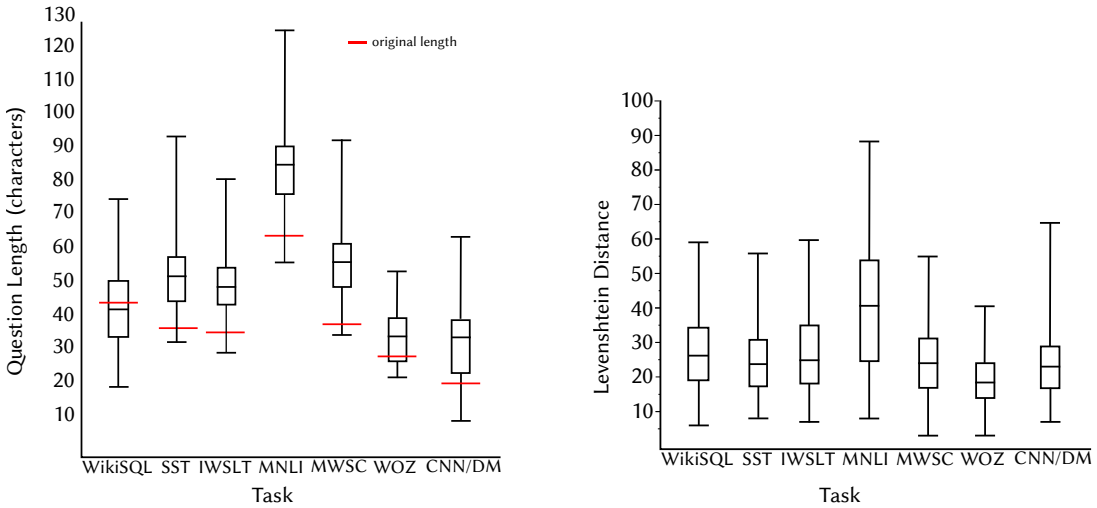
### 3.3.1 The PQ-decaNLP dataset

I create a paraphrased version of decaNLP questions: PQ-decaNLP, using the crowdsourcing platform Amazon Mechanical Turk[3]. Workers were given a description of a decaNLP task and were asked to provide 5 paraphrases of the fixed question. For tasks which have instance-specific information, for example NLI, where the hypothesis is embedded in the questions, I transform them into generic templates (Figure 3.3a).

Of the 10 decaNLP tasks, I limit my work to the 7 tasks which have a fixed question template, removing Question Answering, Semantic Role Labelling, and Zero-Shot Relation Extraction where the question differs for every example. Techniques for improving the robustness of these excluded tasks are not decaNLP-specific and have been widely studied elsewhere (Fader, Zettlemoyer and Etzioni, 2013; Bordes, Chopra and Weston, 2014; Dong et al., 2017).

These were then inspected using the open source project LanguageTool[4] for spelling and grammar mistakes. Additionally, I removed paraphrases which did not preserve the meaning of the original question, were ungrammatical, or were duplicates, via a manual review. 73.1% of the paraphrases were accepted, rejecting 3.7% due to grammatical errors, 2.9% due to duplication, and 20.3% which were not paraphrases of the original.

[3] www.mturk.com
[4] www.languagetool.org




(a) Paraphrase corpus lengths. The original question length is highlighted in red.

(b) Paraphrase corpus edit distances (Levenshtein)

Figure 3.4: Summary of PQ-decaNLP statistics

Figure 3.3b shows examples of paraphrases for the summarisation task. I collect 100 paraphrases per task to ensure a variety of paraphrases types while minimising duplication. The resulting 700 paraphrases (100 per task) are split 70/30 into train/test sets. Figure 3.4a shows the distribution of question lengths for the paraphrase corpus compared with the original decaNLP questions. The majority of the paraphrases are longer than the original fixed question, suggesting that authors tend to add complexity when paraphrasing, contrary to the findings of Barrón-Cedeño et al. (2013). This may be because the decaNLP questions are already simplistic in comparison to the more complex sentences from Project Gutenberg[5] books as used in the work of Barrón-Cedeño et al. (2013). Figure 3.4b shows edit distances, with paraphrases of MNLI differing most from the original question.

For evaluating the models, I define the PQ-decaScore as the sum of the task-specific metrics for 7 tasks that I consider on the PQ-decaNLP dataset, similarly to how the decaScore of McCann,

[5] www.gutenberg.org

Keskar et al. (2018) is defined over the full set of 10 tasks. I reuse the train/test splits from the original decaNLP datasets, replacing the fixed questions with randomly selected paraphrased questions.

**Annotation**

To gain an understanding of exactly which kinds of paraphrasing reduce the performance of the models, the PQ-decaNLP test set is hand-annotated using the typology of Vila, Martí and Rodríguez (2014) described in Section 2.8.1. As my dataset exclusively contains questions and imperative statements, only a subset of the paraphrase phenomena as shown in Figure 3.5 are observed. Same-polarity and Addition/Deletion are the most frequent phenomena in my dataset, confirming the similar findings of Vila, Bertran et al. (2015) on the P4P, and MSRP-A datasets. There are no examples of Negation, and Opposite-polarity, but higher frequencies of sentence modality change.

### 3.3.2 Proposed Improvements

Investigating the performance of the models on the PQ-decaNLP paraphrase questions, the results show lower scores across all tasks, indicating the models are not robust to paraphrasing of the question. These results and analysis are presented in section 3.4.

To enhance the robustness of the models, I propose several improvements. As my focus is the exploration of the existing models, I restrict my scope to modification of the data (adding/modifying decaNLP tasks) rather than the model architectures themselves.

For all my experiments, I use the top-performing version of the Multitask Question Answering Network (MQAN) presented in the original work of McCann, Keskar et al. (2018). This model is trained on all ten decaNLP tasks using an anti-curriculum strategy, where the model is first

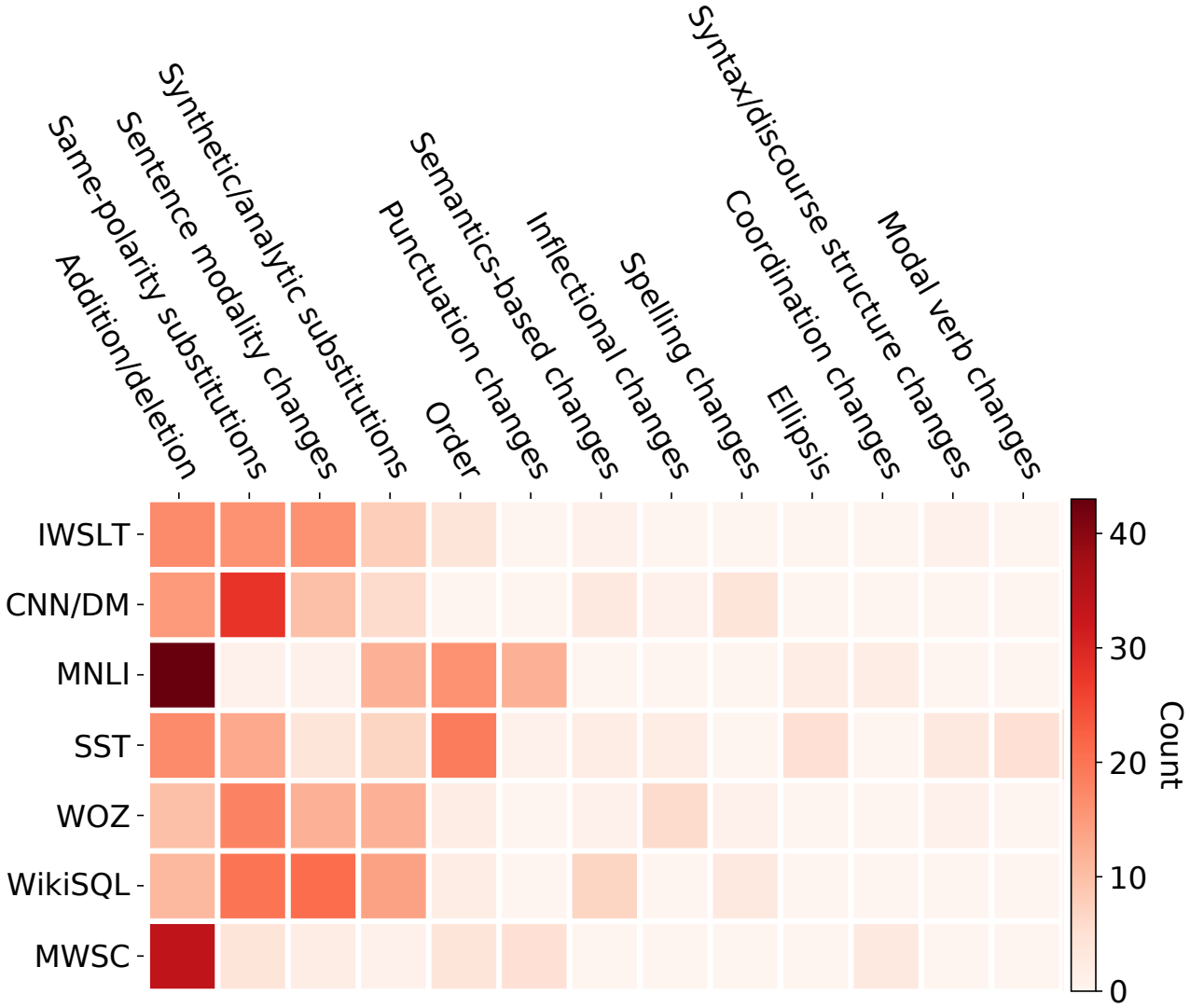



Figure 3.5: Counts of the paraphrase phenomena occurring in the test set

trained on the SQuAD dataset alone (phase one) before sampling batches "round-robin", from all the tasks in a fixed order (phase two). I use the t5-base version of the T5 model, pretrained on the C4 corpus and finetuned following the procedure in Raffel et al. (2020).

**Training on PQ-decaNLP**: My first method is to directly train the model on PQ-decaNLP. For each example in the decaNLP training set, uniformly distributed random selections are used to pick a question from the PQ-decaNLP training set to replace the fixed question. This directly trains the model to consider different paraphrases of the question.

**Adding paraphrase tasks** Secondly, I propose to exploit the multitasking abilities of the models by adding a new task to indirectly teach the model about paraphrasing in general. To do this, I introduce a paraphrase detection task (identifying whether 2 sentences are a paraphrase pair), or a paraphrase generation task (generating a paraphrase of the given sentence).

Introducing a new task rather than changing the data of existing tasks has the advantage of preserving the ease of extending decaNLP to additional tasks in the future. Using this approach, new tasks can still be added with fixed questions as before. Only a dataset of general paraphrase pairs is needed.

For paraphrase detection, the model is asked the question: "*[paraphraseCandidate1] – paraphrase, or nonparaphrase?*", and provided with *[paraphraseCandidate2]* as the context (where *[paraphraseCandiate1]* and *[paraphraseCandiate2]* are the two sequences in a possible paraphrase pair). I expect that, similar to the existing decaNLP tasks of SST and NLI, the model will learn to select the output classes 'paraphrase' and 'nonparaphrase' from the question.

Paraphrase generation is framed as a sequence-to-sequence task using the question: " *What is the paraphrase?*" and *[paraphraseCandidate1]* as the context. I train the model with the target of *[paraphraseCandidate2]*.

Additionally, I experiment with variants of these tasks which don't explicitly instruct the model to perform paraphrase detection/generation in the question (The 'Without task information' setting). For detection, I use the question: *[paraphraseCandidate1]*, context: *[paraphraseCandidate2]* and train the model to output 'yes' or 'no' from the external vocabulary. This directly trains the model to paraphrase the entire question. For paraphrase generation, I use *[paraphraseCandidate1]* as the question with a blank context. I train the model to generate *[paraphraseCandidate2]*. These are not valid decaNLP tasks but can be used as pretraining tasks, with the task specific information given by the answer the task is supervised on.

Table 3.2 shows examples of these formulations with a sample paraphrase pair.

| Detection | Generation |
|---|---|
| **With task information** | |
| Question: "How do you start a bakery?"– paraphrase, or nonparaphrase? | What is the paraphrase? |
| Context: "How can one start a bakery business?" | "How do you start a bakery?" |
| Answer: "paraphrase" or "nonparaphrase" | "How can one start a bakery business?" |
| **Without task information** | |
| Question: "How do you start a bakery?" | "How do you start a bakery?" |
| Context: "How can one start a bakery business?" | |
| Answer: "yes" or "no" | "How can one start a bakery business?" |

Table 3.2: Example of paraphrase question formulation

## 3.4 Results and Discussion

To examine the robustness to paraphrasing, I evaluate on the PQ-decaNLP dataset. In Table 3.3 a decrease in scores across many tasks is observed, with the MQAN model showing the largest decreases in performance for IWSLT, MNLI, CNN/DM, and SST, and the T5 model with the CNN/DM and MWSC tasks.

Trained on the original decaNLP dataset, the T5 model outperforms MQAN, suggesting a transformer-based language model is better suited to the decaNLP task. Additionally, the T5

| | Dataset | Base | With task information | | Without task information | | PQ-decaNLP trained |
|---|---|---|---|---|---|---|---|
| | | | Detection | Generation | Detection | Generation | |
| | | | | **MQAN** | | | |
| **decaNLP** | IWSLT | 13.7 | 14.8 | 14.5 | 13.0 | 14.7 | 15.9 |
| | CNN/DM | 24.6 | 24.5 | 25.1 | 24.7 | 24.0 | 24.6 |
| | MNLI | 69.2 | 71.3 | 70.9 | 69.2 | 71.3 | 71.0 |
| | SST | 86.4 | 86.6 | 84.7 | 85.1 | 86.4 | 86.2 |
| | WOZ | 84.1 | 81.8 | 84.0 | 83.6 | 87.2 | 84.2 |
| | WikiSQL | 58.7 | 63.3 | 57.3 | 60.8 | 65.2 | 65.6 |
| | MWSC | 48.4 | 40.2 | 40.2 | 51.2 | 36.6 | 45.1 |
| | decaScore | 385.1 | 382.5 | 376.7 | 387.6 | 384.4 | 392.6 |
| **PQ-decaNLP** | IWSLT | 2.8 | 4.5 | 3.1 | 5.7 | 4.8 | 14.4 |
| | CNN/DM | 8.4 | 10.8 | 10.3 | 11.1 | 7.5 | 22.5 |
| | MNLI | 43.4 | 16.1 | 42.6 | 20.2 | 48.3 | 69.9 |
| | SST | 23.4 | 21.0 | 72.7 | 23.2 | 32.7 | 84.6 |
| | WOZ | 71.2 | 66.4 | 62.0 | 65.3 | 52.3 | 78.5 |
| | WikiSQL | 55.8 | 26.8 | 41.0 | 28.1 | 54.9 | 62.9 |
| | MWSC | 40.0 | 26.5 | 36.7 | 19.2 | 26.4 | 38.1 |
| | PQ-decaScore | 245.0 | 172.1 | 268.4 | 172.8 | 226.9 | 370.9 |
| | | | | **T5** | | | |
| **decaNLP** | IWSLT | 31.8 | 30.0 | 31.6 | 31.5 | 31.8 | 31.0 |
| | CNN/DM | 31.9 | 32.0 | 32.5 | 32.0 | 32.1 | 32.3 |
| | MNLI | 76.8 | 76.4 | 76.4 | 76.7 | 77.3 | 75.8 |
| | SST | 91.2 | 90.0 | 90.8 | 89.9 | 89.9 | 87.2 |
| | WOZ | 85.3 | 83.7 | 80.8 | 81.9 | 82.4 | 80.8 |
| | WikiSQL | 58.9 | 60.0 | 58.5 | 59.5 | 59.5 | 56.5 |
| | MWSC | 54.9 | 50.0 | 50.0 | 48.8 | 45.1 | 48.8 |
| | decaScore | 430.8 | 422.1 | 420.6 | 420.3 | 418.1 | 412.4 |
| **PQ-decaNLP** | IWSLT | 25.6 | 24.7 | 22.3 | 25.8 | 21.3 | 31.3 |
| | CNN/DM | 22.7 | 19.6 | 21.3 | 22.5 | 20.4 | 25.3 |
| | MNLI | 73.3 | 72.6 | 72.6 | 73.3 | 74.1 | 75.5 |
| | SST | 89.9 | 89.7 | 89.8 | 89.5 | 89.5 | 87.9 |
| | WOZ | 75.1 | 79.6 | 78.9 | 76.8 | 70.3 | 79.1 |
| | WikiSQL | 57.9 | 58.4 | 57.2 | 57.9 | 57.9 | 56.0 |
| | MWSC | 44.2 | 40.5 | 35.5 | 39.6 | 34.3 | 48.1 |
| | PQ-decaScore | 388.7 | 385.1 | 377.6 | 385.4 | 367.7 | 403.2 |

Table 3.3: Validation metrics for decaNLP and PQ-decaNLP datasets: I show paraphrase detection and generation in settings which indicate the task in the question (with task information), and for those where task information is only indicated by the supervision (without task information). The last model is trained only on PQ-decaNLP questions (PQ-decaNLP trained). I report different metrics for each task as described in Table 3.1. The decaScore and PQ-decaScore reported here are the sum of the task specific metrics.

model is more robust to paraphrasing, only loosing 42.1 of its total score when compared to the 140.1 lost by MQAN.

I find that MQAN trained on PQ-decaNLP (PQ-decaNLP trained) reduce this drop across all tasks except MWSC. I hypothesise that the lack of improvement in MWSC is because the original question: "*{mainQuestion} {choice1}* ***or*** *{choice2}*" already varies greatly between examples in the dataset - only the word 'or' separating the two choices is constant. For WikiSQL, I also find an improvement of 6.9 lfEM on the original dataset, suggesting that this task benefits from more varied questions.

When adding paraphrase detection or generation as an additional task, the MQAN model is able to learn these new tasks with 85.7 f1, and 31.4 bleu scores. I find that while the tasks have little impact on the performance of the original decaNLP data (some scores are slightly higher), they perform worse than the original MQAN on PQ-decaNLP. This suggests that the knowledge learnt about paraphrasing does not help the robustness to paraphrasing of MQAN. Adding these new tasks significantly harms the performance of the T5 model on the original decaNLP data.

To better understand how the models behave on paraphrased questions, I conduct a range of analysis. I find only a weak negative correlation between the edit distance of the paraphrase (compared with the original question) and the score (R=-0.2373 for MQAN). This suggests that while many paraphrases which deviate further from the original question perform worse, other factors such as the type of paraphrase may also be significant.

| Paraphrase Type | IWSLT | CNN/DM | MNLI | SST | WOZ | WikiSQL | MWSC |
|---|---|---|---|---|---|---|---|
| | | | **MQAN** | | | | |
| Addition/deletion | -1.0/0.0 | -0.4/1.6 | -13.8/ 0.1 | 0.3/0.2 | -5.6/3.7 | 5.4/0.0 | -0.3/1.5 |
| Same-polarity substitutions | -1.9/0.0 | 2.2/1.1 | 17.8/-0.2 | -5.5/-0.3 | -0.6/-5.7 | 3.9/0.0 | 0.6/-5.7 |
| Sentence modality changes | **-4.1**/-0.1 | 1.9/-0.8 | 2.3/ 0.0 | 5.3/-0.3 | -1.1/0.7 | **-7.6**/0.1 | 1.6/-1.4 |
| Synthetic/analytic substitutions | 0.1/0.0 | -0.1/1.0 | -0.2/-0.1 | -5.3/0.3 | -8.5/4.7 | 4.5/0.0 | -3.5/-7.7 |
| Order | -3.2/-0.1 | - | **-38.7**/-0.1 | **-29.2**/-0.3 | 10.1/4.2 | 5.1/0.1 | 2.4/4.5 |
| Punctuation changes | - | - | 10.7/ 0.1 | -15.2/-1.4 | - | - | -5.0/-0.7 |
| Semantics-based changes | -2.8/0.0 | -3.9/1.0 | - | -11.8/-0.1 | -3.6/3.9 | 4.4/-0.1 | - |
| Inflectional changes | - | 0.0/0.9 | - | -9.9/-0.5 | -0.1/-6.1 | - | - |
| Spelling changes | - | -2.5/-2.6 | - | - | 9.9/4.4 | 5.2/0.0 | - |
| Ellipsis | - | - | -5.9/ 0.2 | -14.5/-0.6 | - | - | - |
| Coordination changes | - | - | 19.0/ 0.1 | - | - | - | -7.4/-3.3 |
| Syntax/discourse structure changes | -2.9/0.0 | - | - | 56.0/0.4 | -3.9/0.2 | - | - |
| Modal verb changes | - | - | - | 2.9/-0.1 | - | - | - |
| | | | **T5** | | | | |
| Addition/deletion | -1.2/0.0 | -0.1/0.0 | 1.3/1.4 | 0.0/0.2 | -3.9/-0.3 | -0.1/0.0 | 0.0/-1.2 |
| Same-polarity substitutions | 3.2/0.0 | 0.0/-0.1 | 2.4/-1.5 | 0.0/-0.2 | -3.2/-0.2 | 0.0/0.0 | -4.2/-0.3 |
| Sentence modality changes | **-2.7**/-0.0 | -0.4/-0.1 | -1.4/0.0 | -0.7/0.2 | -6.3/-0.6 | **0.0**/0.0 | -4.2/4.7 |
| Synthetic/analytic substitutions | -3.3/-0.1 | -0.1/0.1 | -0.1/0.2 | -0.2/0.2 | 2.8/0.7 | -0.1/0.0 | 2.2/0.7 |
| Order | 0.7/-0.1 | - | **0.3**/0.2 | **-0.3**/-0.1 | 1.7/0.1 | -0.1/-0.1 | -4.2/-6.6 |
| Punctuation changes | - | - | -1.6/0.0 | 0.5/0.3 | - | - | -3.6/-5.6 |
| Semantics-based changes | -9.4/0.0 | 0.1/0.0 | - | -0.7/-0.2 | 8.8/0.0 | 0.2/0.0 | - |
| Inflectional changes | - | 0.2/0.0 | - | -0.2/0.2 | -3.7/-0.8 | - | - |
| Spelling changes | - | 0.2/-0.1 | - | - | 9.7/0.7 | 0.3/0.0 | - |
| Ellipsis | - | - | 3.2/1.1 | -0.1/0.2 | - | - | - |
| Coordination changes | - | - | -12.3/-3.6 | - | - | - | -7.5/-10.1 |
| Syntax/discourse structure changes | 1.2/0.0 | - | - | 0.6/0.0 | -6.1/2.5 | - | - |
| Modal verb changes | - | - | - | -0.8/-0.2 | - | - | - |

Table 3.4: Average difference in performance when paraphrase phenomena are present. Where the value is negative, paraphrases which contain this phenomena perform worse than those without. Entries marked with '-' indicate where the task does not contain any examples of that type. I present the results X/Y where X is the performance difference on the original model, and Y is the performance difference of the PQ-decaNLP trained model.

### 3.4.1 Impact of Paraphrase Phenomena

Table 3.4 shows the average difference in performance for paraphrases where a paraphrase phenomenon is present compared to those where the phenomenon is not present [6].

Paraphrased questions which contain an 'Order' annotation perform worse in classification tasks with fixed labels (MNLI, SST) than other paraphrases when using MQAN. The 'Order' tag occurs in 73% of the sentiment analysis task (SST), primarily in the swapping of the class labels 'positive' and 'negative'. I find that manually swapping the class labels back to the same order as the original question, increased the performance by 38.2 (to 61.6), suggesting the model is memorising the position of the labels rather than their semantics. This sensitivity to label order harms the ability of MQAN to perform zero-shot learning. The T5 model looses less performance on MNLI and SST, with 'Order' paraphrases no longer causing the largest decrease in performance, suggesting the trained T5 model is much more robust to label position and is comprehending their meaning.

An interesting aspect of the original decaNLP framing is that English to SQL translation is formed as a question ("What is the translation from English to SQL?"), where English to German translation is framed as an imperative command ("Translate from English to German."). Table 3.4 shows that MQAN performs especially poorly on WikiSQL and IWSLT paraphrases which contain a change in sentence modality. Inspecting the answers for these WikiSQL cases reveals that the model outputs German words, indicating confusion between English-SQL translation and English-German translation. This suggests the model overly relies on the indicators of sentence modality ("What is the", "translate"/"translation") rather than the source and target languages. Again we see that the T5 model is more resilient to changes of sentence modality.

I find that the models trained on PQ-decaNLP have a smaller range of performance, suggesting they perform similarly across all paraphrase types.

[6] Analysing the performance in this way mitigates the effect of phenomena frequency.

I find no correlation between the number of phenomena present and the score.

### 3.4.2 Analysing Pointers

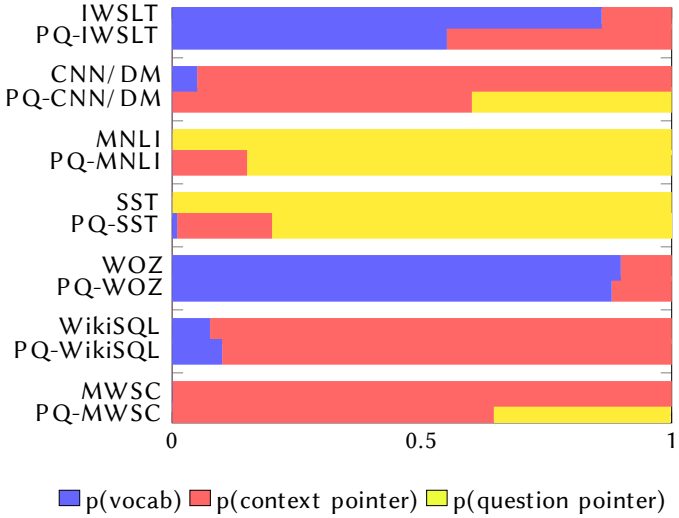



Figure 3.6: Comparison of where MQAN selects output from. The top bar for each dataset shows the pointer values on the original decaNLP dataset, while the bottom one shows the values when evaluated on PQ-decaNLP.

To gain a better insight into why MQAN fails, I analyse where the model copies its answers from: the question, the context, or the external vocabulary, which I present in Figure 3.6. A unique property of MQAN, the model's learned pointers allow us to better understand how it makes errors - Does the model pick words from the wrong source indicating that it is confused about what kind of task it is being asked to solve, or simply pick an incorrect word when solving the correct task? I confirm the findings of McCann, Keskar et al. (2018) that the MQAN does not confuse tasks when asked the original decaNLP questions. However, when evaluated on the paraphrased questions we see more confusion. For the classification tasks (MNLI, SST, MWSC), where the class labels are contained within the question, we see a decrease in copying from the question. For translation (IWSLT), we see an increase in copying from the context, and for

semantic parsing (WikiSQL) we see an increase in copying from the external vocabulary. These indicate the confusion between these tasks.

## 3.5 Conclusion

In this work, I explore how robust text-to-text models are to paraphrasing of questions asked. I introduce a diagnostic corpus annotated with paraphrase phenomena and show how simple transformations such as changing the label order and altering the sentence modality can harm the performance. I believe that the creation of similar typology-annotated corpora will provide useful insights into the robustness to paraphrasing of many models across NLP.

Additionally, I find that training models on paraphrased questions improves its robustness to paraphrasing. I find that knowledge learnt from adding the tasks of paraphrase generation or paraphrase detection does not transfer to increased robustness in other tasks for either model.

I hope that the paraphrase corpus of decaNLP questions will encourage further research into more robust multitask question answering models.

## Epilogue

In this chapter, I explored how multitask models can be made more robust to paraphrasing of their input. Next, I begin to look at the extension of multitask models to longer documents, developing a benchmark of long NLP tasks which tests the ability of multitask models to solve problems where the inputs are over 10,000 tokens long.

# Chapter 4

# MuLD: The Multitask Long Document Benchmark

Impressive progress in NLP techniques has been driven by the development of multitask benchmarks such as GLUE and SuperGLUE. While these benchmarks focus on tasks for one or two input sentences, there has been exciting work in designing efficient techniques for processing much longer inputs.

In this chapter, I present MuLD: a new long document benchmark, consisting of only documents over 10,000 tokens in length. By modifying existing NLP tasks, I create a diverse benchmark which requires solutions to successfully model long-term dependencies in the text. I evaluate how existing models perform, and find that my benchmark is much more challenging than the 'short document' equivalents. Furthermore, by evaluating both regular and efficient transformers, I show that models with increased context length are better able to solve the tasks presented,

suggesting that future improvements in these models are vital for solving similar long document problems. This is a problem I tackle in chapter 5.

This chapter is based on the following publication: Hudson, G. Thomas, and Noura Al Moubayed. "MuLD: The Multitask Long Document Benchmark" in *LREC 2022*.

## 4.1 Introduction

Pretrained language models have been highly influential in Natural Language Processing (NLP), leading to state-of-the-art results across a wide range of tasks. Based on the transformer architecture, these language models have shown capable at text classification, question answering, and translation among many other NLP problems.

The rise of pretrained language models in NLP has been driven by influential benchmarks such GLUE (Wang, Singh et al., 2018), and SuperGLUE (Wang, Pruksachatkun et al., 2019), which combine multiple existing datasets to provide a standardised evaluation of general-purpose Natural Language Understanding (NLU) approaches. Similar benchmarks have been created to evaluate multilingual approaches (Yao et al., 2021), and other types of NLP task such as Natural Language Generation (NLG) (Liu, Yan et al., 2021).

However, a key component of the success of the transformer model - self-attention - is also a major limitation when it comes to processing longer sequences. Comparing each token to all other tokens in the previous layer yields a $O(n^2)$ complexity, limiting the ability of standard transformers only a few hundred or thousand tokens on standard hardware. With many real world tasks involving the need to process documents in the range of tens of thousands of tokens, this is an important problem to solve.

Recently, approaches such as Longformer (Beltagy, Peters and Cohan, 2020), Reformer (Kitaev, Kaiser and Levskaya, 2020), and Linformer (Wang, Li et al., 2020) have explored techniques for

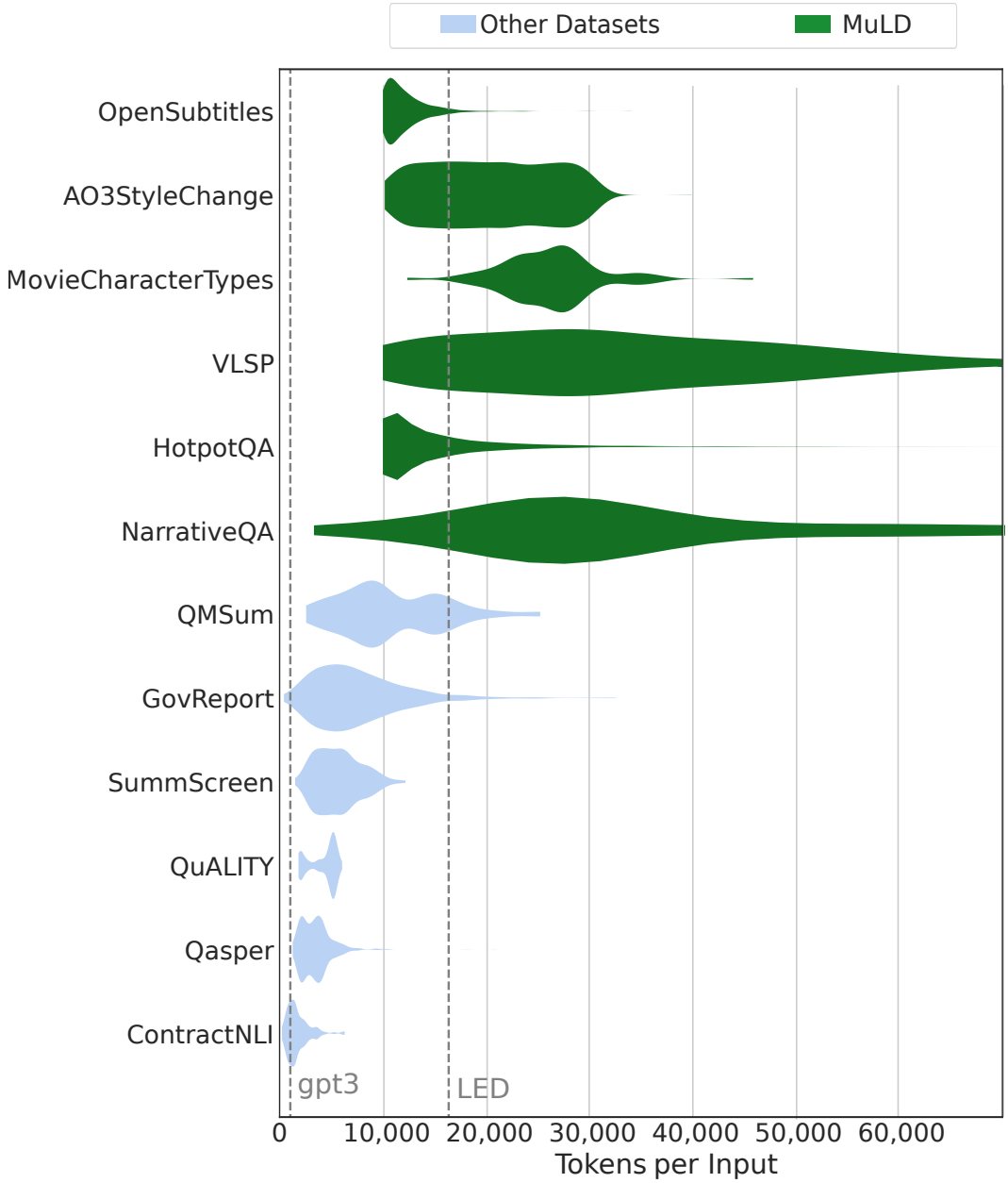



Figure 4.1: Comparison of lengths of multitask long document benchmarks using a BERT-style tokenizer (splitting on spaces and punctuation). The maximum input length of gpt3 and Longformer (LED) are included for comparison.

improving the efficiency of transformers, allowing them to operate on much longer sequences. However, these have mainly been evaluated on artificial datasets, with limited evaluation on real-world data.

There has been some work on creating long-text datasets. A notable example is NarrativeQA (Kočiský et al., 2018) which challenges models to answer questions about the plot of entire novels and movie scripts which have an average length of around 60,000 tokens - well beyond the input size of current typical efficient transformer models. There has however been little work in developing benchmarks of this length across a wide variety of NLP tasks.

Many 'long document' benchmarks use datasets consisting of at most a few thousand tokens (Figure 4.1). Notably, the Long-Range Arena (Tay et al., 2021) uses a maximum length of 8K tokens, and the QuALITY benchmark (Pang et al., 2021) uses a maximum of 6K tokens. I argue that these lengths are more akin to short essays, and the common usage of the term 'long document' would imply documents in the tens of thousands of tokens - similar in length to novels.

It is to this end that I present MuLD: The **Mu**ltitask **L**ong **D**ocument benchmark. This is a set of six long document datasets where each input is at least 10,000 tokens, spanning a range of dataset sizes, genres, and formulations designed specifically to test the ability of different approaches to model long-term dependencies in real world text. The datasets are formed by filtering, extending, or modifying existing NLP datasets.

I create baseline results for both standard approaches (T5) and efficient transformer methods (Longformer), finding that Longformer's extended input context allows it to perform better.

The MuLD dataset is available at `www.github.com/ghomasHudson/muld`.

## 4.2 The MuLD Benchmark

The MuLD benchmark is based on six long document NLP tasks, which span a wide range of domains.

### 4.2.1 Desiderata

I pick these datasets based on the following principles:

**(1) Long input size:** While many benchmarks use 'long-text' to mean inputs of a few thousand tokens, I consider 'true' long documents to be in the tens of thousands of tokens long. This more closely matches the common usage of this term, where an essay may be a few thousand words long and considered fairly 'short', while common everyday examples of long documents such as novels and reports may be 50,000-100,000 words in length. It is for this reason that I only include documents over 10,000 tokens in our benchmark, with many documents exceeding 100,000 tokens.

**(2) Variety of dependency on the input:** Within long document tasks, some may require understanding more of the input than others - either analysing the whole text or just using relevant sections. For example, in summarization a model must have a holistic overview of the document, while in other tasks such as question answering, the answer can be often be given by referencing just a few sections of the document. In reality, most tasks fall somewhere between these two extremes, and I endeavour to capture a variety of input dependency in my benchmark.

**(3) Variety of output length:** While the input length should be long, there is a range of different possible output lengths ranging from a single word classification label, a short answer, all the way up to an output of equivalent length to the input.

**(4) Existing Task Formulation** I don't seek to invent new types of task, but instead use proven tasks which are already agreed as being challenging for regular 'short' transformer models. Instead, the datasets used are created by either filtering existing datasets, expanding the length of existing datasets with additional text, or replicating the methodology of existing datasets with longer source text.

**(5) Easily Evaluated** I pick tasks where the performance can be easily measured with multiple automatic metrics. I also acknowledge that for some tasks, the current metrics used on short documents don't fully capture the challenges that long document evaluation poses.

### 4.2.2 The Datasets

The six long document datasets are described below and summarized in Table 4.1.

| Dataset | Task | Metrics | # Documents | avg. # Tokens |
|---|---|---|---|---|
| **NarrativeQA** | Question Answering | bleu, rouge, meteor | | |
| Train | | | 32,159 | 91,938 |
| Valid | | | 3,461 | 90,832 |
| Test | | | 10,261 | 88,579 |
| **HotpotQA** | Question Answering | bleu, rouge, meteor | | |
| Train | | | 87,752 | 23,775 |
| Valid | | | 7,405 | 22,669 |
| **AO3 Style Change detection** | Style change detection | F1-score | | |
| Train | | | 6,354 | 29,657 |
| Valid | | | 705 | 29,304 |
| Test | | | 2,352 | 30,502 |
| **Movie Character Types** | Classification | F1-score | | |
| Train | | | 167 | 44,640 |
| Test | | | 86 | 48,165 |
| **Very Long Scientific Papers** | Summarization | bleu, rouge, meteor | | |
| Test | | | 482 | 57,473 |
| **Open Subtitles** | Translation | bleu, rouge, meteor | | |
| Train | | | 4,252 | 12,330 |
| Test | | | 1,385 | 18,932 |

Table 4.1: MuLD data statistics

**NarrativeQA** The NarrativeQA Reading Comprehension Challenge Dataset (Kočiský et al., 2018) consists of user-submitted questions regarding the plot of movies or novels. Annotators were only given access to a human-written plot summary to encourage general questions which require a full understanding of the narrative. When these are given to a question-answering system along with the full text of the narrative (either a movie script or the novel text), this is a test of reading comprehension. As each sentence the annotator read can summarise the plot of an entire chapter/scene of a book or movie, models evaluated on the full text must model the dependencies between multiple sections of the narrative. The majority of these documents are longer than the 10,000 token minimum - any documents shorter than this are simply filtered out.

**HotpotQA** The HotpotQA dataset consists of questions from crowd workers which require information from multiple Wikipedia articles in order to answer, thus testing the ability for models to perform multi-hop question answering. The data is commonly presented as a list of paragraphs containing relevant information plus a setting where the addition of 'distractor paragraphs' fully test the ability of the model to comprehend which information is relevant to the question asked. To transform this into a long document, each paragraph is expanded with its full Wikipedia page as well as adding additional distractor articles from similar topics (randomly chosen from links on the existing pages) in order to meet the 10,000 token minimum length requirement for this benchmark. These articles are shuffled and concatenated to form the model input.

**Character Archetype Classification** I introduce a character archetype classification dataset based on the methodology of Skowron et al. (2016). For this dataset, each example consists of a movie script along with a named character, and the task is to classify whether the character is a Hero/Protagonist or Villain/Antagonist based on understanding their role in the narrative.

To gather this data, scripts are first picked from the web following Kočiský et al. (2018)[1], these are then matched with summaries of the plot from Wikipedia using a combination of matching names and titles with additional manual verification.

I extract character name candidates using a number of methods. Firstly, many of the scripts use the common format where the character name is given before each line of dialogue (e.g. 'HARRY: Where have they gone?'). Secondly, the Wikipedia pages for many films include a list of characters that can be parsed and matched with the script. These character name candidates are then filtered, only retaining those that also appear in the plot summary, eliminating false matches as well as some minor characters who don't impact the plot.

Annotators on Amazon Turk were given a description of the task and the character types of Skowron et al. (2016). They were then provided with the plot summary and asked to select the character type for each character name candidate extracted previously. Multiple annotators were used on each example to ensure the accuracy of the labels. From this process I eliminated the character types Mentor, Sidekick, and Spouse as there was a lot of disagreement between these classes, perhaps due to our use of movies of all genres rather than the limited set of action movies used by Skowron et al. (2016).

**Open Subtitles** The Open Subtitles corpus (Lison, Tiedemann and Kouylekov, 2018)[2] consists of aligned subtitles from movies and TV shows from the website `opensubtitles.org` in 60 languages and can be used for machine translation. Importantly, rather than individual lines, the data consists of the subtitles for an entire individual movie or TV show, many of these being very long files and I filter to remove any document with less than 10,000 tokens.

[1] Primarily from the Internet movie script database: `www.imsdb.com`
[2] `opus.nlpl.eu/OpenSubtitles2016.php`

One of the most common mistakes made by models which don't consider document-level context is the mistranslation of pronouns. For example, in the English phrase "It is cold.", the pronoun "it" could be translated differently depending on if the preceding context was "The ice formed at night" or "The camera was left outside". For this reason, I make use of the English-German pairs from the ContraPro annotation of open subtitles (Müller et al., 2018), which explicitly tests a model's ability to disambiguate these possibilities. This will allow future use of this contrastive evaluation, where models are asked to pick between the two possible translations and must show knowledge of the surrounding context.

**AO3 style change detection** Style change detection is the task of identifying the points where the author changes in a document constructed from the work of multiple authors. The PAN 2021 Style Change detection shared task (Zangerle et al., 2021) introduced this task, forming documents from StackExchange comments to produce a challenging dataset with multiple increasingly difficult subtasks. Task 1 is a binary classification task to identify whether the document contains multiple authors or just a single author. Task 2 is to identify the points where the style changes, and Task 3 is to assign each paragraph uniquely to an author out of the number of authors in the document. As Task 3 is the most challenging (and answers for the other two can be constructed from the output of task 3), I only report scores for it in our benchmark.

To extend this approach for a long document dataset, I instead use stories contributed to the fanfiction website *Archive of Our Own*, which contains numerous different works submitted by fans of popular films, TV, game, and book characters.

I use stories written about four of the most popular character relationships: Sherlock Holmes & John Watson, Castiel & Dean Winchester, Steve Rogers & Tony Stark, and Draco Malfoy & Harry Potter. Since the aim is for this task to be a test of style change detection and not topic change detection, each constructed document only contains paragraphs taken from the

same relationship to ensure that the difference between sections is the author style not the topic. Additionally, I reserve the "Draco Malfoy & Harry Potter" documents for the test set. After downloading, the data is cleaned by removing any images and special formatting characters, then split the stories into paragraphs, removing any with less than 100 characters.

To construct the style change detection documents, I first randomly choose a minimum document length (10,000-30,000) and a number of authors (1-4) with 50% of documents having a single author. I then assign authors to the document, randomly partition the lengths for each section, and finally draw, shuffle, and concatenate paragraphs to form the complete text.

**Very Long Scientific Papers (VLSP)** I follow the process of the Scientific papers (Cohan et al., 2018) summarization dataset, extracting papers from the open-access preprint server Arxiv.org using both the arxiv short abstract and the one included in the work (where available) as the reference summaries. In contrast to Cohan et al. (2018), rather than removing very long documents, I explicitly include them - removing any document with less than 10,000 tokens. With this new filtering, the dataset mostly consists of theses which, like the scientific papers dataset, have a regular structure consisting of multiple chapters. Due to the long time required to compile these large documents into text files, I only provide a small test set of 482 documents, so I train our models on the original smaller-length scientific papers dataset.

A box-plot of lengths for all six tasks is shown in Figure 4.2. NarrativeQA has the longest documents, but all tasks have a median length between 10,000 and 100,000 tokens.

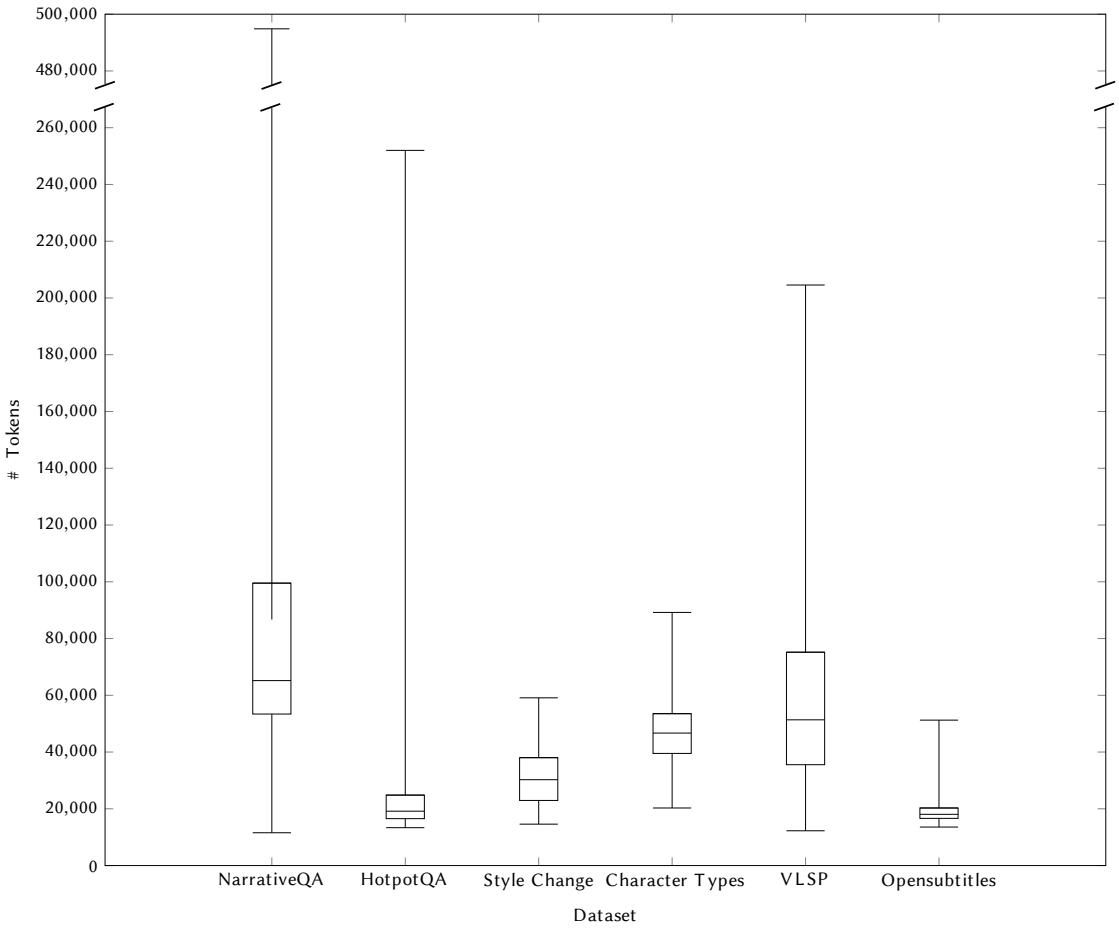



Figure 4.2: Dataset lengths (#tokens)

## 4.3 Baselines

In this section, I describe the models evaluated on the MuLD benchmark. I experiment firstly with a model based on a 'standard' transformer architecture: T5, an encoder-decoder network which was pretrained on multiple text-to-text tasks and can take a maximum input length of 512 tokens (Raffel et al., 2020). I compare this to the Longformer model: an 'efficient transformer' which uses sliding window attention (with global attention on some predetermined tokens) to

supports up to 4096 tokens (or 16,384 in the encoder-decoder variant) allowing us to see the benefits of using longer contexts for the benchmark(Beltagy, Peters and Cohan, 2020).

As it is not possible to directly feed the entirety of many long documents into models on reasonable hardware (even with longformer), I use chunking techniques to divide, solve, and recombine parts of the input Where not specified, the text is divided into fixed-size chunks matching the maximum input length of the model used. I use the following methods:

- **NarrativeQA/HotpotQA** - I follow the approach of Kočiský et al. (2018) by first dividing the document into chunks of 200 tokens. I then calculate the cosine similarity between the TF-IDF representations of each chunk and the question text in order to score every chunk. Using this metric, I pick the top 10 chunks and concatenate them together to pass into the model along with the question input.
- **Open Subtitles** - For a simple baseline, I split the document into regular chunks while respecting line breaks (a chunk is never ended in the middle of a line). Each chunk is then passed into the model to be translated, and all the translated chunks are concatenated together to form the translated document.
- **Style Change Detection** - I use the methodology of Zhang, Han et al. (2021), by training a classifier on paragraph pairs with the target of predicting whether the two paragraphs are by the same author. For the subtask I report (task 3), I additionally use the methodology of Strøm (2021): The first paragraph is assigned to the first author, then the model is used to determine if the second paragraph is similar to any of the previous paragraphs. If so, it is assigned to the author of the most similar paragraph. If not, a new author is added. Although I only report the score for task 3, this classifier could also be used to solve all the remaining two subtasks: for task 1, pass consecutive paragraph pairs into the model and output "single-author" if all paragraphs pairs are by the same author, otherwise classify

the document as "multi-author". For task 2, simply append all the model outputs together to form a list of style changes.

- **Character Archetype detection** - I select chunks containing the first mention, last mention, and most frequent mention of the character in concern. These are concatenated, passed to the model, and used to predict the character type class.
- **VLSP** - I split the document into chunks based on the article section headings and summarize the text from the first introduction section of the thesis onwards (ignoring contents, list of tables, list of figures sections).

Following Kočiský et al. (2018), I evaluate tasks which require text-generation (QA, Summarization, Translation) using Bleu-1, Bleu-4 (Papineni et al., 2002), Meteor (Denkowski and Lavie, 2011), and Rouge-L (Lin, 2004), using multiple references where these are available. For the style change detection and character type classification tasks, I simply report the F1-score.

## 4.4 Results and Discussion

The results on each of the benchmark datasets are presented in Table 4.2 for both the T5 and Longformer models.

The Longformer model consistently outperforms the T5 model across many of the datasets, suggesting that models which are able to make use of a longer context perform well on my benchmark.

Both models find the NarrativeQA dataset more challenging than HotpotQA, which I hypothesise is due to its longer average length, and the higher complexity of narrative understanding involved in NarrativeQA in contrast to the factual Wikipedia data of HotpotQA which typically involves

| Task | Bleu-1 | Bleu-4 | RougeL | Meteor | F1 |
|---|---|---|---|---|---|
| | | **T5** | | | |
| NarrativeQA | 17.67 | 0.55 | 19.03 | 3.36 | |
| HotpotQA | 28.11 | 13.63 | 27.61 | 4.46 | |
| Style Change | | | | | 26.49 |
| Character Type | | | | | 54.01 |
| VLSP | 28.85 | 0.84 | 16.55 | 7.98 | |
| OpenSubtitles | 34.07 | 1.63 | 35.35 | 38.53 | |
| | | **Longformer** | | | |
| NarrativeQA | 19.84 | 0.62 | 22.09 | 4.52 | |
| HotpotQA | 30.38 | 16.76 | 30.49 | 4.98 | |
| Style Change | | | | | 28.17 |
| Character Type | | | | | 82.58 |
| VLSP | 46.74 | 3.05 | 19.52 | 9.58 | |
| OpenSubtitles | 22.74 | 0.20 | 22.17 | 22.95 | |

Table 4.2: T5 and Longformer results on the benchmark

only a limited number of hops. Additionally, each HotpotQA question commonly involves understanding only a small number of sentences (even though these may be widely distributed throughout the document).

The T5 model also outperforms Longformer on the OpenSubtitles translation dataset. I suggest that this is due to the challenge of having to output a much longer sequence (512 vs 4096) tokens being greater than the benefit gained on the few lines where the correct translation depends on context more than 512 tokens away (Müller et al. (2018) find that in around half of cases, the antecedent is in the previous sentence, and very few are more than 3 sentences away).

## 4.5 Conclusion

To enable the evaluation of long document models, I introduce MuLD: a benchmark of varied NLP tasks where each document consists of more than 10,000 tokens. The six datasets in my benchmark are created by filtering, extending, or modifying existing NLP datasets and are designed to require a long context for high performance.

I evaluate simple chunking-based baselines, and find that the Longformer model is able to outperform the T5 model suggesting our benchmark is a good test for the ability of models to make use of longer contexts.

I believe that the technique explored in this work, of augmenting and extending existing 'short document' datasets, can be applied to many other NLP tasks. As the performance of efficient transformers improves, I anticipate the need to update this benchmark with more challenging tasks. While I am focused on creating a benchmark which tests a model's ability to solve real-world long document tasks, I also expect improvements in the efficiencies of the models themselves which may make datasets with more than 100,000 tokens necessary which may require a fundamentally different approach to creating long document datasets. I leave for future work both the development of improved chunking methods, and more efficient transformers which make such methods unnecessary.

I hope that the MuLD benchmark will encourage this further research into efficient models for long document NLP. To this end I provide the data, baseline models, and other code at `www.github.com/ghomasHudson/muld`.

## Epilogue

This chapter introduced a new multitask long document NLP benchmark based on extending existing NLP datasets, as well as simple baseline models which involve devising separate strategies for each task.

However, task-specific logic in the baseline model limits the ability to use multitask learning via task-reformulation. In the next chapter, I create a model based on task-decomposition for solving long document NLP which requires no task-specific logic within the model, enabling multitask learning and the extension to new tasks via only editing input data.

# Chapter 5

# A Multitask Decomposition Approach to Long Document NLP Problems

In this chapter, I explore how a task decomposition model can be used to solve long document NLP tasks.

One of the major limitations of current NLP techniques is the length of the text that they are able to process. While models such as BERT (512 input tokens) have shown great performance on tasks involving single sentences, sentence pairs, or a few paragraphs (Devlin et al., 2019) - there are many real-world tasks which require processing much longer sequences. Problems such as summarizing lengthy reports and translating entire novels require this ability to efficiently process thousands of words.

There have been a variety of attempts in the machine learning field to use task-decomposition as a framework for breaking down complex tasks (Singh, 1992; Dayan and Hinton, 1992). This decomposition may take many forms, and the choice of decomposition approach is often highly task-specific.

In this work, I explore a task-decomposition approach for long document NLP problems, which allows inputs of any length to be processed. By decomposing a large task into several smaller subtasks, we can solve a wide variety of common NLP tasks, without any task-specific logic outside the model. This allows any task to be included simply by finetuning the model, as well as allowing for multitask training. This produces a highly interpretable model where the intermediate outputs can be inspected to provide insight into the model (more details in section 5.2).

My contributions are:

- A task-decomposition approach which has no task-specific modelling.
- Evaluation on a set of long document datasets, showing that my approach outperforms the baseline model on 2 of the tasks.
- Analysis of intermediate outputs.

## 5.1 Datasets

Following the outline of the MuLD benchmark introduced in Chapter 4, I evaluate my approach on five NLP tasks. As my approach involves decomposing the tasks into smaller subtasks, the burden for task-specific logic is moved from the model to the dataset format. This requires datasets with supervision for each of these subtasks, requiring some modifications to the datasets used by MuLD:

**BookSum/Goodreads2020** - As the Very Long Scientific Papers (VLSP) dataset introduced in Chapter 4 does not provide supervision for the intermediate subtasks required by my approach, I use a book summarization dataset used by Wu, Ouyang et al. (2021) facilitating direct comparison with their similar task-decomposition approach for summarization [1]. I use BookSum (Kryściński et al., 2021) for training, which consists of public domain works from the Project Gutenberg repository paired with human-written summaries at the paragraph, chapter, and book level. This allows me to provide supervised training examples at all stages of our task-decomposition approach. I then compare against a test set of human summaries of the 40 most popular books in 2020 on the GoodReads website as collected by Wu, Ouyang et al. (2021).

**HotpotQA** - The HotpotQA dataset (Yang, Qi et al., 2018) is designed to evaluate models' ability to answer questions that require information from multiple Wikipedia articles. I use the version introduced in Chapter 4, which transforms this into a long document task by providing full Wikipedia articles appended together as the input instead of individual paragraphs.

**Style Change Detection** - A dataset which challenges models to identify which paragraphs are written by the same author in a document formed by combining the work of multiple authors on the same topic. The dataset improves on the task formulation outlined in the work of Zhang, Han et al. (2021) using longer documents and is described in chapter 4.

**Character Type Identification** - As described in chapter 4, this is a dataset of movie scripts where the objective is to identify whether a given character is the protagonist or antagonist.

[1] As BookSum was added after the creation of MuLD, it was not included in my benchmark.

**OpenSubtitles** - A translation dataset based on movie subtitles in English and German. We use the subset of the full subtitle files which are over 10,000 tokens long as described in chapter 4. We make use of the English-German pairs used by the ContraPro annotation of OpenSubtitles (Müller et al., 2018) which includes information about which parts of the context surrounding a subtitle line are needed to unambiguously translate it.

These tasks are chosen as they span a wide variety of NLP tasks, and allow for the creation of supervised data for the subtasks which are needed by our method.

## 5.2 Method

I propose a task-independent decomposition architecture, designed to solve NLP tasks where the major complexity of each task derives from their length.

The input for each task is formed into a *</key/> value* pair format which is specific for each task type and designed to ensure the final output is in the correct format and context is passed between chunks when needed. In all tasks, one of these *</key/> value* pairs is a long document consisting of at least 10,000 words (this approach could be extended to tasks which compare multiple long documents). The tasks are decomposed by chunking the text of this long *</key/> value* pair into smaller subtasks, while copying the other *</key/> value* pairs across to each one. The subtasks are then solved using the same language model for each, which produces an output in a similar *<key> value* format. The output texts produced by these models are then combined with the model input text using the following method:

- Any keys contained in the input text but not the output text are retained.
- Any key which appears both in the input text and the output text is overwritten by the new value.
- Any new output keys not appearing in the input are sent to a global output immediately and not retained.

These intermediate outputs are then combined and re-chunked into the next layer of the hierarchy – getting smaller each time, until the inputs fit within the input size of the model, which is then added to the global output.

Granting the ability for models to add to the global output both at intermediate stages and at the final stage allow my approach to solve both tasks which wait till the entire document is seen to produce an output (e.g. document classification), and other tasks such a translation which produce a very long output, many parts of which can be solved at intermediate stages.

As illustrated by Figure 5.1, the processing of a document using my approach can be thought of as a tree. I use the standard tree terminology where the depth of a subtask is its distance (number of layers) from the final root node, and the height is the distance of the longest path to a leaf subtask. The model often trained to perform different operations depending on whether the current subtask is a leaf, internal, or root node. This is inferred by the structure of the data passed as input for the subtask.

Figure 5.2 shows a worked example of this process for the question answering task.

One of the key benefits of this task-decomposition architecture is its scalability. It can be applied to any document length and task type, making it a versatile and adaptable solution for a wide range of NLP problems. However, by moving any task-specific logic from the model into the dataset format, this requires labelled data for each of the intermediate steps, which can either be generated automatically or in some cases, human-labelled. Multitask learning may reduce the number of labelled examples needed when adding related tasks.

Additionally, the hierarchical nature of the architecture provides interpretability through the inspection of intermediate outputs. This makes it possible to understand how the model arrived at its final answer, which can be useful for debugging and improving the system, as well as understanding failure cases.

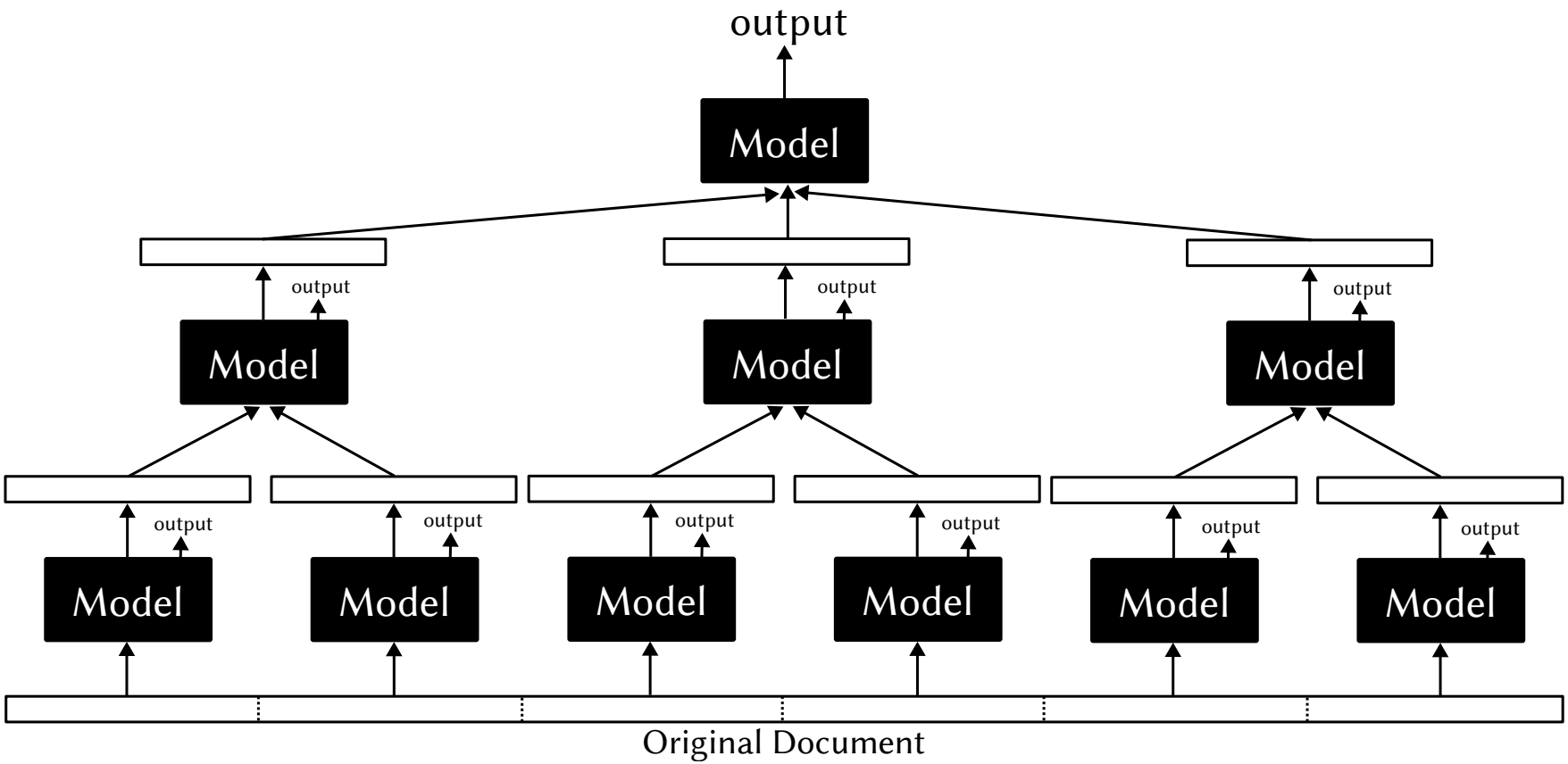
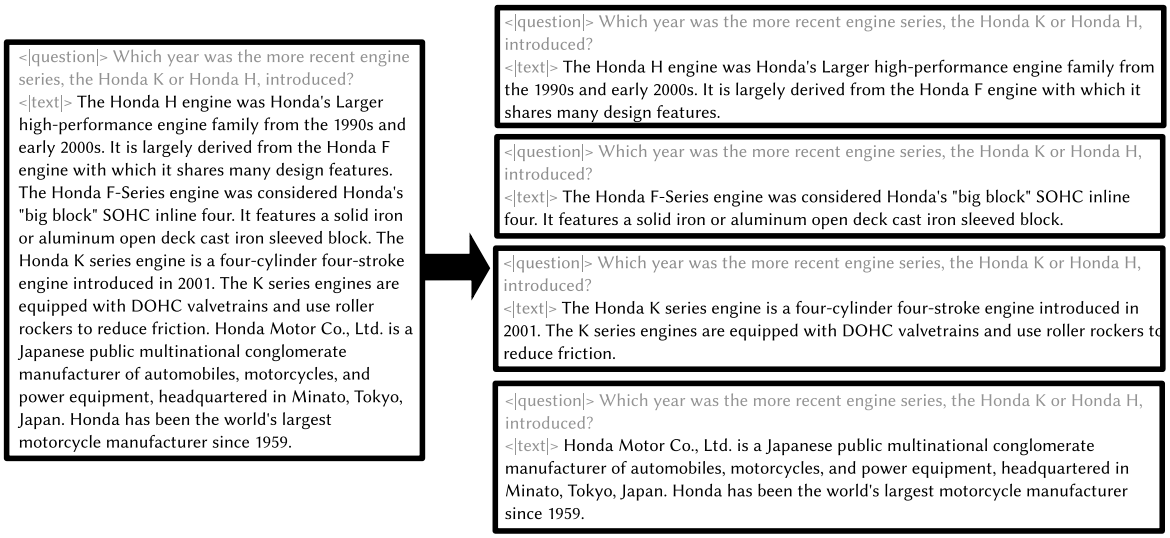
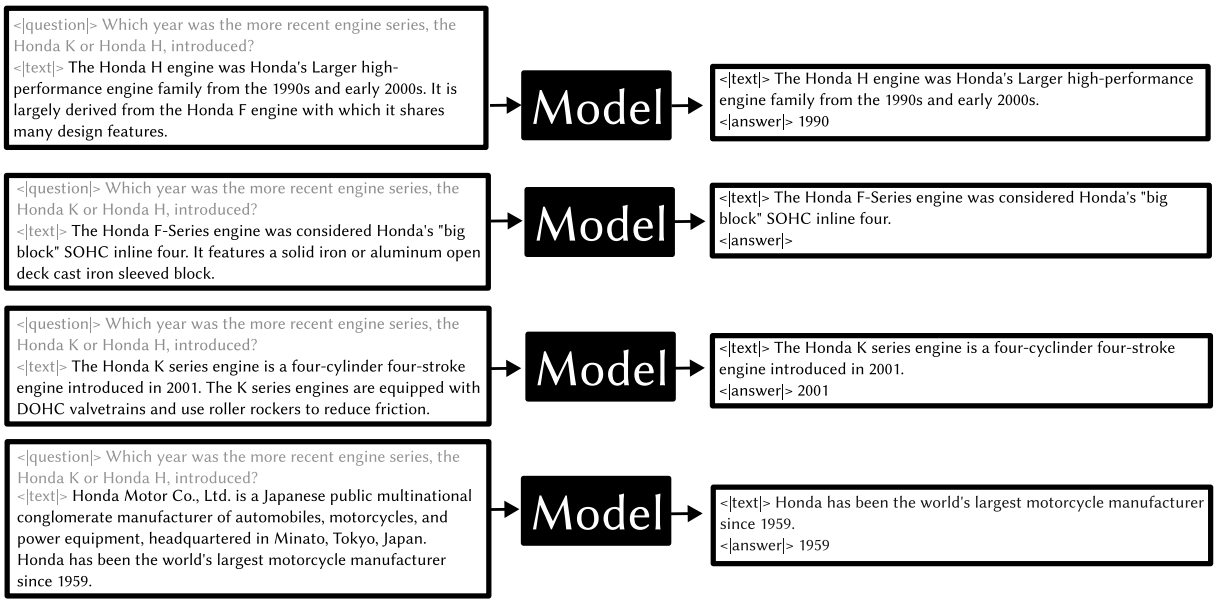


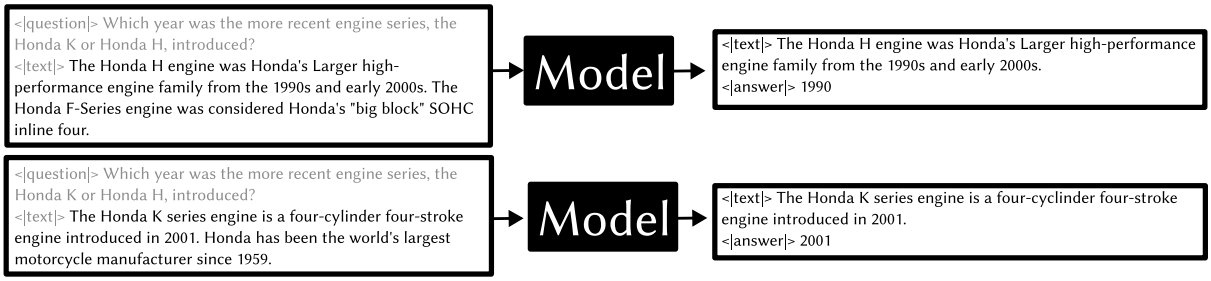

Figure 5.1: Illustration of my task-decomposition method. The input document is first split into chunks, then each chunk is passed to a model which either outputs immediately or creates an intermediate output to be passed to the next layer (or both). This is recursively repeated until the entire intermediate output fits within the chunk size.




(a) The document is first broken down into chunks using a fixed-size chunking algorithm. Each chunk is formatted in the same <|key|> value format as the original input.




(b) Each chunk is passed into the base model which, in the case of QA, extracts the most relevant sentences. This is different for every task.




(c) These chunks are merged, reformatted and passed into the same model for the next layer.


(d) This process is repeated until the input is less than the chunk size and can be processed by the model in a single pass. This produces a final answer which is output.

Figure 5.2: Example of the process for question answering

The following sections describe how I frame each of the tasks to work within this architecture. It should be possible to create similar formulations for any NLP task.

### 5.2.1 Summarization

**Input**

```
<|text|> Johnnie, a determined 19 year old girl, works at the cotton
   mill to support her family. She plans to pay back all the debts
   they have accumulated. [...]
```

**Output**

```
<|text|> Johnnie works at the cotton mill to support her family. She'
   s attracted to her boss, Gray Stoddard, a passionate socialist who
    believes marriage should be based on the potential of the
   children. [...]
```

Figure 5.3: Example of the format used for summarization. Each chunk is formed into the input format and the model is trained to output the summarized text beneath. This is then passed as the new <|text|> input for the next layer up.

Summarization is a task that is particularly well-suited to my decomposition-based approach, as each intermediate step is simply summarization at a different scale. The final summarization can be constructed based on the summarizations of its parts - we only need to learn one task. The model is simply trained to summarize chunks of the input document at every stage. The output of each chunk is then merged with its neighbours and summarized again till the whole document can be summarized in one step.

As this book summarization was the subject of previous work by Wu, Ouyang et al. (2021), I replicate their approach using the same publicly-released dataset[2], which provides supervised pairs at every stage of the decomposition.

[2] https://github.com/salesforce/booksum

### 5.2.2 Translation

Input

```
<|text|> However, the European Central Bank (ECB) took an interest in
    it. Bitcoin has been described as ‘the most successful virtual
   currency’.
<|context|>
<|subject|>
```

Output

```
<|text|> However, the European Central Bank (ECB) took an interest in
    it.
<|output|> Bitcoin wurde als "die erfolgreichste virtuelle Wahrung"
   beschrieben.
<|context|> European Central Bank (ECB), ORG, it |
<|subject|> Bitcoin
```

Figure 5.4: Example of the format used for translation. Each chunk is formed into the input format and the model is trained to output the translated text beneath. In this example, the second sentence can be immediately translated, so is output immediately. The first sentence relies on previous context, so the translation is delayed until the context from the previous chunk can be integrated at the next layer.

Because the output for translation can be of a similar length (or even longer) than the input, it is not possible to output the translation in the final, root node due to the requirement that the document must get shorter at every stage (guaranteeing the model will finish executing).

However, I observe that in the majority of cases, sentences can be translated correctly without referring to any context outside the current chunk. To exploit this, I make use of the model's ability to choose between outputting immediately or at the final level.

The model is trained to immediately output those sentences which need no additional context outside the current chunk. Any sentences requiring more context are passed up to the next levels until enough context is present within the current chunk. The labels for this come from

the ContraPro annotations (Müller et al., 2018), which specify the nearest context needed for a correct translation.

In order to pass context between chunks, we additionally extract the following information at the tree leaves:

- Named Entity and pronoun pairs via coreference resolution
- The subject of the final sentence of the chunk using part of speech (POS) tagging

The named entity extraction and coreference resolution are performed using the Spacy package (Honnibal et al., 2020) and POS tagging is performed by the python nltk package (Bird and Loper, 2004).

The internal nodes combine this information by merging entity-pronoun pairs and only keeping the final subject of the now larger chunk. This minimizes the size of the chunks passed to the next level while preserving context needed for the translation.

Figure 5.4 demonstrates how sentences in a single chunk can either be outputted immediately (in the case of the second sentence) or passed to the next layer (in the case of the first sentence) to wait for additional context.

### 5.2.3 Classification

For the character type classification task, the goal is to predict the type (Hero or Villain) of character depicted in a document, based on the text in the document. The output of the task is a single prediction for the entire document.

**Input**

```
<|character|> Chad
<|text|> Chad leans forward, making sure his words resonate with the
   student body. CHAD And as for the perpetrators of this heinous act
   , even if you are still a minor, you will be tried as an adult,
   and, if convicted, you will do hard time. He pauses to let the
   impact of this wash over the students. CHAD Are there any
   questions? ELENA (calling out) Yeah, I got [...]
```

**Output**

```
<|text|> Hero |
```

(a) Lowest level format. The model outputs a single classification label given a chunk of the text

**Input**

```
<|character|> Chad
<|text|> Hero | Hero | Antagonist | Hero | [...]
```

**Output**

```
<|text|> Hero |
```

(b) Higher level format. The model performs a majority vote on the outputs of previous levels.

Figure 5.5: Example of the format used for character-type identification detection (classification)

The lowest level of the hierarchy is responsible for assigning Hero/Villain labels to individual chunks of text (as shown in Figure 5.5a). The higher levels of the hierarchy are then used to combine these chunk-level labels into a single document-level label. This is achieved by using a majority vote mechanism, where the most frequently predicted label across all the chunks is selected as the final prediction for the document (as shown in Figure 5.5b).

To create the training data, first all chunks of each document which mentions the character in question is taken. This is formed into the input format described in Figure 5.5. Chunks which do not mention the character are given a blank output.

For the internal nodes, randomly generated hero/antagonist lists of varying lengths are used along with the correct majority vote label.

### 5.2.4 Style Change Detection

In style-change detection, the goal is to label every input paragraph with the corresponding author. This task is split into two stages of processing:

- At the leaves, the model is trained to assign authors to paragraphs within individual chunks of the input text. When a change of author occurs within a chunk, the label assigned by the model is changed accordingly.
- At other nodes, the model reconsiders the labels assigned at the lower levels based on the context of other neighbouring chunks. This step helps to ensure that the authors are consistently labelled across the entire document. If it is found that paragraphs that were previously assigned different labels are now written by the same author, the labels are reassigned accordingly.

Figure 5.6 provides a visual representation of how this process is carried out. At the lowest levels, the chunks are partitioned and a sample of each author is passed to the next level. The next level then reassigns the labels to ensure that the author labels are consistent within the larger chunk. This two-stage process gives us document-level author labels.

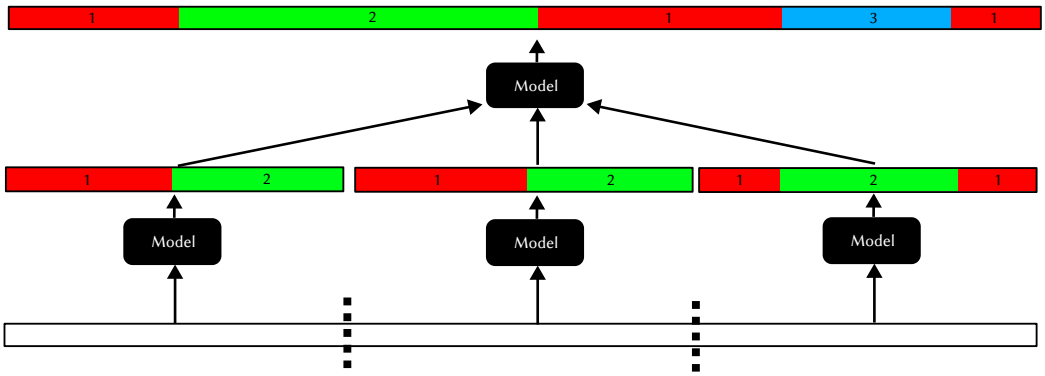



Figure 5.6: Representation of how style change detection is performed. The lowest level assigns the initial labels to each chunk. The next layer takes these outputs (and samples from each section) and recontextualises them based on each other. In this example we see how the model identifies that the first author of chunk 2 is actually the same as the second author of chunk 1, the second author of chunk 2 is the first author of chunk 1, while the second author of chunk 3 is not seen elsewhere so is given its own label.

### 5.2.5 Question Answering

For question answering tasks, the model is presented with a question and a long document that is expected to contain the answer. At each stage of processing, the model is trained to generate both the answer to the question and any relevant facts extracted from the input document that support this answer. These supporting facts are then passed on to be used as the input for the next level up, where the model continues to refine the information and extract only the most pertinent parts of the document.

This continues until the final layer, which outputs the answer based on the most relevant information gathered from the entire document, which has been distilled using the most important

Input

```
<|question|> What government position was held by the woman who
   portrayed Corliss Archer in the film Kiss and Tell?
<|text|> Kiss and Tell is a 1945 American comedy film starring then
   17-year-old Shirley Temple as Corliss Archer. In the film, two
   teenage girls cause their respective parents much concern when
   they start to become interested in boys. [...]
```

Output

```
<|text|> Kiss and Tell is a 1945 American comedy film starring then
   17-year-old Shirley Temple as Corliss Archer.
<|answer|>
```

Figure 5.7: Example of the format used for question answering. The question and a chunk of the input document are passed as input to the model. The model will then output any sentences from the text which are needed to answer the question, along with the expected answer. This is then passed as the new <|text|> input for the next layer.

pieces of information from the previous layers. In this sense, question answering can be thought of as extractive summarization guided by a question in the non-root nodes.

### 5.2.6 Training Details

I use the open-source GPT-J-6B (Wang, 2021) as a base model for my method. This model was pre-trained on the large-scale curated dataset the Pile (Biderman, Bicheno and Gao, 2022) and closely follows the model design of GPT-3 (Brown, Mann, Ryder et al., 2020b). I train the model on each of the tasks individually using the language model formats described above and the hyperparameter and architecture choices of Wu, Ouyang et al. (2021). The model takes 2048 tokens as input, so this is used as the chunk size for all tasks. I follow Wu, Ouyang et al. (2021) in using 0.6 as the temperature for model generation.

The model is trained using both finetuning and a RLHF (Reinforcement learning from human

feedback) style approach (Stiennon et al., 2020)[3]. This training process has 3 key stages:

1. The model is finetuned on a subset of the data.
2. A reward model is trained to imitate human preferences. The model is provided with two candidate responses $(y_k, y_j)$ and is tasked to optimise on the loss function,

$$\text{loss}(\theta) = -E_{(x,y_j,y_k)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_j\right) - r_\theta\left(x, y_k\right)\right)\right)\right]$$

   This has the effect of training the model to give a higher score to the preferred candidate, $y_j$ over the alternative $y_k$.
3. Reinforcement Learning - The model is now run in a reinforcement learning loop and the outputs compared to those from the untrained model at each stage by using the reward model.

In order to create the pairs $(y_k, y_i)$, different strategies are employed to generate an incorrect but plausible output, $y_k$ for each task:

- **Question Answering** - For QA, the <facts> section is constructed using a random sentence from the context, or the empty string. The answer is a random sequence of words from the context or "yes" or "no".
- **Summarization** - The context is copied over.
- **Style Change** - For Style Change detection, a random sequence of style changes is generated corresponding to the length of the input

[3]Despite the name, the RLHF approach used here has no human-in-the-loop.

- **Character Identification** - For the lowest level, the incorrect classification ("Hero" or "Antagonist") is generated or the empty string (to simulate when the chunk doesn't contain the character in question). For the higher majority-vote tasks, the incorrect response (e.g. "Hero" if $y_j$ is "Antagonist") is used.
- **Translation** - For examples with ContraPro annotations, the alternative (incorrect) translation is given for the <output>.

## 5.3 Baselines

I compare my model to the baseline introduced in 4. This uses a basic chunking method, which requires separate task-specific logic for every task:

- **Question Answering** - I follow the approach of Kočiský et al. (2018) by first dividing the document into chunks of 200 tokens. We then calculate the cosine similarity between the TF-IDF representations of each chunk and the question text in order to score every chunk. Using this metric, we pick the top 10 chunks and concatenate them together to pass into the model along with the question input.
- **Translation** - I split the document into regular chunks while respecting line breaks (we never end a chunk in the middle of a line). Each chunk is then passed into the model to be translated, and all the translated chunks are concatenated together to form the translated document.
- **Style Change Detection** - I use the methodology of Zhang, Han et al. (2021) and Strøm (2021), by training a classifier on paragraph pairs with the target of predicting whether the two paragraphs are by the same author. I assign the first paragraph to the first author, then check if the next paragraph is similar to the previous one. If so, I assign this to the author of the most similar paragraph if over some threshold. If not, I add a new author.

- **Character Archetype detection** - I select chunks containing the first mention, last mention, and most frequent mention of the character in concern. These are concatenated, passed to the model, and used to predict the character type class.

Additionally, I compare summarization against the similar decomposition approach of Wu, Ouyang et al. (2021).

## 5.4 Results

| Task | Bleu-1 | Bleu-4 | RougeL | Meteor | F1 |
|---|---|---|---|---|---|
| **Our Model** | | | | | |
| HotpotQA | 29.56 | 17.07 | 29.20 | 5.46 | |
| Style Change | | | | | 29.75 |
| Character Type | | | | | 66.00 |
| OpenSubtitles | 67.88 | 8.76 | 46.40 | 36.45 | |
| GoodReads2020 | 46.52 | 0.07 | 15.81 | 9.97 | |
| **Baseline** | | | | | |
| HotpotQA | 30.38 | 16.76 | 30.49 | 4.98 | |
| Style Change | | | | | 28.17 |
| Character Type | | | | | 82.58 |
| OpenSubtitles | 22.74 | 0.20 | 22.17 | 22.95 | |
| GoodReads2020 | | | 39.53 | | |

Table 5.1: Single task results using my task-decomposition approach. My model outperforms the baseline mode for the translation and style change detection tasks.

Following Kočiský et al. (2018), I evaluate tasks which require text-generation (QA, Summarization, Translation) using BLEU-1, BLEU-4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), and ROUGE-L (Lin, 2004), using multiple references where these are available. For the style change detection and character type classification tasks, I simply report the F1-score. The results of these experiments are shown in Table 5.1.

In general, the model is able to achieve similar performance to the baseline model while adopting a task-agnostic approach, and outperforms it on 2 of the tasks: Style Change Detection and Translation.

Additionally, I examine the performance of the model as compared to the length of the input document and find no strong correlation (r=0.05).

### 5.4.1 Summarization

My findings regarding the performance of our model in producing book summaries are similar to those reported by Wu, Ouyang et al. (2021). The model can generate good quality summaries, some of which meet human-level quality, as measured using a 7-point likert comparison (Figure 5.8) where human evaluators were asked to rate summaries based on:

- *Coverage* - whether the level of detail in the summary is appropriate.
- *Accuracy* - whether the summary correctly reflects the information in the original book.
- *Coherence* - whether the summary itself is comprehensible intrinsically, without reference to the original book.

From this we see that while human summaries still outperform our model, it still produced summaries on average, 13% and 4% of them reach 5 and 6 points respectively.

While the model is able to generate good summaries of books, the structure of these summaries often resembles a list of events that occur in the book rather than a cohesive and coherent description. This is particularly noticeable for books where the narrative is not as significant as the central themes and ideas that the book explores.

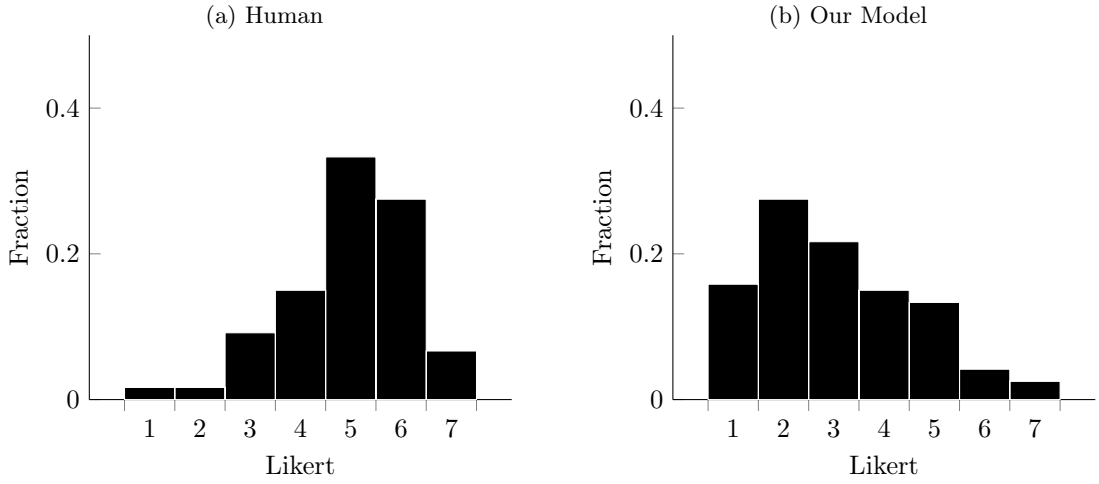



Figure 5.8: Likert scores for the summarization task comparing human summaries with ones produced by my model.

In comparison, human summarizers tend to capture the key themes of the book in their summaries, which shows a more comprehensive understanding of the book's content. This highlights a limitation of the approach, which struggles to capture the broader context and meaning of a book and instead focuses on recounting a series of events. This is an intrinsic limitation of an approach which constructs summaries hierarchically.

### 5.4.2 Question Answering

I find that for question answering, errors frequently occur at the lowest level of the processing tree. To quantify this, I track how supporting facts from the input data are extracted and passed up to the next level, finding that only 12% of facts are correctly extracted at this first stage.

This is the stage where the model is tasked with identifying relevant facts from the original text, and I have found that when the model gives an incorrect answer, many of these facts are often missed. This can result in the model not having all the information it needs to generate an accurate answer to the question.

I believe that this issue is due to the fact that wider context is sometimes needed in order to determine if a sentence is relevant to the question. For example, when a new technical term is defined in the document, it can be difficult for the model to determine its relevance to the question without understanding how it is used elsewhere in the document.

This limitation is a fundamental challenge of any approach that seeks to process a document using decomposition in a single pass.

### 5.4.3 Translation

For translation, I find that the model performs well, producing the correct translation in the majority of cases. When the context needed is far away from the required translation, the model fails to extract it. This is particularly true in cases where evaluating whether context is required depends on knowing which sentences require the context and is not captured by our simple heuristics. Perfect accuracy on this would again require a model which can process the document in multiple passes and can refer back to other sentences while translating the current one.

### 5.4.4 Character Type Classification

For character type classification, errors mostly occur in the lowest level of the hierarchy, where the model only picks the correct label 59% of the time based only on a single chunk. In contrast, when doing majority vote over the labels from multiple chunks, the model outputs the correct response 90% of the time.

### 5.4.5 Chunk analysis

To better understand how chunks are processed in the model, I compare the proportion of chunks at each layer (Figure 5.9). I use standard tree nomenclature - The layer height is the length of the longest downward path to a leaf from nodes at that layer.

In summarization, an exponential decrease in the number of chunks produced at each layer is observed, due to each chunk being trained to produce an output which is roughly the same proportion to the input length.

As QA can be thought of as summarization guided by a question in my approach, it follows a similar pattern albeit with a quicker rate of decay as only short fragments (1 sentence or less) are commonly relevant to the question in each chunk.

For translation, the majority of chunks can be translated in a single step without requiring context. This outcome aligns with expectations, since most sentences in the dataset do not rely on context from other sentences. However, for the few chunks that do require additional context, neighbouring chunks at the next layer can typically provide the necessary information for translation (In fact, in the contraPro annotations, only 3.7% of examples have context more than 3 sentences away - which means most end up within the same chunk).

Similarly, for character type identification, the model is able to solve most problems within 3 layers, as all apart from the first layers are simply majority vote.

For Style Change Detection, a long tail is seen as each section is recontextualized based on the surrounding chunks. This is always done until every chunk has seen the context of the whole document.

### 5.4.6 Multitask

As well as single task results, I also train the model on all tasks jointly in a multitask setting. This is done by prepending the prompts in Figure 5.10 to each example, and then training the model with randomly ordered chunks from each task. Because the only difference between tasks is their training data, the model can be easily trained for multitask learning without any further modification - a key advantage of our task-agnostic approach.

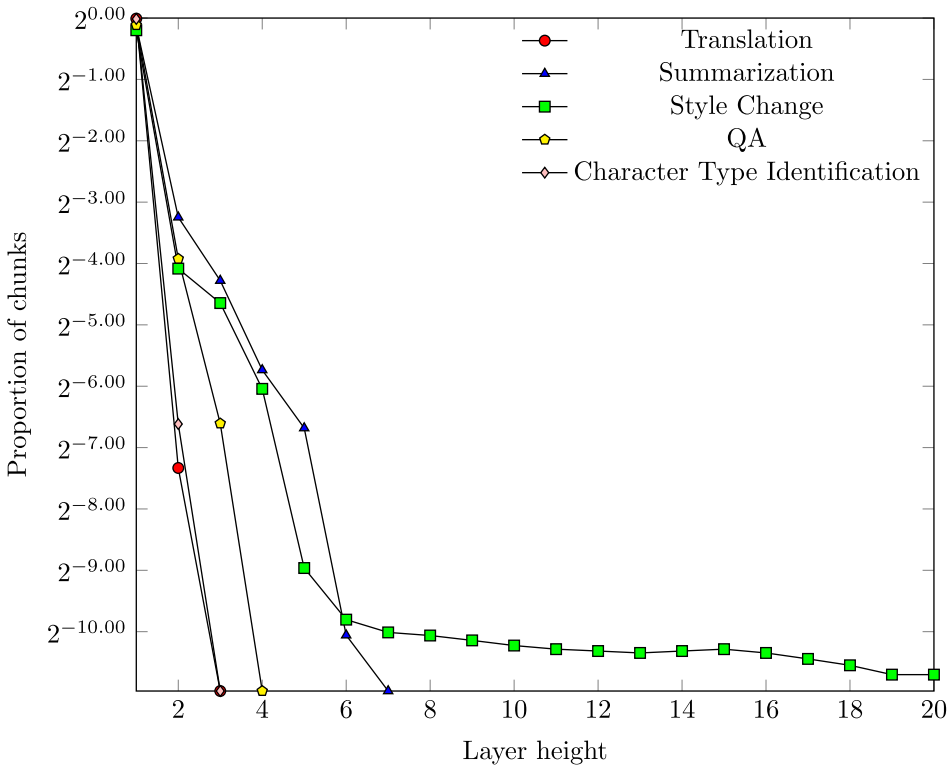



Figure 5.9: Proportion of chunks at each tree height across the different tasks.

Table 5.2 shows that while the multitask training does produce a performance penalty, this is minimal, the model is able to successfully distinguish between tasks.

For the two tasks (summarization and translation) which overlap with the PQ-decaNLP dataset introduced in Chapter 3, we repeat the evaluation of how robust our model is to paraphrased questions. We find that for both these tasks, the model only looses between 1-2% performance, suggesting it is much more robust than our previous approach. We attribute this mostly to the use of a base language model which has more parameters, and has been pretrained on a more varied training set.

What is the summary?
Translate from English to German
Is CHARACTER_NAME a Hero or Antagonist?
Identify the author for each paragraph

Figure 5.10: Task prompts used for multitask training. Question answering does not use a task prompt, as the task is implicit in the question. CHARACTER_NAME is replaced with the name of the character to be classified.

| Task | Bleu-1 | Bleu-4 | RougeL | Meteor | F1 |
|---|---|---|---|---|---|
| HotpotQA | 27.56 | 17.07 | 29.20 | 4.46 | |
| Style Change | | | | | 20.26 |
| Character Type | | | | | 43.75 |
| OpenSubtitles | 66.35 | 6.40 | 43.22 | 35.39 | |
| GoodReads2020 | | | | 10.37 | |

Table 5.2: Multitask results using our task-decomposition approach.

## 5.5 Conclusions

In this work, I demonstrate how task-decomposition can be used to solve multiple NLP problems where the input is very long. I achieve results competitive with the baseline, surpassing the baseline performance for 2 of the tasks. This performance is achieved using a model without any task-specific components, moving any task-specific logic from the model to the dataset preparation. This enables multitask learning via providing the model with a task prompt. Additionally, this approach allows investigation of intermediate outputs, giving insights into the decision-making process of the model.

While my approach proves successful in numerous scenarios, I recognize its limitations when confronted with tasks like multi-hop question answering. These tasks necessitate multiple reasoning steps and the capacity to extract information from different document sections, drawing on understanding derived from other parts. However, with my approach, the model processes the document in a single pass and is unable to jump arbitrarily between different parts of the document.

To overcome this fundamental limitation, a more flexible approach is necessary, empowering the model to choose which document sections to read and allowing it to revisit previously seen sections. By enabling the model to dynamically make decisions based on the content it has already processed, we can mimic human problem-solving strategies. Humans evaluate the context, determine the required information, and navigate the document accordingly.

Such a flexible and adaptable approach would not only enhance performance in multi-hop question answering, but also improve outcomes in a wide range of complex tasks. By reducing the assumptions made during data formation and instead replicating human approaches to these tasks, a more robust and capable model can be created. Future research should focus on developing methodologies that facilitate the model's decision-making process and enable it to navigate and comprehend long documents more effectively. This will bring us closer to achieving human-level performance in handling complex document-based tasks.

## Epilogue

In this chapter, I introduced a task-decomposition based model for solving long document NLP tasks. I show how, via prompts similar to those used in decaNLP, this can also be used for multitask learning via task reformulation.

The next chapter takes these two key ideas (task-decomposition and task-reformulation) and explores how they can apply to other modalities - namely computer vision. To do this in vision, we need a task-reformulation supertask suitable for this new modality. I dub our choice of this supertask - *Visual Modelling*; inspired by its NLP analogue, language modelling.

# Chapter 6

# The Visual Parallel to Language Modelling Evaluated on Dynamic Simulations

In order to explore how the ideas from the previous chapters (namely multitask learning via task-reformulation and long document models via task-decomposition) can apply to other modalities, this chapter introduces the concept of visual modelling - the visual parallel to language modelling.

Visual modelling is a first step towards a "supertask" for vision, much in the same way that language modelling and question answering can be used as supertasks for NLP under multitask learning via task reformulation.

This chapter has 2 main contributions. First, I show that models can be built to solve this task, and that the knowledge gained from pretraining on visual modelling can help downstream tasks.

I create a multitask benchmark of 6 vision tasks based on dynamic physics simulations. These datasets are designed to test the ability of models to capture physical laws in a variety of settings. I run multiple benchmark models on these to assess the ability of models inspired by NLP to tackle complex vision tasks, as well as probing what knowledge is induced while pretraining on visual modelling.

Second, I suggest ways that the concepts explored in previous chapters - task decomposition and task reformulation, can be used within this modality. I discuss how visual modelling allows any vision or NLP task to be reformulated in terms of this single "supertask". I illustrate this potential by training a model to predict the next frame of a video consisting of Amazon product review text and product rating to explore the ability of visual modelling to also solve NLP tasks. Additionally, I explore how the model from Chapter 5 could be modified to handle long video sequences in a task-agnostic way - similar to how it performs on NLP tasks.

This chapter stems from a joint project and my contributions involve the creation of the physics simulation datasets as well as the design and analysis of experiments using the baseline model. The work on how this approach could be used to unify modalities is also my contribution. Other contributions from Tom Winterbottom, and Daniel Kluvanec include code, design, and analysis of experiments, as well as further results not included in this chapter.

**Note to the reader:** While I argue showing that proving that the visual modelling paradigm works on video is a necessary precursor to using it was a multimodal "supertask", readers primarily interested in its use for uniting NLP and vision may want to skip to Section 6.7 where I discuss the ways it can be applied to NLP. I hope the reader will find these preliminary ideas an interesting direction for future work.

## 6.1 Introduction

Generative video prediction is a popular and active area of research (Castrejon, Ballas and Courville, 2019; Oprea et al., 2020; Zhou, Dong and El Saddik, 2020) that has recently adopted transformer-based architectures (Carion et al., 2020; Rakhimov et al., 2021; Farazi and Behnke, 2019; Farazi, Nogga and Behnke, 2021) which have led to great progress in language modelling. However, where other work focuses on the state-of-the-art performance a transformer-based model can bring to video generation (Yan et al., 2021; Farazi, Nogga and Behnke, 2021; Rakhimov et al., 2021), I instead explore the visual equivalent of the predictive language modelling training approach that transformers are well known for. I dub this generative pretraining 'visual modelling' (*i.e.* image-sequence-to-image).

This raises many interesting questions. What are the similarities and differences of predictive modelling in vision and language? Under what circumstances could it be easier to predict a frame of a video than the next word of a sentence? Is it always easier to predict a few language tokens than it is to fully generate output pixels? Is language a more information dense modality? Or is a picture worth 1,000 (or 16×16; Dosovitskiy, Beyer et al., 2021) words?

In parallel with similarly motivated work (Ranzato et al., 2014; Chen et al., 2020), I seek to push this conversation to the forefront of the field. However, the remarkable successes of predictive language modelling casts a large shadow. Where language models can generate paragraphs of text comparable to human quality (Brown, Mann, Ryder et al., 2020a), instead video generation models are comparatively primitive. I identify three major barriers to closing this research gap:

*1)* Generating videos instead of language tokens is fundamentally more challenging. Instead of generating confidence scores for a word from a fixed language token vocabulary, video requires precisely predicting values for clusters of pixels, or even entire images.

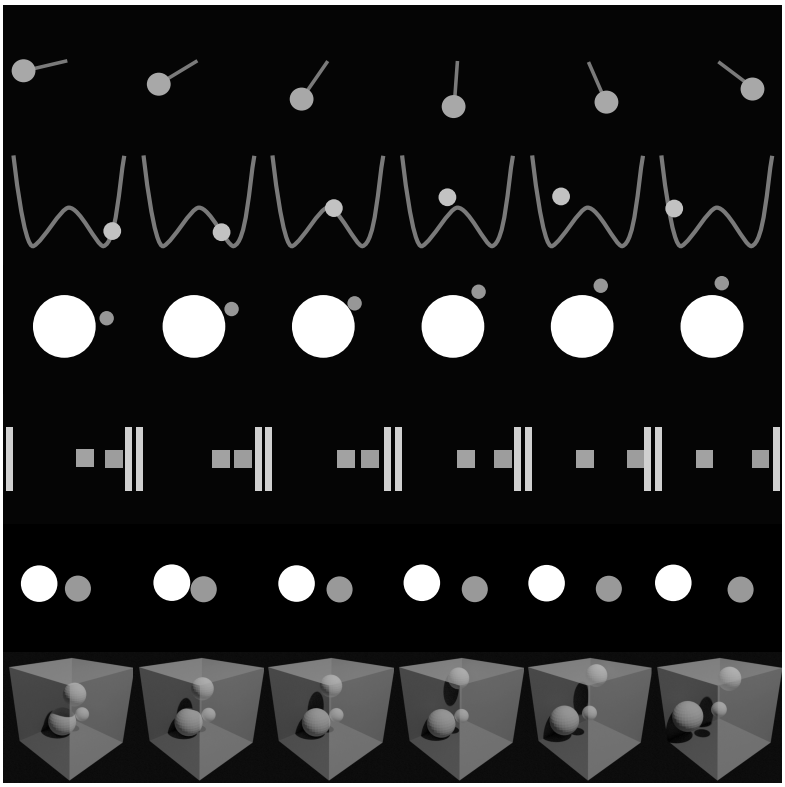


Figure 6.1: Example visualisations of 6 sequential frames from each of my six dynamic simulation video data sets. From top to bottom: **Pendulum**, **Roller** coaster with flight, Mars **Moon**, Colliding **Blocks**, **2D Bouncing** balls, and **3D bouncing** balls.

*2)* Video data sets have a much higher memory and storage overhead than language data sets of comparative scale. Even the most ambitious and well resourced video models down-sample frames to 64×64 (Chen et al., 2020; Yan et al., 2021).

*3)* Though the first few frames of video predictions are impressive (Chen et al., 2020; Yan et al., 2021), the quality of longer term predictions is lacking (*i.e.* 10, 20, 50 frames into the future, see Section 7 of Oprea et al., 2020). This implies that these models do not have a strong understanding of the physical laws underpinning the video.

Together these barriers highlight the often underestimated complexity of video data sets (*e.g.* the simple action of walking involves simulatenous bends and rotations of various body parts) and poor predictive performance resulting from poorly understood visual laws. Motivated by this, I focus on simpler dynamic simulations that can be used to verify the visual understanding that visual modelling pretraining induces. I verify this understanding *qualitatively* in the observed properties of output frames, *quantitatively* with pre-established vision metrics, and *experimentally* with test-tasks that can only be solved if the model understands the appropriate visual laws. To this end, I propose six dynamic simulation datasets for video prediction pretraining (see Figure 6.1) with a total of seven 'probing tasks' defined amongst them (*e.g.* a 2D bouncing ball video data set that also functions as a gravity regression task). I couple the Moving-MNIST (MMNIST) video dataset (Srivastava, Mansimov and Salakhutdinov, 2015) with MNIST classification (Lecun et al., 1998) for a total of seven video data sets with eight probing tasks. As I explore the themes of vision and language modelling, I evaluate these data sets on three appropriate models: A fully convolutional 2D *'CNN'* serving as a baseline inspired from vision, a

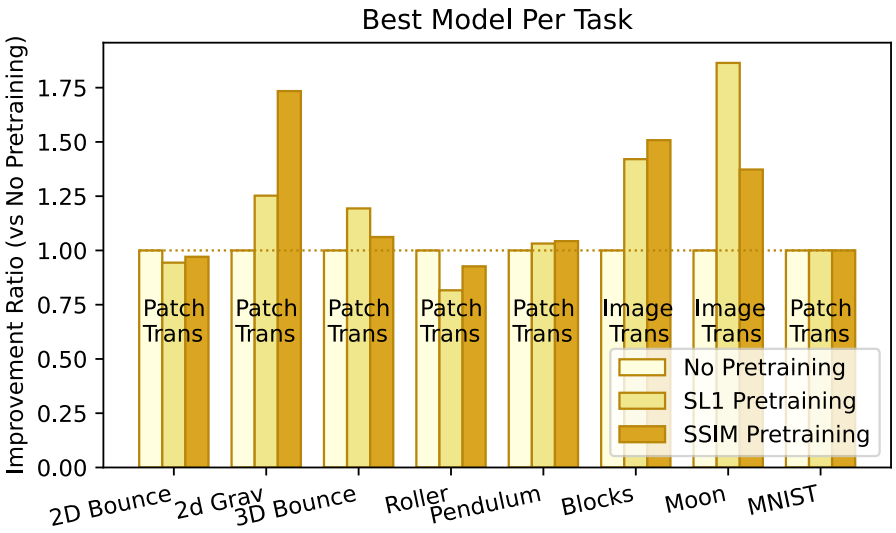



Figure 6.2: The improvement ratio in scores for each task when pretrained on visual modelling, compared with no pretraining, for the best performing model of each task.

language model style *'Image Transformer'* as a baseline inspired from language modelling, and a *'Patch Transformer'* using convolutions and transformer blocks serving as an overlap of both vision and natural language processing (NLP).

I find that appropriately focused and simple data sets can demonstrate the potential of visual modelling pretraining for downstream vision tasks. I find both the convolution based models (in particular the patch transformer) outperform the image transformer in frame generation, highlighting the importance of convolutions in video models. I find that these models can generate physically reasonable simulations over 20 frames into the future, despite a buildup of small errors, demonstrating the potential of long-term video prediction when visual laws are properly understood. The probing experiments demonstrate that pretraining on the visual modelling task induces features that are directly useful to downstream tasks. Furthermore, I find that finetuning models on test-tasks that has been pretrained on visual modelling either does not substantially harm, and often greatly, improves test-task performance. Four of these tasks improve by 20-80% (see Figure 6.2). This demonstrates the potential for predictive pretraining in vision tasks. My implementation is available on GitHub[1].

## 6.2 Related Work

I give an overview of work exploring the overlapping themes of vision and language modelling (*i.e. visual modelling*) and highlight my unique contributions. As I use modern deep learning model architectures to predict videos with visual dynamics and learn physical laws, I briefly summarise both the recent history of *video generation* models, and the state of *visual physics modelling* in deep learning.

[1] `https://github.com/Visual-modelling/visual_modelling`

### 6.2.1 Visual Modelling

Though I coin the term 'visual modelling', other works have explored visual parallels to language modelling. Ranzato et al. (2014) present an early baseline for unsupervised video feature learning, and demonstrate its use in representing motion and deformation. They find poor long-term prediction results as elements tend to a static position. Oord, Li and Vinyals (2018) introduce a contrastive learning strategy for unsupervised feature learning in four domains (including image), and the BigBiGAN model introduced by Donahue and Simonyan (2019), achieves state-of-the-art image generation and representation learning results on ImageNet (Deng, Dong et al., 2009). However, neither of these approaches consider video data. My work is most similar in motivation to that of Chen et al. (2020) who explore unsupervised representation learning for images using a minimal adaption of transformers and vector-quantised Variational Autoencoder (VAE) (Oord, Vinyals and Kavukcuoglu, 2017) architectures. By leveraging a very large amount of computational resources, (a model comparable to GPT-2 *i.e.* 48 layers and ∼1.4B parameters; Radford et al., 2019b) and using auto-regressive next-pixel prediction or masked-pixel prediction training strategies, Chen et al. (2020) present an extensive probing study of the learned representational capacity of layers in their models. They further demonstrate that unsupervised image pretraining leads to state-of-the-art performance on downstream tasks that increases with model scale. Unlike the previously discussed approaches, I aim specifically to pair my visual pretraining data sets with quantifiable and observable test-tasks, allowing me to strongly argue the benefits of pretraining that the experiments demonstrate. Where Ranzato et al. (2014) found poor long-term prediction quality, modern vision architectures and more focused data sets can improve both long-term prediction and downstream task performance. I show that visual modelling pretraining can lead to substantial performance increases on test-tasks even without the large-scale computational resources necessary (Chen et al., 2020) to try and approach the scale of modern language models.

### 6.2.2 Video Generation

The recurrent CNN proposed by Ranzato et al. (2014) for unsupervised frame prediction and filling drew inspiration from early language models (Bengio et al., 2000) and RNNs (Mikolov et al., 2010). Model predictions often tend towards still images after a few frames allegedly due to the local spatial and temporal stationary assumption made by the model. The authors note that predicting beyond a few frames inevitably invokes the curse of dimensionality and argue it could necessitate moving from pixel-wise prediction to higher-level pixel cluster features. Dosovitskiy and Brox (2016) propose a family of deep perceptual similarity metrics 'DeepSiM' to avoid the 'over-smoothed' results of pixel-wise predictions by instead computing distances between image feature vectors. Amersfoort et al. (2017) propose a convolutional network to generate future frames by predicting a transformation based on previous frames and constructing the future frames accordingly, leading to sharper images and simultaneously avoiding the curse of high dimension predictions. Wang, Wu et al. (2020) propose an integrated Bayesian framework to cope with uncertainties caused by noisy observations (*i.e.* perceptual) and forward modelling process (*i.e.* dynamics). Yilmaz and Tekalp (2021) use deformable convolutions (Dai et al., 2017) to try and exploit a larger and more adaptive receptive field as opposed to normal convolutions. The recently released VideoGPT (Yan et al., 2021) is a video generation model which combines vector-quantised VAE and transformer designs with large-scale training setups similar to those used by Chen et al. (2020); *i.e.* similar in scale to Image-GPT and trained on up to 8 Quadro RTX 6000 GPUs. Video-GPT yields very high quality frame predictions on the UCF-101 (Soomro, Zamir and Shah, 2012) and TGIF (Li, Song et al., 2016) data sets . However, due to the inherent difficulty of modelling complex real-world long-term videos, errors in motion still build up.

I acknowledge the difficulties that even richly resourced models encounter and echo Yan et al. (2021): videos are just simply a "hard modelling challenge". I instead focus my efforts on demonstrating what is possible when the visual laws underpinning the video data are kept appropriately

simple. See the work of both Oprea et al. (2020) and Zhou, Dong and El Saddik (2020) for a thorough review and survey of video generation.

### 6.2.3 Visual Physics Modelling

Wu, Lim et al. (2016) collect the 'Physics 101' data set which facilitates models explicitly learning physical properties of objects in videos (*e.g.* mass, acceleration, and friction). Where they focus on encoding physical laws into neural networks, I additionally explore if generative visual modelling is a sufficient or desirable method to induce a quantifiable understanding of these laws. Neural networks have successfully modelled a variety of dynamic systems using images: *e.g.* fluid flow (Tompson et al., 2017), Lyapunov functions (Manek and Kolter, 2020), motion flow (Bezenac, Pajot and Gallinari, 2018) and precipitation nowcasting (Shi et al., 2017). Li, Kovachki et al. (2021) propose a novel Fourier neural operator that can learn Burger's equations, Darcy flow, and Navier-Stokes with differing input image resolutions. Recently, Wang, Walters and Yu (2021) push for more generalisable physical modelling with their proposed multi-task DyAd approach.

## 6.3 Models and Configurations

To explore the visual parallels of language modelling, I focus on both CNNs (due to their long history of state-of-the-art success in computer vision) and Transformers (because of their well established dominance in language modelling). To this end, I experiment on 3 models:

*1)* A fully convolutional *'CNN'* model with skip connections as a baseline model from vision (Figure 6.3).

*2)* A multi-head attention transformer typical of language modelling with minimal redesigns to accommodate video prediction to serve as a candidate from language models. I call this the *'image transformer'* (Figure 6.4).

*3)* The recently proposed SegFormer (Xie et al., 2021) transformer minimally adapted from semantic segmentation to video generation as a transformer model directly designed for use in vision. I call this adaptation the *'patch transformer'* (Figure 6.5).

Each of these models has been designed or adapted to take a sequence of video frames as input, and output predictions for the next frame (further described in Section 6.4).

### 6.3.1 Fully Convolutional 2D CNN

Introduced by Long, Shelhamer and Darrell (2015), fully convolutional neural networks (FCN) use 'deconvolution' layers (Zeiler and Fergus, 2014) *i.e.* convolutional layers with fractional strides. Deconvolution layers can be used to 'reverse' the convolution layers and generate a full sized output. Note that upsampling between layers can be learned or can be fixed (*e.g.* bilinear upsampling). As such, FCNs offer relatively inexpensive forward and backward computation, and a reversal of convolution layers back up to the original input dimensions for outputs. My CNN baseline model (depicted in Figure 6.3) is a U-Net style (Ronneberger, Fischer and Brox, 2015) FCN with skip connections (He et al., 2016). Given a sequence of $m$ frames of a video, the model takes as input the $m$ 64×64 grayscale input frames in temporal order as inputs for $m$ channels into the first of a series of U-Net style double-convolution units. Each subsequent step halves the image resolution and doubles the number of channels from a starting factor of 64. The upwards pass uses bilinear upsampling and halves the number of channels, mirroring the downward pass in reverse, resulting in the final output frame.

### 6.3.2 Image Transformer

I adapt the model introduced by Vaswani et al. (2017) and deploy it on sequences of images that form a video instead of word embeddings that form language (Figure 6.4). I use every 64×64 pixel image as a single token consisting of 4096 features. Unlike the original transformer that

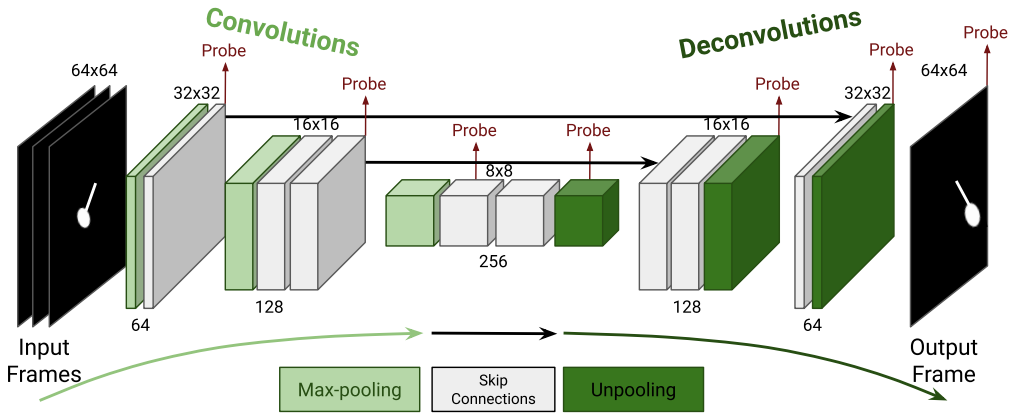



Figure 6.3: Fully Convolutional 2D **CNN** model. Each convolution unit is made from two convolution layers, *i.e.* an initial convolution layer that changes the input resolution, followed by another of kernel size 1×1 that does not. The arrows labelled 'Probe' indicate which points in the network are extracted to form linear probes used in Section 6.4.2.

was trained on a language translation task, here the output is only conditioned on its previous video frames. I choose not to use an encoder and only use the decoder to predict the next frame output frame because the input and output image sequences are not synchronised over the same time span (there are $m$ input frames translated to a single output frame; *i.e.* $m \neq 1$). Since the tokens represent images, I use a pixel regression layer at the end of the architecture. Using a dedicated pixel regression layer allows the output tokens to represent images while alleviating this requirement from the transformer blocks. Although I can train using batches with $m$ input frames and predict one output frame, as in the CNN and patch transformer, the model can be instead trained with the entire sequence at once. This can be done by predicting the next frame for every input using a masked multi-head attention where every output is only conditioned on the past frames. I find negligible difference in performance between the two approaches, and therefore decide to use the fixed $m$ input and one output for a more direct comparison with the other two models. I found negligible difference in performance when using more than two transformer blocks. Similarly, I found a smaller number of heads in the multi-head attention block to be beneficial, and thus only used four.

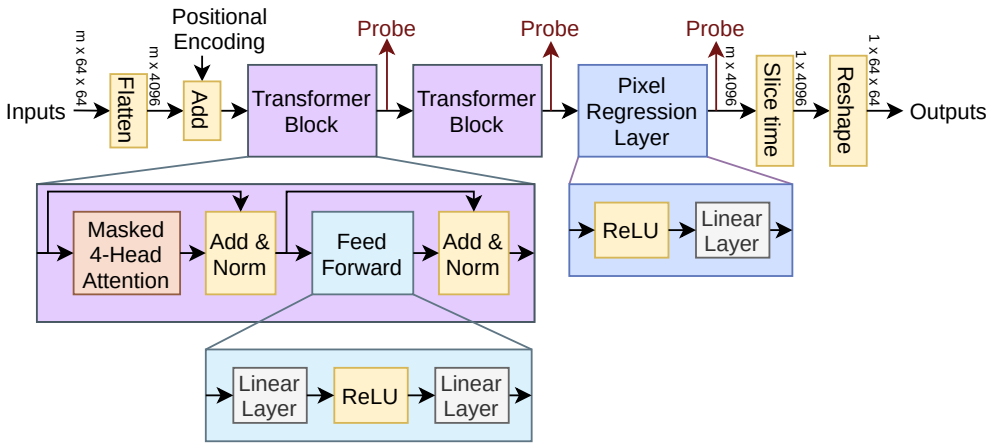



Figure 6.4: The **Image Transformer** model serves a candidate from language modelling. I use 64×64 images as tokens. The arrows labelled 'Probe' indicates which points in the network are extracted to form linear probes used in Section 6.4.2.

### 6.3.3 Patch Transformer

I use a transformer-based semantic segmentation model and modify it for visual modelling (Figure 6.5). Instead of feeding the model RGB images with three channels, I use the sequence of video frames as the input channels and predict the following frame as the output. Unlike the Image Transformer that applies the attention layers across the time sequence of video frames, the patch transformer applies the attention layers across patches of the images.

I base the patch transformer architecture on the SegFormer model (Xie et al., 2021) with the following light modifications. Since my task requires the same resolution of the input and output images and the SegFormer outputs at 1/4 resolution, I balance this by predicting 16 times as many channels and fold each pixel with 16 channels into a 4×4 patch with one channel.

When compared to the original SegFormer study, the smaller size of images used allows me to use a reduction ratio 'R' of 1 in the efficient attention layer (*i.e.* not compromising the representational capacity). This makes it functionally equivalent to a standard multi-head attention

layer.

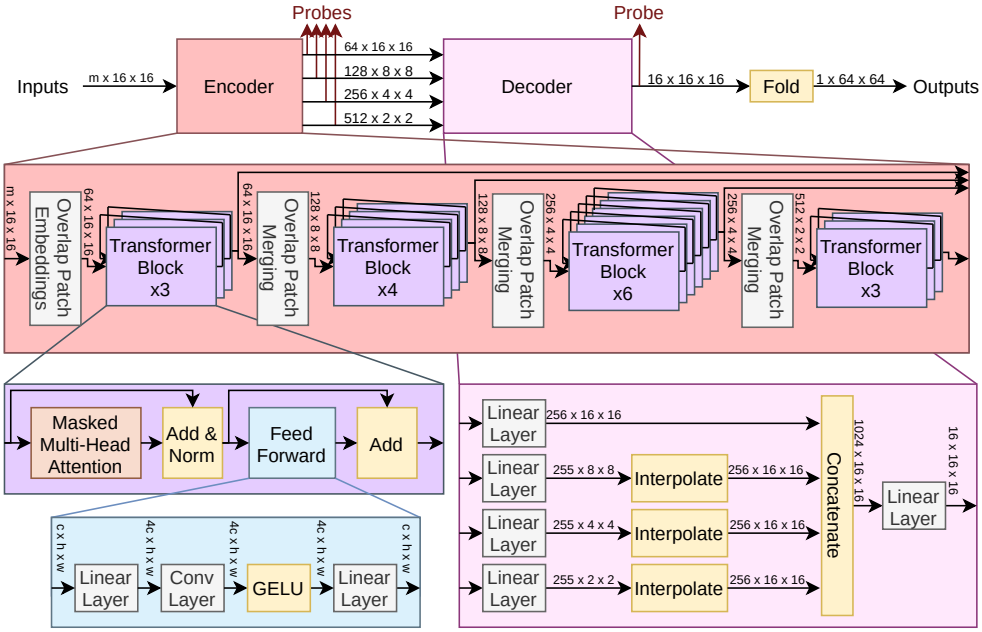



Figure 6.5: The **Patch Transformer** model, adapted from the SegFormer (Xie et al., 2021) for video generation. The arrows labelled 'Probe' indicate which points in the network are extracted to form linear probes used in Section 6.4.2.

### 6.3.4 Datasets

To explore the potential of visual modelling, given the previously discussed limitations of the field, I focus my experiments on data sets that are both:

*1)* Simple dynamic simulations that can be used for video prediction. This is akin to language model pretraining in NLP.

*2)* Naturally affiliated with a classification or regression task accompanying the modelling. These are used both as downstream tasks to finetune the model, as well as for probing what the model has learnt.

See Table 6.1 for examples of the datasets and details on the test-tasks they pair with.

I ablate on two real world data sets, CMU Motion Capture 'MOCAP' data set[2] and the human motion database 'HMDB-51' (Kuehne et al., 2011) in order to help further demonstrate the limitations of the three models.

### 6.3.5 2D and 3D Bouncing Balls

The 2D bouncing data set consists of videos with 1-3 balls which can collide with both each other and the borders of the image. In this explicit Euler simulation, I vary: the number of balls, ball radius (per ball), initial position (per ball), initial velocity (per ball), gravity strength, gravity direction, background colour, and ball colour (per ball). I define two tasks on this 2D data set: *total number of bounces prediction* and *y-directional gravity prediction*. Where the 2D version represents the balls as simple circles, I extend this approach to 3D, rendering the balls using realistic lighting from a single light source. This 3D scenario is designed to be a more challenging dataset as balls occlude each other and cast shadows on both the environment boundary and on other balls. I define one test-task for this 3D bouncing data set: *total number of bounces prediction*.

### 6.3.6 Myphysicslab

Myphysicslab[3] is a series of open-source animated physics simulation software involving pendulums, springs, collisions, and more. These simulations are calculated using the Runge-Kutta method (Butcher, 1996). I experiment with the following simulations:

[2] `http://mocap.cs.cmu.edu/`
[3] https://www.myphysicslab.com

**Mars Moon**

A simplified simulation of an asteroid orbiting a moon using a rigid body simulation. I vary the initial velocity, moon radius, moon mass, and asteroid radius. The test-task is to *predict the mass of the moon.*

**Colliding Blocks**

Simulates two blocks that move along a single axis colliding with both the boundary walls and each other. I vary the masses of *one* of the blocks (leaving the other block mass fixed), the starting positions, and starting velocities. The test-task is to *predict the difference in the masses of the two blocks.*

**Pendulum**

A single pendulum modeled as a point mass at the end of a massless rod. I vary the initial angle, gravity strength, pendulum length, and pendulum mass. The test-task is to *predict the gravity acting on the pendulum.*

**Roller Coaster with Flight**

A ball of mass $M$ is released down a curved track under gravity $g$ following $F = M \cdot g \cdot \cos(\theta)$. The ball can switch to free flight when the acceleration normal to the curve is greater than $v^2/k$ (where $v$ is the velocity of the ball, and $k$ is the radius of curvature at the current point along the curve). I vary the gravity strength and track position. The test-task is to *predict the strength of gravity acting on the ball.*

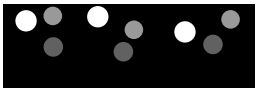
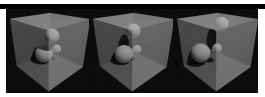
| Data set | Example | # Videos | Vid Length | Affiliated Test-Task(s) |
|---|---|---|---|---|
| **2D Bouncing** |  | 20,000 | 60 | **Bounce Regression:** *59 input frames.* Count total bounces demonstrated in the input video. Both ball-to-ball and ball-to-wall bounces, capped at a maximum of 50. **Gravity Regression:** *5 input frames.* Predict the gravity demonstrated in the given 5 frames. Gravity is in the $y$ axis with 7 potential values [-3e-4, -2e-4, $\cdots$, 3e-4]. |
| **3D Bouncing** |  | 10,000 | 100 | **Bounce Regression:** *99 input frames.* Count total bounces demonstrated in the input video. Both ball-to-ball and ball-to-wall bounces, capped at a maximum of 50. |
| **Roller** |  | 10,000 | 100 | **Gravity Regression:** *5 input frames.* Predict the gravity demonstrated in the given 5 frames. Gravity is in the $y$ axis with 201 potential values [0, 0.5, 1, $\cdots$,100]. |
| **Pendulum** |  | 10,000 | 100 | **Gravity Regression:** *5 input frames.* Predict the gravity demonstrated in the given 5 frames. Gravity is in the $y$ axis with 41 potential values [0, 0.5, 1.0, $\cdots$, 20]. |
| **Blocks** |  | 10,000 | 100 | **Block Mass Difference Regression:** *49 input frames.* Two blocks of different masses move towards each other on a smooth surface and collide. Predict the difference of the masses between the blocks, positive or negative (positive direction is fixed). Block 1 is always of mass 10, block 2 takes 39 different masses [0.5, 1.0, $\cdots$, 19.5]. |
| **Moon** |  | 10,000 | 100 | **Moon Mass Regression:** *5 input frames.* Predict the gravity demonstrated in the given 5 frames. Gravity acting on the small moon towards the centre of a planet. Masses have 26 potential values [70, 75, 80, $\cdots$,195]. |
| **MMNIST** |  | 5,200 | 100 | **MNIST Classification:** *1 input frame.* Input MNIST frame. Copied 5 times for a model pretrained on 5-1 input-ouput for example. |

Table 6.1: Further details of the data sets and their affiliated test-tasks. The constants for predictions for all test-tasks (aside from MNIST classification) are normalised such that the standard deviation of constants across each individual data set is 1.

### 6.3.7 Moving MNIST

Moving MNIST (MMNIST[4]) refers to a video data set (Srivastava, Mansimov and Salakhutdinov, 2015) where one or more MNIST (Lecun et al., 1998) digit(s) are moving around a black background (overlapping each other and bouncing off the frame boundaries). The digits move at fixed constant velocity with no friction or gravity acting on them. The original data set was 10,000 sequences with two digits in each. I adapt code[5] to generate an MMNIST data set containing videos with 1-3 digit(s). MMNIST serves as a simple dynamics data set that can be naturally considered alongside a vision test-task: MNIST classification.

### 6.3.8 CMU Motion Capture

The CMU Motion Capture data set comprises of videos focusing on the motion of humans and objects. I extract images from the 937 videos[6] at 10 frames-per-second. I convert the coloured frames to grayscale and then downscale and crop them to a resolution of 64×64, in line with the other data sets.

### 6.3.9 HMDB-51

Motivated to challenge the high performance of models on the relatively simple action data sets of the time, the Human Motion Recognition database (HMDB-51) (Kuehne et al., 2011) is a large action video data set with 51 action categories. HMDB-51 aims to better capture the "richness and complexity of human actions", with videos including more challenging clutter and occlusion not typical of action benchmarks at the time. Categories include 'brush hair', 'sword exercise', and 'jumping'. I extract images at 10 frames-per-second and convert and crop the images to grayscale in 64×64 resolution.

[4] http://www.cs.toronto.edu/~nitish/unsupervised_video/
[5] https://gist.github.com/tencia/afb129122a64bde3bd0c
[6] http://mocap.cs.cmu.edu/allmpg/

## 6.4 Experiments

My experiments are carried out on the same three model architectures but are comprised of two separate phases of training:

**Visual Modelling Pretraining** - a generative training strategy intended to mirror language modelling and to induce an understanding of visual dynamics.

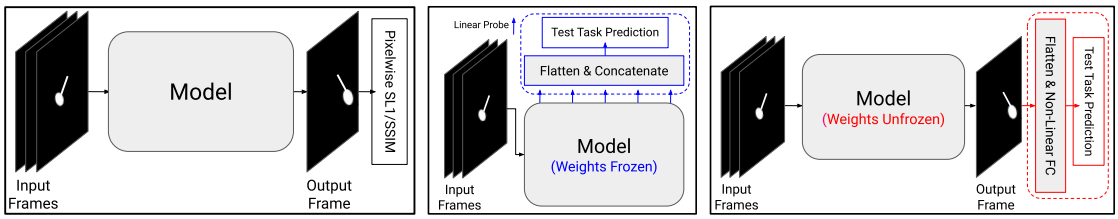
These are followed by **Downstream Test Tasks**, a classification or regression task paired with their appropriate modelling counterpart that function as finetuned vision tasks.

Together these tasks allow exploration of which laws of dynamics current vision benchmarks can model, what information and understanding visual modelling (*i.e.* next-frame prediction) *induces*, and to what extent this *induced understanding* is desirable as a starting point for downstream vision tasks.

### 6.4.1 Visual Modelling Pretraining

Given a sequence of $m+1$ frames of a video, the model takes as input the first $m$ 64×64 grayscale input frames in temporal order and predicts as output the next frame. This predicted frame is compared against the final (ground truth) frame in the $m+1$ sequence. A value is predicted for each pixel (*i.e.* dense prediction) and the sigmoid function is used as activation for the outputs of the final layer, which are then multiplied by 255 to create the resulting output grayscale image. I use either smooth-$L1$ (SL1) or Structural Similarity (SSIM) as loss functions. As SSIM takes values between -1 and 1, and should be maximised, I reformulate it as a minimisation problem (as in Equation 6.1) in order to use it as a loss function:

$$\mathcal{L}_{\text{SSIM}} = 1 - \frac{1 + SSIM}{2} \in [0, 1]. \tag{6.1}$$




(a) Modelling (b) Linear Probes (c) Direct Learning

Figure 6.6: The three different experimental setups. All three of these model architectures can be adapted to visual modelling (*i.e.* video prediction; Figure 6.6a), and the accompanying test-tasks with either linear probing of frozen models (Figure 6.6b) or directly learning and finetuning on the test-task (Figure 6.6c).

Using the three models introduced in Section 6.3, I perform modelling experiments (see Figure 6.6a) on the nine data sets introduced in Section 6.3.4. The values of $m$ in my experiments (*i.e.* number of input frames) depends on the downstream test-task it will be paired with. For example, the 2D bouncing, roller, pendulum, and moon data sets all have downstream test-tasks involving gravity, thus 5 input frames should be sufficient to demonstrate gravity. The 2D and 3D bouncing data sets have 'counting bounces' tasks associated with them, thus $m = 59$ and 99 for 2D and 3D data sets respectively (*i.e.* the length of the video clip; it is necessary to see the entire clip to predict the total number of bounces). See Table 6.1 for further details. To assess the performance of the modelling task, I consider the Peak Signal-to-Noise Ratio (PSNR) (Horé and Ziou, 2010), Structural Similarity (SSIM) (Zhou Wang et al., 2004), and $L1$ scores between the predicted frame and the ground truth frame. I do not consider metrics such as Learned Perceptual Image Patch Similarity (LPIPS) (Zhang, Isola et al., 2018) and Fréchet Video Distance (FVD) (Unterthiner et al., 2019). LPIPS is a deep model based metric that is only well-defined on RGB images, and FVD requires an image resolution of at least 224×224.

### 6.4.2 Test-Tasks

Each of the modelling data sets is designed with a complimentary test-task in mind (except 2D bouncing which has two), and I further pair the MMNIST data set with the test-task of MNIST classification for a total of 8 test-tasks. I apply a cross entropy loss for MNIST classification and a smooth-$L1$ loss with $\beta = 0.01$ for all other tasks. In this subsection I describe the three categories of test-task experiments: random baselines, frozen model probing, and direct training or finetuneing.

#### Random Scores for Tasks

To contextualise the scores for each task, I ablate with three scenarios designed to give a lower bound on performance (seen on the first, second, and third lines respectively of each section in Tables 6.3 and 6.4):

*1)* Constant Output: I instantiate a layer of biases and optimise them directly on the loss of each task. This scenario takes no inputs but *is* given the ground truth. This should theoretically learn to mimic the average output of values that the ground truth alone would induce.

*2)* Image + Linear Layer: I flatten the input images and pass them into a trainable linear layer.

*3)* Frozen Random Model: I randomly instantiate the full model and freeze the weights. I pass the loss function through trainable linear probes in the same way as in the probing experiments.

#### Probing Frozen Models

I seek to ascertain if training on a given modelling data set induces an understanding of the appropriate physical laws (*e.g.* does pretraining on bouncing balls data set induce a verifiable understanding of gravity?). To this end, I freeze the weights of a pretrained model and repurpose

it for the appropriate test-task (see Figure 6.6b) by flattening and concatenating the outputs of each substantial layer throughout the network (see 'Probe' arrows in Figures 6.3, 6.4 and 6.5) and passing them through a trainable linear layer and into the appropriate loss function. It has been shown that CNN and transformer-based language models 'learn representations that vary with network depth' (Peters, Neumann, Zettlemoyer et al., 2018), and that there is varying transferability of representations in different layers of language models (Liu, Gardner et al., 2019). This motivates forming the linear probe from the output of all substantial blocks of the networks. Though the model as a whole may contain the information needed to solve a test-task, only considering the outputs of the final layer can be insufficient as that layer is likely to be focused on solving the pixel regression.

**Finetuning for Downstream Test-Tasks**

The test-tasks allow verification of whether pretraining provides an advantageous starting point for downstream vision tasks when compared with random initialisation. Given a network that is either pretrained on visual modelling or randomly instantiated: I unfreeze the weights, flatten the outputs of the final layer, and pass them through a non-linear (GELU) fully connected unit with dropout and batch normalisation. Lastly, the outputs of the fully connected unit are passed to the appropriate loss function (see Figure 6.6c).

## 6.5 Results and Discussion

In this section, I discuss the results of my visual modelling and test-task experiments.

### 6.5.1 Modelling Quality

I consider the quality of the generated frames by computing PSNR, SSIM and $L1$ scores between the predictions and the ground truth frame (Figure 6.7). To allow closer inspection of these, I

also provide these results in tabular form (Table **??**). In the following subsections, I discuss the differences in modelling quality with respect to the different data sets, models, and losses.

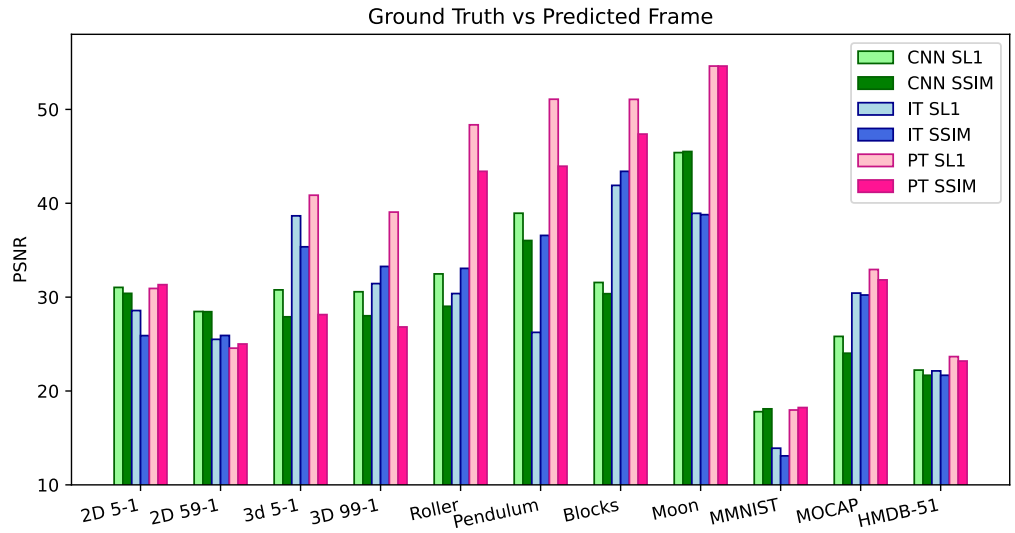
**Modelling Quality by Data Set**

The modelling tasks yielding the highest scores are roller, pendulum, blocks, and moon. This is most obviously seen in the PSNR values from Figure 6.7a (over 8 PSNR higher than the other data sets for the best model). This trend is echoed with the very high SSIM ($\sim$0.998; see Figure 6.7b) and lower $L1$ scores (see Figure 6.7c). I believe this is because the background for these tasks is always black, and there are less structural variations than the other data sets. Next are the 2D and 3D bouncing datasets which score slightly lower than the top performing four data sets. Though I expected the 3D bouncing data set to be more complicated than the 2D version because of the added potential for occlusion and $z$-dimensional movement, the 3D bouncing data set yields slightly higher scores than the 2D bouncing data set ($\sim$2-10 PSNR higher for the Image and Patch Transformers). I argue that although 3D bouncing must model motion in an extra dimension, the fixed background and colour scheme of the 3D data set allow an extra boost to metric scores when compared to the 2D bouncing data set which has varying the background and ball colours. I find that increasing the number of input frames ($m$) for the 2D and 3D bouncing data sets from 5 to 59 and 99 respectively causes a very minor but consistent decrease in all three scores (by comparing the $1^{st}$-$2^{nd}$, and $3^{rd}$-$4^{th}$ entries of each sub-figure in Figure 6.7 respectively), implying there is little to gain from increasing the input context for these modelling tasks. The real world MOCAP and HMDB-51 data sets score lower than all other data sets except MMNIST. Though my models fail to capture the far more complex motion laws dictating these real world data sets, the single predicted frame from these real world data sets does not change substantially from the inputs. I argue that the movements in these videos, despite being underpinned by complex motion, are relatively small as the entities in them are small. This means that predicted movements are often smaller in terms of absolute pixel variations when

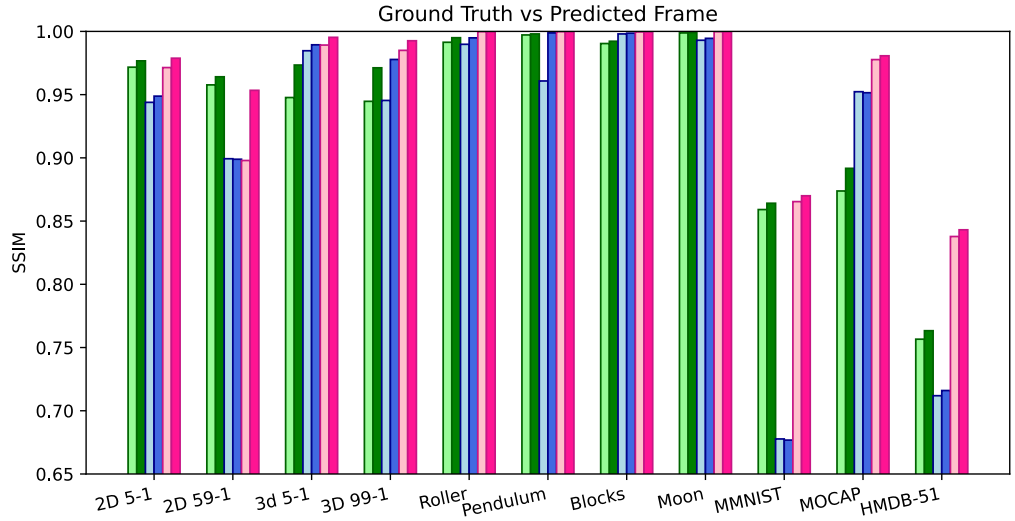
compared to the movements of the high-contrast shapes seen in my simulation data sets, causing smaller penalties in the three metrics than might be expected. I discuss the inability of my models to model *long-term* real-world movements in Section 6.5.2. The MMNIST modelling task scores the lowest on the three metrics of all data sets. Though intuitively MMNIST should score higher than the real world data sets, I believe this result is unsuprising for several reasons. MMNIST features a black background contrasted with multiple white digits, meaning that a prediction with a white digit in a slightly incorrect location will yield very high absolute differences in raw pixel values and thus lower metric scores. Furthermore, the shape and structure of digits is more complex than the circles and spheres typical of the other data sets. The holes and lines tend to fill and fade, and each digit is moving in different directions. As explored later in Section 6.5.2, the long-term predictions for MMNIST are more physically realistic than those generated from the real world data sets despite these lower first-frame prediction metrics. This stresses how crucial it is to complement metrics for first-frame prediction with long-term predictive analysis in video generation.

**Modelling Quality by Model**

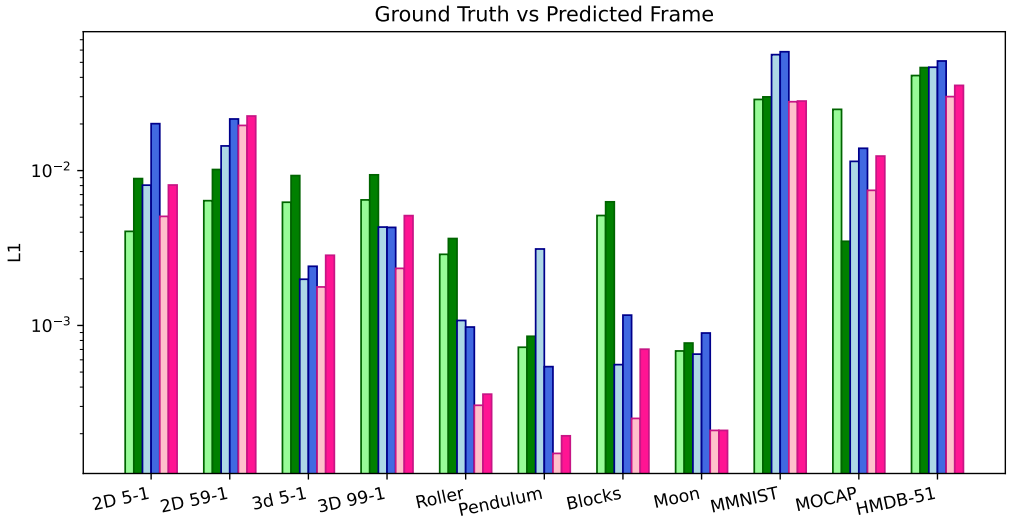
The patch transformer consistently has the best PSNR, SSIM, and $L1$ metric scores for all but one of my experiments (2D bouncing with $m = 59$). The CNN and image transformer alternate as the second best model across both the easier and harder tasks. The patch transformer gives moderately higher scores for 2D bouncing ($m = 5$), 3D bouncing ($m = 5$), MMNIST, and the real world data sets ($\sim$1-3 higher PSNR scores). On the four data sets with best scores overall (roller, pendulum, blocks, and moon), the patch transformer demonstrates considerably higher scores when compared with the other two models. This implies that the patch transformer excels at replicating the more controlled conditions in these data sets, indicated by its very high SSIM scores ($\sim$0.999).




(a) PSNR↑




(b) SSIM↑




(c) L1↓

Figure 6.7: Metrics calculated between the ground truth and the predicted frame on each modelling data set. ↑ (↓) indicates that a higher (lower) score is better.

**Modelling Quality by Loss**

I see that models trained with the SSIM-based loss demonstrate higher SSIM scores on their predictions compared to those pretrained on SL1. This can be seen in Figure 6.7b, where the higher values of the dark bars of each colour represent SSIM training for each model variant (∼0.01-0.05 increase consistently). Conversely, models trained with the mean pixelwise SL1 loss have a lower (*i.e.* better) mean pixelwise $L1$ score as seen in the lower lighter bars representing SL1 training in Figure 6.7c. This improvement in each metric of models trained with that metric's respective loss counterpart further highlights how limited any single metric is in demonstrating the quality of generated images. Despite each score's preference for its own loss function, I find that SL1 trained models almost always give a higher PSNR than their SSIM counterparts (Figure 6.7a). This more pronounced covariance of PSNR and SL1 scores is expected behaviour as PSNR tends to infinity as mean squared error (and hence SL1) tends to zero. This suggests that PSNR is optimised for by an SL1 loss.

### 6.5.2 Long-Term Self-Output Prediction

As previously described, the models are trained using subsets of clips of size $(m + 1)$ where $m$ is the number of input frames and the final frame serves as ground truth for the prediction. In order to gauge the model's capacity for long-term video generation, I consider 'self-output' experiments, *i.e.* for a video in the test set:

1. Generate the predicted frame from the first $m$ frames.
2. Create, as new inputs, the starting $m$ frames with the first frame removed and the newly generated frame added to the end.

This effectively tests the model's capacity to continue generating the video from the initial $m$ frames by using the predicted frame to shift the next inputs into the future one frame at a time.

**Algorithm 1** Self-Output Visualisation.

**Input:** *video*, $m$

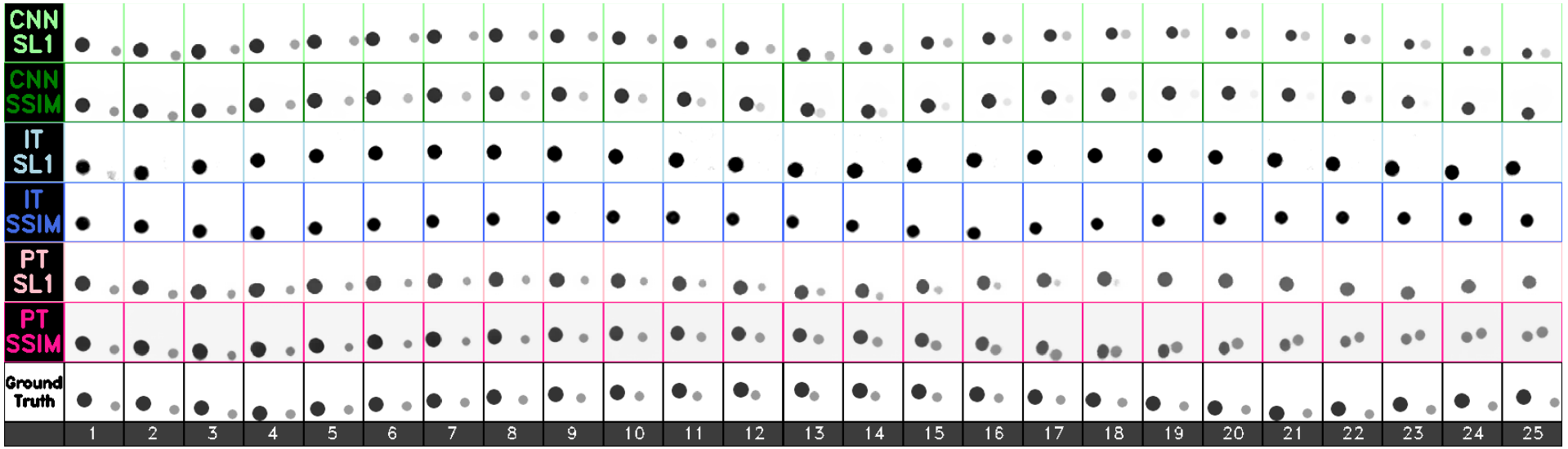
```
totalFrames ← length(video)
inputs ← video[0 : m]                          ▷ Python-style slicing.
for i ← m to totalFrames do
    output ← model(inputs)
    groundTruth ← video[i]
    plotTogether(groundTruth, output)          ▷ See Figure 6.8.
    inputs ← concatenate(inputs[1 : m], output)   ▷ Shift one frame into the future.
end for
```

Using the process described in Algorithm 1, I create a side-by-side comparison of the predicted video frames and their ground truth counterparts in Figure 6.8. As it would be intractable to discuss detailed behaviours from each model on each data set and with both loss functions, I focus on the major trends and properties casn be observed in the generated videos. I invite readers to explore the complete set of self-output videos and metrics for each test set [7]. In the following subsections, I discuss the differences in self-output predictions with respect to the different models and losses.

[7] https://github.com/Visual-modelling/visual_modelling#all-self-output-gifs

| Metric | CNN | | Img Trans | | Patch Trans | |
|---|---|---|---|---|---|---|
| | **SL1** | **SSIM** | **SL1** | **SSIM** | **SL1** | **SSIM** |
| **2D Bouncing** $m = 5$ | | | | | | |
| PSNR↑ | 31.036 | 30.400 | 28.572 | 25.900 | 30.924 | 31.328 |
| SSIM↑ | 0.9717 | 0.9767 | 0.9439 | 0.9488 | 0.9714 | 0.9788 |
| L1↓ | 4.051e-3 | 8.879e-3 | 8.053e-3 | 2.009e-2 | 5.062e-3 | 8.069e-3 |
| **2D Bouncing** $m = 59$ | | | | | | |
| PSNR↑ | 28.468 | 28.446 | 25.504 | 25.915 | 24.564 | 25.007 |
| SSIM↑ | 0.9577 | 0.9641 | 0.8993 | 0.8989 | 0.8979 | 0.9534 |
| L1↓ | 6.393e-3 | 1.016e-2 | 1.443e-2 | 2.154e-2 | 1.957e-2 | 2.251e-2 |
| **3D Bouncing** $m = 5$ | | | | | | |
| PSNR↑ | 30.772 | 27.903 | 38.661 | 35.631 | 40.859 | 28.138 |
| SSIM↑ | 0.9477 | 0.9734 | 0.9847 | 0.9894 | 0.9892 | 0.9953 |
| L1↓ | 6.247e-3 | 9.284e-3 | 1.988e-3 | 2.410e-3 | 1.773e-3 | 2.839e-3 |
| **3D Bouncing** $m = 99$ | | | | | | |
| PSNR↑ | 30.571 | 28.013 | 31.443 | 33.269 | 39.060 | 26.823 |
| SSIM↑ | 0.9447 | 0.9712 | 0.9454 | 0.9778 | 0.9850 | 0.9926 |
| L1↓ | 6.468e-3 | 9.388e-3 | 4.318e-3 | 4.297e-3 | 2.337e-3 | 5.120e-3 |
| **3D Bouncing** $m = 99$ | | | | | | |
| PSNR↑ | 30.571 | 28.013 | 31.443 | 33.269 | 39.060 | 26.823 |
| SSIM↑ | 0.9447 | 0.9712 | 0.9454 | 0.9778 | 0.9850 | 0.9926 |
| L1↓ | 6.468e-3 | 9.388e-3 | 4.318e-3 | 4.297e-3 | 2.337e-3 | 5.120e-3 |
| **Roller** $m = 5$ | | | | | | |
| PSNR↑ | 32.475 | 29.023 | 30.381 | 33.063 | 48.358 | 43.404 |
| SSIM↑ | 0.9914 | 0.9950 | 0.9898 | 0.9949 | 0.9998 | 0.9998 |
| L1↓ | 2.879e-3 | 3.642e-3 | 1.077e-3 | 9.768e-4 | 3.051e-4 | 3.600e-4 |
| **Pendulum** $m = 5$ | | | | | | |
| PSNR↑ | 38.941 | 36.031 | 26.246 | 36.574 | 51.085 | 43.948 |
| SSIM↑ | 0.9972 | 0.9980 | 0.9608 | 0.9988 | 0.9998 | 0.9998 |
| L1↓ | 7.231e-4 | 8.512e-4 | 3.117e-3 | 5.420e-4 | 1.492e-4 | 1.938e-4 |
| **Blocks** $m = 49$ | | | | | | |
| PSNR↑ | 31.561 | 30.365 | 41.905 | 43.402 | 51.069 | 47.371 |
| SSIM↑ | 0.9904 | 0.9922 | 0.9980 | 0.9986 | 0.9996 | 0.9996 |
| L1↓ | 5.128e-3 | 6.285e-3 | 5.585e-4 | 1.166e-3 | 2.510e-4 | 7.025e-4 |
| **Moon** $m = 5$ | | | | | | |
| PSNR↑ | 45.399 | 45.515 | 38.928 | 38.785 | 54.618 | 54.618 |
| SSIM↑ | 0.9988 | 0.9990 | 0.9930 | 0.9945 | 0.9998 | 0.9998 |
| L1↓ | 6.844e-4 | 7.699e-4 | 6.529e-4 | 8.937e-4 | 2.100e-4 | 2.100e-4 |
| **MMNIST** $m = 5$ | | | | | | |
| PSNR↑ | 17.794 | 18.097 | 13.904 | 13.089 | 17.975 | 18.237 |
| SSIM↑ | 0.8591 | 0.8641 | 0.6777 | 0.6768 | 0.8654 | 0.8700 |
| L1↓ | 2.880e-2 | 2.992e-2 | 5.607e-2 | 0.05853 | 2.784e-2 | 2.812e-2 |
| **MOCAP** $m = 5$ | | | | | | |
| PSNR↑ | 25.816 | 24.024 | 30.431 | 30.226 | 32.943 | 31.835 |
| SSIM↑ | 0.8738 | 0.8917 | 0.9523 | 0.9515 | 0.9777 | 0.9807 |
| L1↓ | 2.485e-2 | 3.495e-3 | 1.147e-2 | 1.393e-2 | 7.456e-3 | 1.241e-2 |
| **HDMB51** $m = 5$ | | | | | | |
| PSNR↑ | 22.229 | 21.675 | 22.141 | 21.673 | 23.664 | 23.191 |
| SSIM↑ | 0.7566 | 0.7633 | 0.7119 | 0.7160 | 0.8378 | 0.8431 |
| L1↓ | 4.109e-2 | 4.627e-2 | 4.649e-2 | 5.102e-2 | 3.004e-2 | 3.549e-2 |

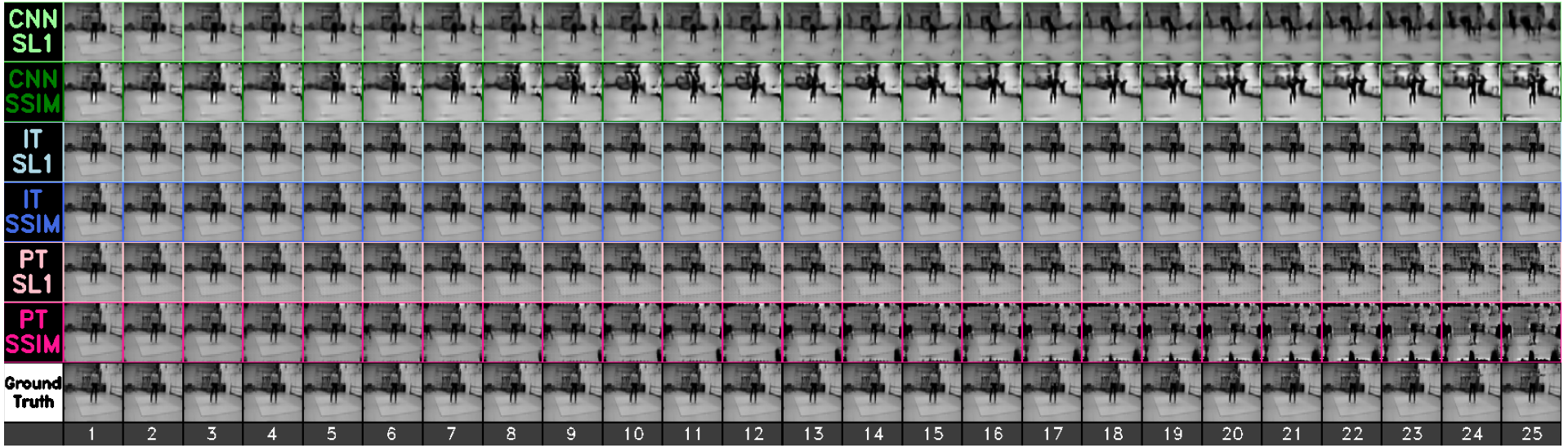
Table 6.2: Metrics between the first generated image and its respective ground truth. All metrics are reported from the best epoch of the models respective loss. The L1 metric is calculate with mean reduction. ↑ (↓) indicates higher (lower) is better.



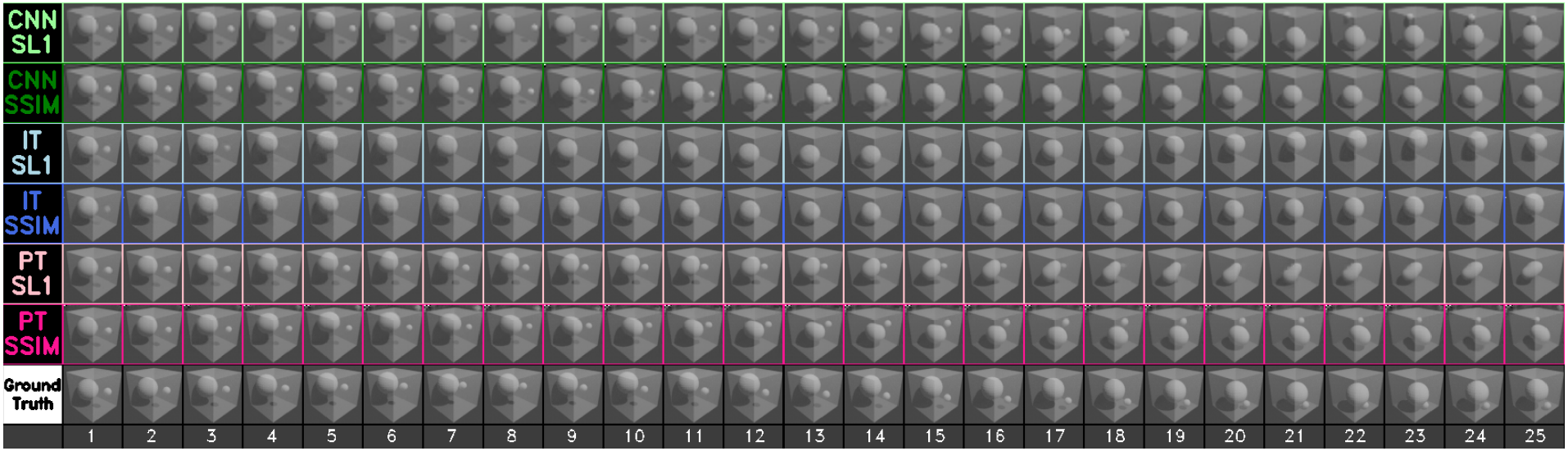

(a) 2D Bouncing


(b) MMNIST

Figure 6.8: Comparison of the first 25 generated frames of each model ($m = 5$) visualised alongside the ground truth.



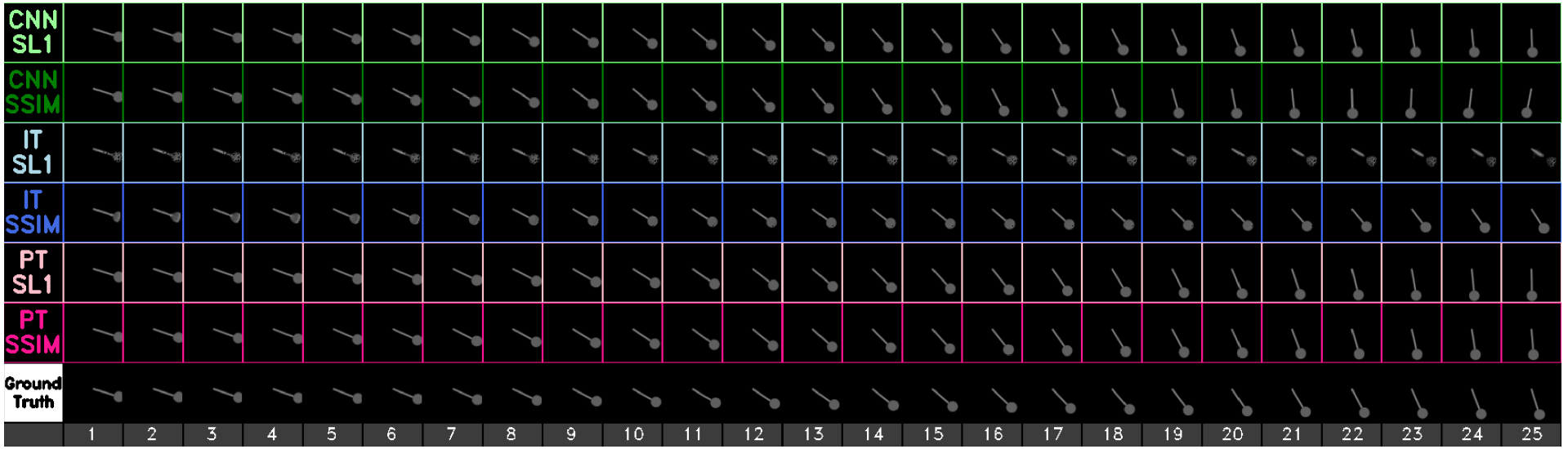

(c) MOCAP




(d) 3D Bouncing

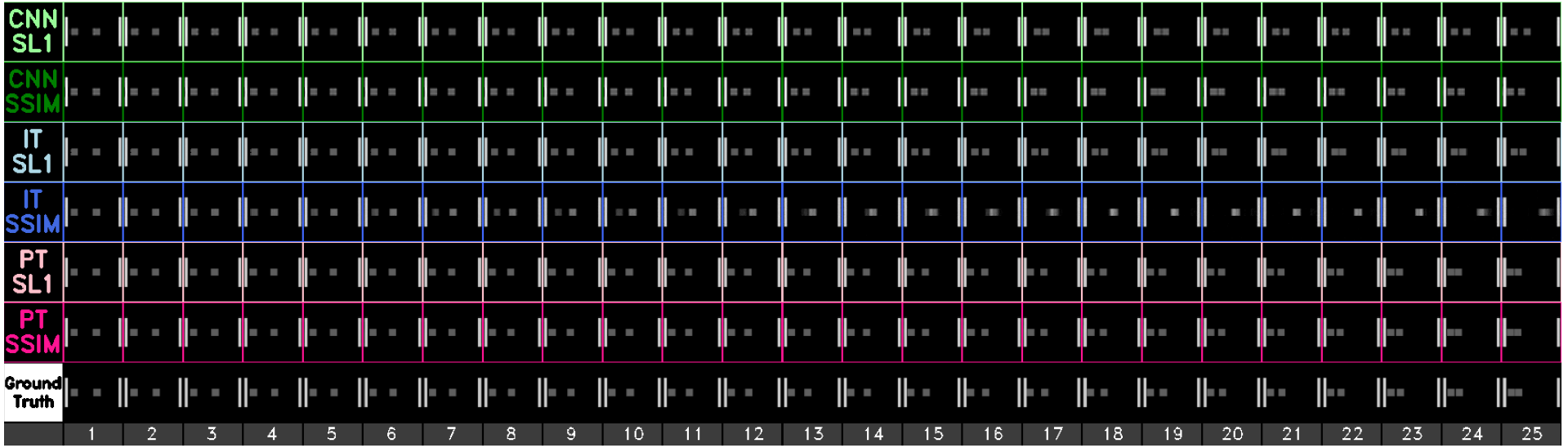
Figure 6.8: Comparison of the first 25 generated frames of each model ($m = 5$) visualised alongside the ground truth.


(e) Roller

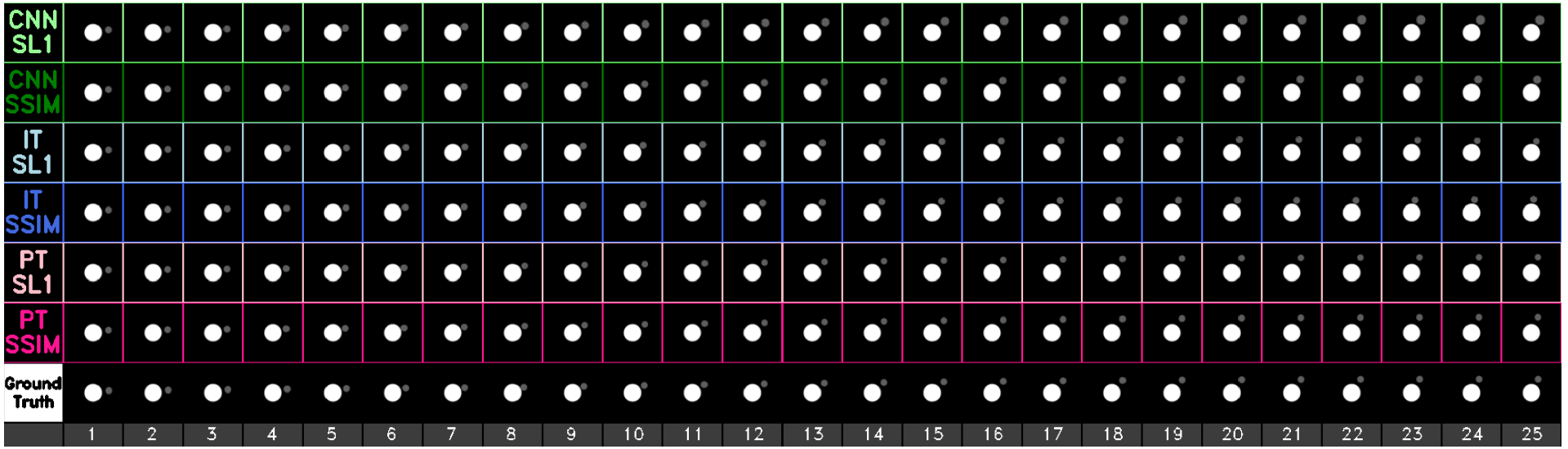



(f) Pendulum

Figure 6.8: Comparison of the first 25 generated frames of each model ($m = 5$) visualised alongside the ground truth.

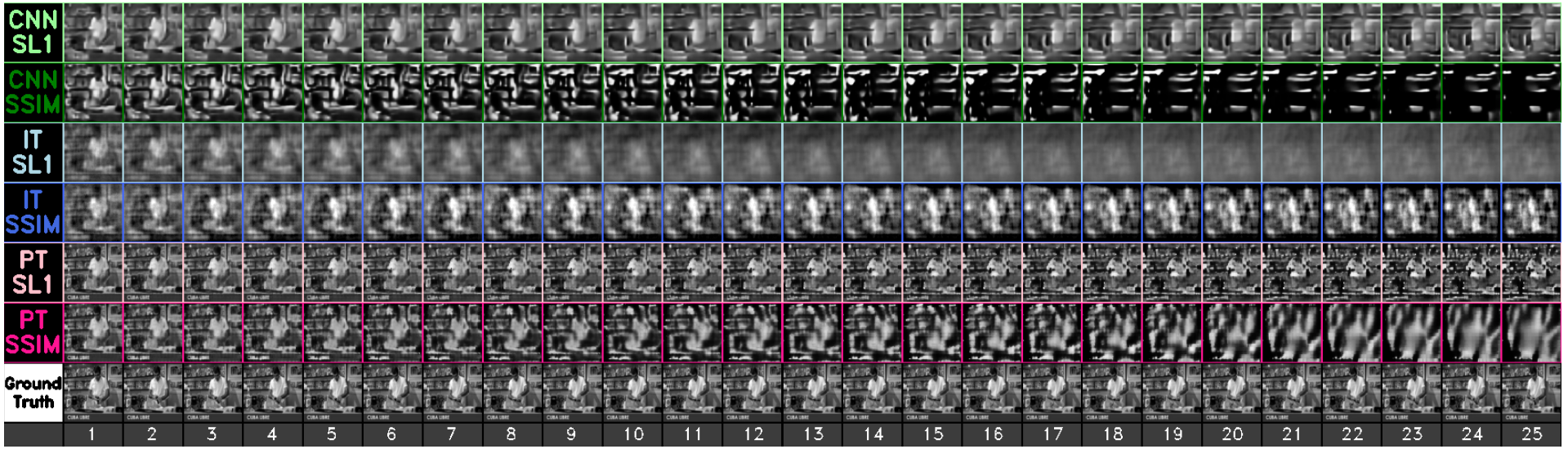



(g) Blocks




(h) Moon

Figure 6.8: Comparison of the first 25 generated frames of each model ($m = 5$) visualised alongside the ground truth.




(i) HMDB-51

Figure 6.8: Comparison of the first 25 generated frames of each model ($m = 5$) visualised alongside the ground truth.

**Self-Output By Models**

The patch transformer generates the most physically accurate long-term predictions (*i.e.* more closely obeys the laws underpinning the video) on the 3D bouncing, roller, pendulum, blocks, and moon data sets. The CNN model performs best on the 2D bouncing (Figure 6.8a) and MMNIST (Figure 6.8b) data sets. The image transformer is less physically accurate than both of the other models on all data sets. The CNN struggles to smooth out lower resolution graininess. There are occasional oddities with the CNN that imply it is overly relying on local spatial information *e.g.* a ball will sometimes accelerate off the screen in the roller data set. I argue that this is because the model has learned that a ball above a rail at a certain angle should be moving upwards. Although it is thought that pure CNN architectures struggle to properly model inter-frame variations in video sequences (Oprea et al., 2020), the CNN model's self-output videos challenge this assumption by demonstrating an understanding of gravity and collision physics (Figure 6.8a). The image transformer struggles to generalise to different shapes. This can be observed in MMNIST where digits degrade immediately within the first 5 frames (Figure 6.8b). Furthermore, the image transformer often accumulates black 'dead' pixels. Though the patch transformer appears to be the best model, it too demonstrates architecture specific artifacts (*e.g.* the resolution of the patches forming the outputs are visible in some examples from MMNIST).

**Self-Output By Loss**

When any of the three models are trained with SSIM loss, their predictions eventually begin to distort the otherwise constant background colour. This can be seen in the SSIM rows of Figure 6.8a as the background shifts from static white to grey. This phenomenon implies a gradual buildup of errors in the background which I argue is due to the insensitivity of SSIM function to changes in the background, and thus the model is not forced to carefully maintain the background. Such background distortion does not happen in SL1 outputs, indicating that the pixelwise-SL1 loss is particularly sensitive to these artifacts and prioritises minimising them. The self-output

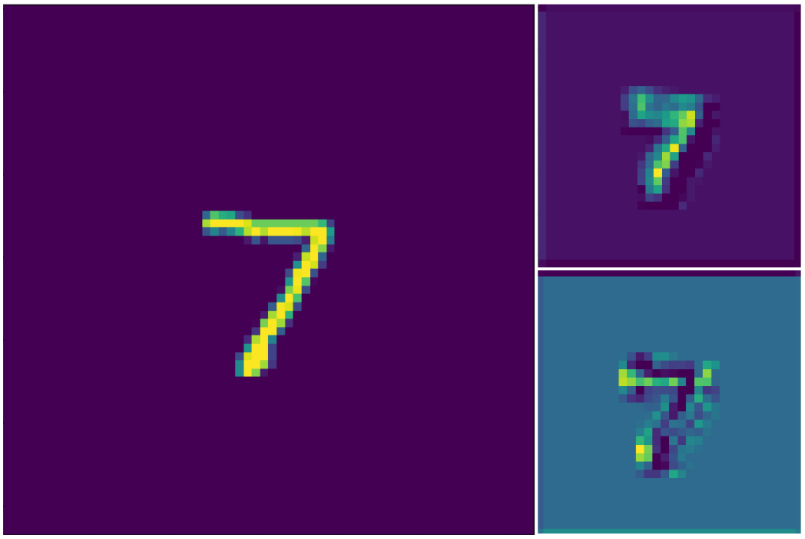
results challenge assumptions about the effects of different types of loss functions on generated videos. In their recent survey paper of video generation techniques, Oprea et al. (2020) argue that SL1 is a 'deterministic' pixelwise loss that learns to accommodate uncertainty by blurring its predictions. It is argued that this allows "plausible predictions in deterministic scenarios" (*i.e.* synthetic data sets), and yet struggle with more uncertain video data. However, the self-output videos demonstrate immediate and substantial blurring on the realistic MOCAP data for *both* the SL1 and SSIM trained models (Figure 6.8c). Furthermore, the simpler simulation data (MMNIST in Figure 6.8b, 2D bouncing in Figure 6.8a) shows almost no blurring for *either* loss function. These results imply that, in some cases, blurring can be caused by the difficulty of the data set and not inherent properties of the SL1 loss. Despite the 'synthetic' MMNIST data set scoring lower PSNR, $L1$ and SSIM metrics compared to the real-world MOCAP for *first-frame prediction* (discussed in Section 6.5.1), the superior *long-term prediction stability* of MMNIST compared to MOCAP stresses the importance of complementing frame-to-frame metrics and loss scores with qualitative long-term prediction comparisons.

### 6.5.3 Test-Task Performance

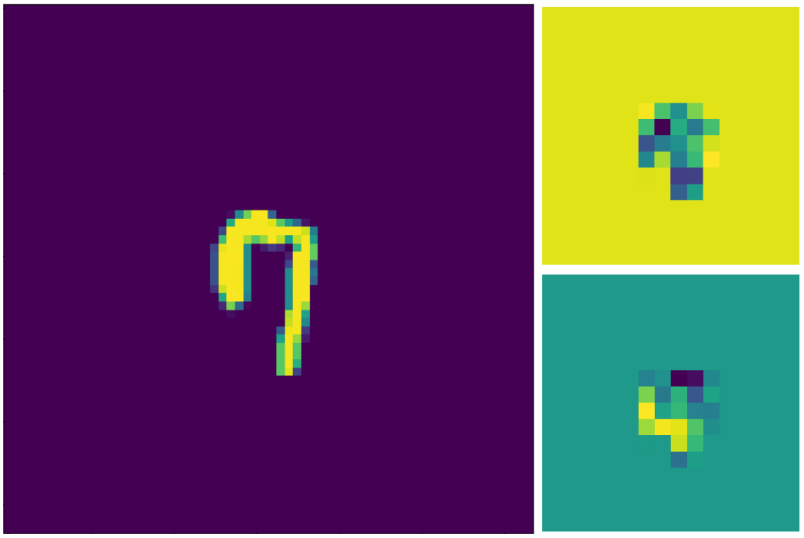
To make performance between the experiments more comparable, I normalise the values for all predictions (excluding MNIST classification) such that the data set has a standard deviation of 1. In the following subsections, I will discuss the results of the four variations of the downstream test-task experiments: random scores as lower bounds on performance, models trained on test-tasks *without* pretraining, linear probing for frozen models *with* pretraining, and models trained on test-tasks *with* pretraining (*i.e. finetuning*).

#### Random Lower Bound

The results for random and frozen models may initially appear too strong to be random baselines (*i.e.* each 3rd row of Tables 6.3 and 6.4; 4th row for MNIST only). However, I note that random



(a) CNN Random Conv2D Layer



(b) Patch Transformer Random Conv2D Layer

Figure 6.9: Visualisation demonstrating that randomly initiated pytorch Conv2D layers (with bias) can allow substantial image information to leak through. (a): A visualisation of a 64x64 grayscale image from the MNIST data set. (b): Visualisations of two different channels (16x16) from the outputs of the first layer in a randomly instantiated patch transformer/CNN model.

and frozen Conv2D PyTorch layers allow through a substantial amount of image information, even with trainable biases. This means substantial image information can still bleed through to the trainable linear probe layers as visualised in the MNIST examples in Figures 6.9a and 6.9b. I find this induces strong performance in the random frozen models, in particular scoring 98.58% accuracy on MNIST. This limits its use as a lower bound of task performance for the CNN and patch transformer. However, as the image transformer does not have Conv2D layers, it does not suffer such information leakage. This allows the random and frozen results from the image transformer to reasonably serve as an indicator for random task performance. For this reason, I specifically refer to the *frozen random score for the Image Transformer* when I discuss 'random' scores in the following sections (*i.e.* 3rd row of subsections of Tables 6.3 and 6.4; 4th row for MNIST).

**Unfrozen Without Pretraining + Non-Linear MLP**

The unfrozen model experiments with non-linear fully connected units serving as classifiers (see Figure 6.6c) are designed to show the potential of the models to learn visual laws directly from the task itself. The results of these experiments can be seen as the last grey row of each section in Tables 6.3 and 6.4. All three of the models can learn each task noticeably better than random (except the image transformer on 3D bouncing and blocks). The tasks with the best performance are 2D bounce counting, 2D gravity prediction, and roller gravity prediction. As these tasks all represent gravity working directly on 2D balls in freefall, I believe this style of simulation is the easiest to learn for these models. The 2D bounce counting task performance implies these models are capable of effectively tracking and counting collisions throughout the input frames. I note in particular the improved scores of the three following tasks. The bounce counting task for 2D bouncing scores 0.09786, which is a drastic improvement compared to its random score of 0.6457. The 2D bouncing gravity prediction task improves even further, scoring 0.02402 from a random score of 0.6442. The roller gravity prediction improves from a random score of 0.8393

| Model Details | | Baseline | CNN | | Image Transformer | | Patch Transformer | |
|---|---|---|---|---|---|---|---|---|
| Pretraining | Task | Task L1 | Modelling Loss | Task L1 | Modelling Loss | Task L1 | Modelling Loss | Task L1 |
| **Modelling Data Set = 2D Bounce** | **Task = Counting Bounces in 59 Frames** | | | | | | | |
| - | Constant Output | 0.6525 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.7077 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.5193 | - | 0.6457 | - | 0.6070 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.1007 | - | 0.2080 | - | 9.786e-2 |
| $m = 59$ SL1 | Frozen Model + Linear Probes | - | 5.003e-3 | 0.2830 | 1.207e-2 | 0.6569 | 1.557e-2 | 0.5532 |
| $m = 59$ SSIM | Frozen Model + Linear Probes | - | 1.797e-2 | 0.2943 | 5.057e-2 | 0.6563 | 2.328e-2 | 0.3839 |
| $m = 59$ SL1 | Unfrozen Model + Non-Linear MLP | - | 5.003e-3 | 0.1247 | 1.207e-2 | 0.2148 | 1.557e-2 | 0.1037 |
| $m = 59$ SSIM | Unfrozen Model + Non-Linear MLP | - | 1.797e-2 | 0.1165 | 5.057e-2 | 0.2737 | 2.328e-2 | 0.1008 |
| **Modelling Data Set = 2D Bounce** | **Task = Gravity, from 5 frames** | | | | | | | |
| - | Constant Output | 0.8676 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.8687 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.4272 | - | 0.6442 | - | 0.6076 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 1.269e-2 | - | 7.697e-2 | - | 2.402e-2 |
| $m = 5$ SL1 | Frozen Model + Linear Probes | - | 3.401e-3 | 0.1621 | 6.529e-3 | 0.7533 | 3.439e-3 | 0.4789 |
| $m = 5$ SSIM | Frozen Model + Linear Probes | - | 1.167e-2 | 0.1390 | 2.558e-2 | 0.7605 | 1.062e-2 | 0.3440 |
| $m = 5$ SL1 | Unfrozen Model + Non-Linear MLP | - | 3.401e-3 | 1.891e-2 | 6.529e-3 | 7.146e-2 | 3.439e-3 | 1.918e-2 |
| $m = 5$ SSIM | Unfrozen Model + Non-Linear MLP | - | 1.167e-2 | 1.774e-2 | 2.558e-2 | 5.960e-2 | 1.062e-2 | 1.385e-2 |
| **Modelling Data Set = 3D Bounce** | **Task = Counting Bounces in 99 frames** | | | | | | | |
| - | Constant Output | 0.6651 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.4255 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.3756 | - | 0.6420 | - | 0.4832 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.2662 | - | 0.6441 | - | 0.2713 |
| $m = 99$ SL1 | Frozen Model + Linear Probes | - | 3.905e-3 | 0.3200 | 2.398e-3 | 0.5982 | 7.494e-4 | 0.3279 |
| $m = 99$ SSIM | Frozen Model + Linear Probes | - | 1.439e-2 | 0.3198 | 1.112e-2 | 0.5721 | 3.713e-3 | 0.3414 |
| $m = 99$ SL1 | Unfrozen Model + Non-Linear MLP | - | 3.905e-3 | 0.2952 | 2.398e-3 | 0.4276 | 7.494e-4 | 0.2273 |
| $m = 99$ SSIM | Unfrozen Model + Non-Linear MLP | - | 1.439e-2 | 0.2711 | 1.112e-2 | 0.4897 | 3.713e-3 | 0.2555 |
| **Modelling Data Set = Roller** | **Task = Gravity, from 5 frames** | | | | | | | |
| - | Constant Output | 0.8645 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.8199 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.4305 | - | 0.8393 | - | 0.4699 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.1034 | - | 0.1679 | - | 7.220e-2 |
| $m = 5$ SL1 | Frozen Model + Linear Probes | - | 2.326e-3 | 0.1966 | 9.248e-4 | 0.8383 | 1.150e-4 | 0.3948 |
| $m = 5$ SSIM | Frozen Model + Linear Probes | - | 2.489e-3 | 0.1765 | 2.573e-3 | 0.8278 | 8.672e-5 | 0.2750 |
| $m = 5$ SL1 | Unfrozen Model + Non-Linear MLP | - | 2.326e-3 | 0.1016 | 9.248e-4 | 0.1318 | 1.150e-4 | 8.844e-2 |
| $m = 5$ SSIM | Unfrozen Model + Non-Linear MLP | - | 2.489e-3 | 0.1018 | 2.573e-3 | 0.1279 | 8.672e-5 | 7.790e-2 |
| **Modelling Data Set = Pendulum** | **Task = Gravity, from 5 frames** | | | | | | | |
| - | Constant Output | 0.8878 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.8903 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.7194 | - | 0.8914 | - | 0.7238 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.2837 | - | 0.3493 | - | 0.2266 |
| $m = 5$ SL1 | Frozen Model + Linear Probes | - | 5.564e-4 | 0.3770 | 2.934e-3 | 0.8982 | 6.726e-5 | 0.3834 |
| $m = 5$ SSIM | Frozen Model + Linear Probes | - | 9.868e-4 | 0.3757 | 5.886e-4 | 0.9021 | 8.816e-5 | 0.3731 |
| $m = 5$ SL1 | Unfrozen Model + Non-Linear MLP | - | 5.564e-4 | 0.3214 | 2.934e-3 | 0.3903 | 6.726e-5 | 0.2196 |
| $m = 5$ SSIM | Unfrozen Model + Non-Linear MLP | - | 9.868e-4 | 0.2898 | 5.886e-4 | 0.3358 | 8.816e-5 | 0.2173 |
| **Modelling Data Set = Blocks** | **Task = Mass Ratio, from 49 frames** | | | | | | | |
| - | Constant Output | 0.8880 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.8755 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.6372 | - | 0.8950 | - | 0.7296 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.5571 | - | 0.7836 | - | 0.4871 |
| $m = 49$ SL1 | Frozen Model + Linear Probes | - | 4.541e-3 | 0.6184 | 3.344e-4 | 0.8892 | 1.038e-4 | 0.7280 |
| $m = 49$ SSIM | Frozen Model + Linear Probes | - | 3.890e-3 | 0.6077 | 7.180e-4 | 0.8915 | 1.990e-4 | 0.6783 |
| $m = 49$ SL1 | Unfrozen Model + Non-Linear MLP | - | 4.541e-3 | 0.5799 | 3.344e-4 | 0.5516 | 1.038e-4 | 0.5296 |
| $m = 49$ SSIM | Unfrozen Model + Non-Linear MLP | - | 3.890e-3 | 0.5585 | 7.180e-4 | 0.5196 | 1.990e-4 | 0.5219 |

Table 6.3: Performance on test-tasks without vs. with modelling pretraining. Results are generated using checkpoints from the best validation epoch for pretraining and test-task.

| Model Details | | Baseline | CNN | | Image Transformer | | Patch Transformer | |
|---|---|---|---|---|---|---|---|---|
| Pretraining | Task | Task L1 | Modelling Loss | Task L1 | Modelling Loss | Task L1 | Modelling Loss | Task L1 |
| **Modelling Data Set = Moon** | | **Task = Mass, from 5 frames** | | | | | | |
| - | Constant Output | 0.8763 | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 0.8512 | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 0.7009 | - | 0.8751 | - | 0.8175 |
| None | Unfrozen Model + Non-Linear MLP | - | - | 0.3287 | - | 0.5829 | - | 0.3250 |
| $m = 5$ SL1 | Frozen Model + Linear Probes | - | 4.781e-4 | 0.382 | 5.656e-4 | 0.8136 | 9.834e-5 | 0.6579 |
| $m = 5$ SSIM | Frozen Model + Linear Probes | - | 5.199e-4 | 0.4909 | 2.751e-3 | 0.8268 | 9.945e-5 | 0.6259 |
| $m = 5$ SL1 | Unfrozen Model + Non-Linear MLP | - | 4.781e-4 | 0.3552 | 5.656e-4 | 0.3127 | 9.834e-5 | 0.3324 |
| $m = 5$ SSIM | Unfrozen Model + Non-Linear MLP | - | 5.199e-4 | 0.3822 | 2.751e-3 | 0.4246 | 9.945e-5 | 0.3528 |
| **Modelling Data Set = MMNIST** | | **Task = MNIST** | | | | | | |
| - | Most Common Class | 11.28% | - | - | - | - | - | - |
| - | Constant Output | 11.28% | - | - | - | - | - | - |
| - | Input Image + Linear Layer | 93.30% | - | - | - | - | - | - |
| None | Frozen Model + Linear Probes | - | - | 98.58% | - | 39.68% | - | 97.60% |
| None | Unfrozen Model + Non-Linear MLP | - | - | 99.44% | - | 99.00% | - | 99.52% |
| $m = 5$ SL1 | Frozen Model + Linear Probes | - | 2.828e-2 | 98.70% | 5.549e-2 | 81.16% | 2.732e-2 | 86.22% |
| $m = 5$ SSIM | Frozen Model + Linear Probes | - | 6.797e-2 | 98.66% | 1.616e-1 | 77.76% | 6.501e-2 | 87.24% |
| $m = 5$ SL1 | Unfrozen Model + Non-Linear MLP | - | 2.828e-2 | 99.16% | 5.549e-2 | 99.10% | 2.732e-2 | 99.56% |
| $m = 5$ SSIM | Unfrozen Model + Non-Linear MLP | - | 6.797e-2 | 99.42% | 1.616e-1 | 99.02% | 6.501e-2 | 99.56% |

Table 6.4: Performance on test-tasks without vs. with modelling pretraining. Results are generated using checkpoints from the best validation epoch for pretraining and test-task.

down to 0.0722. The patch transformer is strongest model on 2D bounce prediction and roller gravity prediction by a margin of ∼0.01-0.03. However, the CNN scores about half the patch transformer's loss on 2D bounce prediction. As MNIST classification is an easy vision data set, it is unsurprising that each model scores above 99%. The moon and pendulum prediction tasks appear to be the most difficult, improving from random scores of ∼0.89 to 0.325 and 0.2266 respectively. Though the patch transformer is most often the best scoring model, the CNN scores are almost as good. The CNN scores slightly better on the 3D bounce prediction task (∼0.01), and substantially better on 2D gravity prediction as previously mentioned. The image transformer is substantially worse than both the CNN and patch transformer on every non-MNIST task, which stresses the strengths of convolutional based models in learning these test-tasks *without* pretraining.

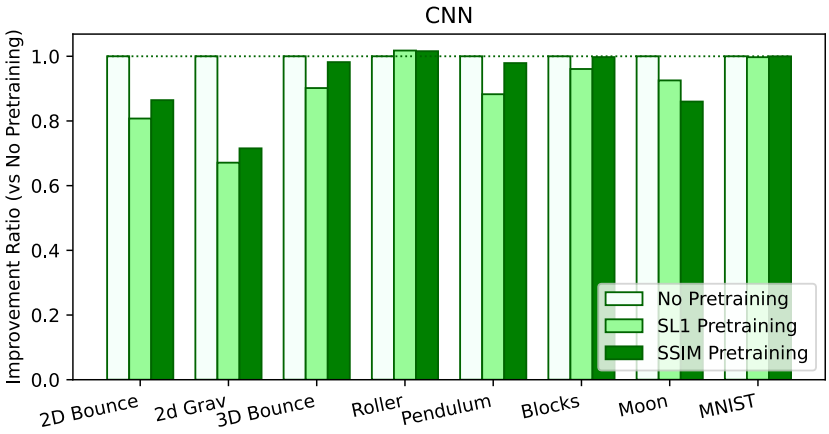
**Linear Probes**

The linear probing experiments (see Figure 6.6b) with **frozen** *and* **pretrained** models are designed to verify if modelling pretraining induces a transferable understanding of visual laws for downstream test-tasks (*i.e.* the top two white highlighted rows in each section of Tables 6.3 and 6.4). The CNN and patch transformer test-tasks on pretrained models consistently score better than random, yet do not perform near as well as the '*without* pretraining and unfrozen' experiments described in the previous subsection. The CNN model now outperforms the patch transformer on the the previous subsection's best performing experiments: 2D bounce prediction, 2D gravity prediction, and roller gravity prediction with ∼0.29 (CNN) vs. 0.553/0.3839 (PT), ∼0.15 (CNN) vs. 0.34/0.4789 (PT), ∼0.18 (CNN) vs. 0.395/0.2750 (PT) for each task respectively. However, I **cannot** directly conclude that all this information has been induced by pretraining because the leakage of image information through random and frozen Conv2D layers demonstrates that the convolution based models feed enough information to linear probes to achieve good scores. This highlights a unique difficulty in probing frozen convolutional models. Despite this information leakage, we can still verify the use of pretraining through linear probes on the CNN and patch transformer by comparing their probing scores with 'input image+linear layer' scores (see the second grey row of each section on Tables 6.3 and 6.4). This 'input image+linear layer' scenario shows how highly a trainable linear layer can score on the test-tasks with the images as input. Given that image information can leak through the frozen convolutions of the CNN and patch transformers and into the trainable linear probes, the 'input image+linear layer' experiment scores provide an upper bound to the performance the linear probes could have gained using *only* the leaked image information. I argue that the 'input image+linear layer' scores *are* upper bounds as *all* image information is provided, and images leaking through convolution layers will be at least partially distorted. As the probing scores are *still* much better than 'input image+linear layer' scores, this implies that modelling pretraining alone has encoded some understanding that is useful to the downstream tasks. Despite this, the linear probing models

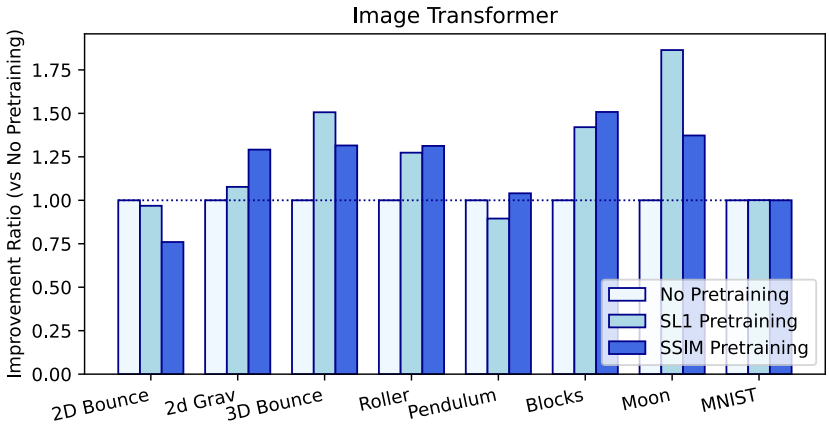
do not match 'unfrozen+without pretraining' performance, showing that although pretraining does induce some detectable understanding, it is still more useful to directly learn the task. As the image transformer does not suffer from random Conv2D leakage, it can more accurately demonstrate the induced understanding from pretraining. The image transformer is however the worst performing model on all tasks by a large margin, and its frozen probing tasks fail to exceed random scores on all but 3D bounce prediction and moon mass prediction (both which only improve by a 0.05). There are few noticeable differences between SL1 and SSIM pretraining for the CNN. SSIM is marginally better on 2D bouncing gravity prediction (by 0.03), and SL1 is 0.11 better for the moon task. The patch transformer however demonstrates improved performance for SSIM pretraining on all but the 3D bouncing bounce counting task, improving by between (0.03-0.15).

**Unfrozen With Pretraining + Non-Linear MLP (Finetuning)**

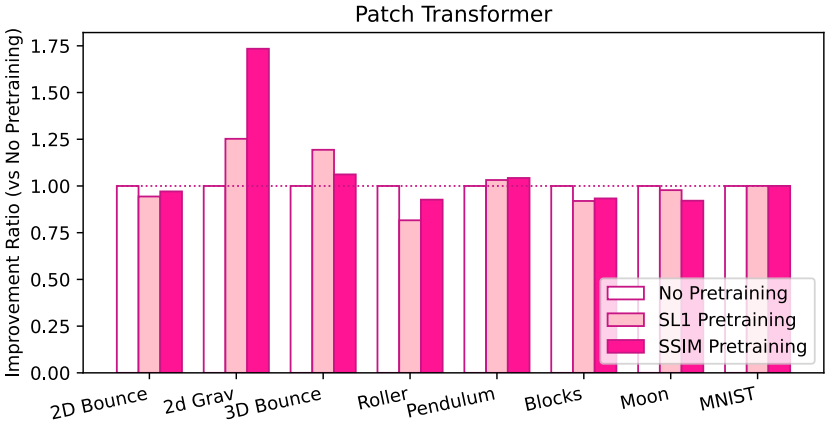
My 'finetuning' experiments are designed to verify if visual modelling pretraining can give an advantageous starting point for learning downstream vision tasks (vs. *without* pretraining). The results of these experiments are on the last two rows for each section on Tables 6.3 and 6.4. The results of directly training on tasks *with* versus *without* pretraining are visualised in Figure 6.10. The CNN noticeably degrades in performance from both SL1 and SSIM pretraining in 3 of 8 tasks. The CNN degrades most substantially on the 2D bouncing gravity and 2D bounce counting tasks, with an improvement ratio of  0.8 and  0.7 respectively (Figure 6.10a). Overall, the CNN fails to benefit from pretraining in its performance on the downstream tasks. The image transformer however improves substantially on 5 of the 8 tasks when finetuning a modelling-pretrained model (Figure 6.10b), with improvement ratios varying between 1.3 to as high as 1.8. Visual modelling pretraining for the patch transformer is almost as good or substantially better compared to no pretraining for all tasks but roller gravity prediction (Figure 6.10c). Interestingly, the image and patch transformer models show the same preference for pretraining loss for each task. SSIM




(a) CNN




(b) Image Transformer




(c) Patch Transformer

Figure 6.10: The improvement ratio in scores (losses) for each task when pretrained on visual modelling (vs. *without* pretraining) for each model architecture.

pretraining consistently gives higher scores compared to SL1 for 2D bouncing, 2D gravity, roller, pendulum, and blocks data sets. Conversely, SL1 outperforms SSIM for the 3D bouncing and moon data sets. When we consider the best performing model for each task (Figure 6.2), *i.e.* either the image or patch transformer, pretraining on modelling tasks is around as good, or considerably better for downstream task performance for 7 of my 8 tasks (improving by a ratio of 1.25 to 1.8) demonstrating the potential of the generative pretraining visual modelling paradigm.

### 6.5.4 General Discussion

I focus specifically on *pairing* the visual pretraining datasets with *quantifiable* and *observable* test-tasks and analysis, allowing me to strongly argue for the benefits of pretraining that the experiments demonstrate. The finetuning experiments show that visual modelling pretraining can yield substantially improved performance on test-tasks while not impacting performance on the majority of other tasks that do not benefit from pretraining. This shows that there can be both little risk and large potential gain in generative pretraining for downstream vision tasks. The image and patch transformer architectures benefit most substantially from visual modelling pretraining. However, the CNN did not improve with pretraining on any of the test-tasks. This would imply that there is some inherent property in a transformer that the CNN lacks (*e.g.* self-attention) that can successfully facilitate such pretraining. When finetuning from pretraining, the image transformer scores the highest on the blocks and moon test-tasks, with the patch transformer performing best on the other 6 tasks. When considered alongside the first-frame prediction and self-output experiments, the patch transformer performs best with visual modelling pretraining, and the image transformer benefits most substantially. Though it would be thematically fitting that a convolutional adaption of a transformer architecture would best serve the overlap of vision and language modelling, this cannot be directly concluded this from the results. Future research focusing more specifically on architecture would help verify the findings. Further improving the size and scale of visual modelling pretraining is a promising

path to realise even greater performance increase. Chen et al. (2020) demonstrate that by leveraging very large computational resources (a model comparable to GPT-2 *i.e.* 48 layers and ∼1.4B parameters), unsupervised image pretraining leads to state-of-the-art downstream ImageNet and CIFAR-101 performance *that increases with model scale*. This trend of increasing downstream task performance as model and data scale increases highlights how promising both visual modelling pretraining and the data sets that can facilitate it can be to the future of downstream vision tasks.

## 6.6 Limitations

In this section, I discuss the main limitations of this study. I highlight areas which I believe are promising for future research.

### 6.6.1 Scale:

My visual modelling approach does not match the sheer scale of modern language modelling. Modern language models can train on a huge amount of data due to the comparative ease of collecting, storing, and tokenising raw language. Such language models are often trained with substantial computational resources. By directly training on and outputting raw images, which are much larger representations than language tokens, I can't yet approach the scale of large pretrained modern language models with the resources available to us.

### 6.6.2 Data Set Complexity:

Larger and more complicated visual data sets are unexplored. Though I do verify the pretraining on HMDB-51 and MOCAP, the gap between these results and the longer term predictive power of simpler data sets highlight the need for a more measured and careful probing approach to simpler visual tasks first.

### 6.6.3 More Nuanced Predictive Training Strategy:

Though I parallel language modelling, I only predict the final frame of a sequence, and do not parallel more intricate language modelling training strategies, *e.g.* bidirectional token prediction for words in the middle of a sentence. Although more nuanced training methods for video has been explored on techniques from several years ago (Ranzato et al., 2014), and more recently for generative pretraining with *images* (Chen et al., 2020), I am able to present strong pretraining results without using such nuanced approaches in this *video-based* study. Regardless, this is an interesting line of research for future work.

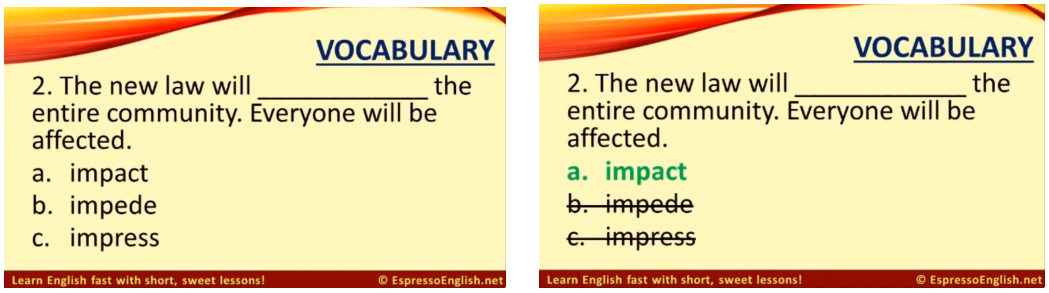
## 6.7 Going Beyond Next Frame Prediction: A Reformulation, Long Document Approach

The benchmark models above show that visual modelling is a feasible paradigm for learning visual features. In the next section, I explore how the ideas from the previous chapters apply to this paradigm.

### 6.7.1 Task Reformulation for Vision

While in the benchmark the test tasks are mostly regression, there are other types of tasks which can be constructed using the same datasets. For instance, sequence-to-sequence tasks such as segmenting the foreground objects from the background, summarizing key moments from the sequence, as well as classification tasks such as predicting whether the moon collides with the planet are all possible.

I suggest that just as question answering or language modelling can be used as "supertask" in NLP for task reformulation, visual modelling can perform a similar role for computer vision. A model pretrained on extensive unsupervised video data should acquire the capability for complex




(a) Input frame of the video containing the question.

(b) Subsequent output frame of the video containing the answer.

Figure 6.11: Example of how an NLP Question Answering task can be reframed as visual modelling. This type of task occurs naturally in popular YouTube videos (Video source: https://www.youtube.com/watch?v=QQn-3HtNG-A).

visual reasoning. This is akin to how models can learn complex linguistic phenomena such as coreference resolution from the language modelling task alone. A well-performing visual model should, when appropriately prompted, be capable of tasks like object classification and person tracking.

Furthermore, I argue that visual modelling can serve as a unifying "supertask" that bridges both vision and NLP. Videos often contain textual elements, as illustrated in Figure 6.11 which showcases a popular YouTube video format featuring a sequence of English Grammar quiz questions followed by their answers. A robust visual model capable of predicting subsequent frames in a video would inherently need to possess knowledge relevant to NLP tasks. For the example in Figure 6.11 this would require question answering abilities.

Traditional multimodal models that integrate NLP and vision typically employ separate encoders for each modality. These usually consist of Convolutional Neural Networks (CNNs) for visual inputs and transformer-based architectures for text inputs. The extracted features are then aligned and fused into a shared feature space, which is subsequently utilized for various downstream tasks.

In contrast, I imagine a paradigm where all NLP and vision tasks are subsumed under the umbrella of "visual modeling". We could create a powerful "visual model" that has been pretrained on a vast, unsupervised dataset comprising a diverse sample of videos. Much like how language modelling serves as a foundation for understanding a broad spectrum of linguistic concepts, this visual model would be expected to induce a comprehensive understanding of both visual and textual phenomena. Such a model would not only revolutionize task-specific applications but also pave the way for more integrated, holistic approaches to machine learning challenges.

While I leave the creation of such a model as future work (see 7.2.4 for more details), in the next section I test this idea on a small scale — showing it is indeed possible to reformulate NLP tasks as visual modelling.

**Amazon Product Review Classification as Visual Modelling**


Figure 6.12: Example of format for Amazon product reviews as a visual modelling task. Each box is a frame of the video (left to right). The final frame is dark in this example, as this has a low star rating.

To test this idea, I construct a simple task based on the Amazon product reviews dataset (Keung et al., 2020). This dataset consists of user-submitted reviews regarding Amazon products, with the task being to predict the 1-5 star rating associated with the review. For this experiment, I simplify the task to binary 1 vs 5 star classification.

To transform this into a visual modelling task, I employ a simple encoding. I take the first 32 characters of the review title (left padding if the title is shorter, truncating if the title is longer. Most Amazon review titles are shorter than this). This string is then converted into a video, one letter per frame in a monospace font (see Figure 6.12). Finally, I multiply the star rating by

51 to convert the labels into a 0-255 grayscale colour - black if the rating is 1 star, white if the rating is 5-star. I add a final blank frame where the background is this colour. This allows for easy conversion of the model output back into the predicted label.

| Model | Accuracy |
|---|---|
| BERT | 0.96 |
| CNN | 0.82 |
| Image Transformer | 0.71 |
| Patch Transformer | 0.75 |

Table 6.5: Performance of visual models on the Amazon product review sentiment analysis task.

I use the models and methodology from the previous sections, using the first 32 frames as input and training models to predict the 1 output frame. Table 6.5 shows the results of this experiment along with comparison against a BERT model (Devlin et al., 2019) on this same task. While the model doesn't perform as strongly as BERT which has been pretrained on a large dataset of text, the best visual model is able to score 82%, suggesting that reformulating NLP as visual modelling, even in this simple setting, can be very successful. With models trained on a much larger and more varied dataset of examples, we could expect to see strong performance across a range of tasks.

### 6.7.2 Task Decomposition

Long documents in this domain are arguably even more of a problem than in NLP due to the information density of video. In the previous experiments, this limited the scope of the work to short clips with 100 frames or fewer. For real world data such as films, videos of talks, and other similar content this means tens of thousands of frames per video. When coupled with modern, large frame video resolutions such as 4K, this poses a significant challenge.

Similarly to how task decomposition can be used to reduce the complexity of long document NLP problems, a similar strategy can be employed for long videos.

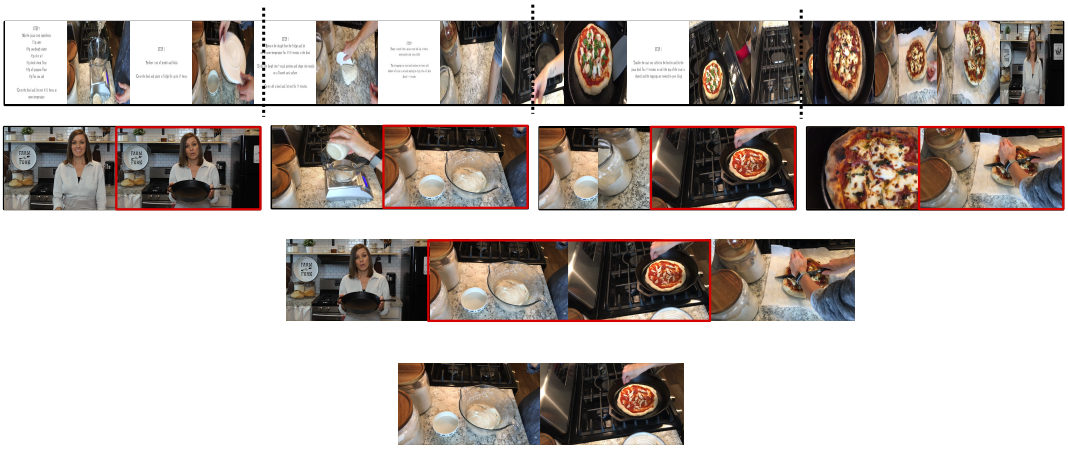


Figure 6.13: Example of how a video can be summarized using a task-decomposition approach. (Video source: https://www.youtube.com/watch?v=rZ4B04eTy7M)

Figure 6.13 shows an example of how such an approach could be applied to video summarization. As in the method described in chapter 5, the input is first split into chunks. Each chunk can then be summarized in turn before being recombined into a final summary.

Video classification can be performed in the same way as chapter 5, with each chunk predicting the output, and their outputs being merged via majority vote.

Segmenting a video into scenes could be performed with a simplified version of the approach used for style change detection in chapter 5. The lowest levels could detect scene changes within their chunks then higher levels can recontextualise this based on surrounding chunks. As we only want to split the video into scenes, we only need to consider if the first or last scene of a chunk is the same as the neighbours, vastly simplifying the problem. Problems such as tracking people can be performed using a similar approach, tracking within chunks then merging at higher levels of the hierarchy.

I leave the exploration of such an approach as future work.

## Epilogue

In this chapter, I introduce the visual modelling task, and explore its effectiveness on a dataset of simple physics simulations. I find that models are able to learn this task well, as the knowledge learnt is at least useful for some downstream tasks. I then explore how visual modelling could be used as a "supertask" in a multimodel model, demonstrating that it can be used to solve NLP problems.

In the final chapter, I draw this thesis to a conclusion by summarizing the contributions made and suggesting interesting future work.

# Chapter 7

# Concluding Remarks

In all my previous chapters where I made contributions, I extensively discussed and concluded the findings and outcomes. In addition to these established conclusions, this final chapter will summarize the conclusions of this thesis as a whole, as well as discussing its limitations and implications for future work.

In this thesis, I addressed two significant challenges in natural language processing: the limitations of multitask learning, and the processing of long documents. I have shown weaknesses in how robust multitask models are to paraphrasing their task-prompt inputs, as well as proposing solutions for this. Additionally, my proposed multitask long document benchmark - MuLD, and my task-decomposition model have provided a framework for handling the problems posed by very long documents in NLP. The work on Visual Modelling suggests how these ideas can be expanded into the vision domain and ultimately the possibility of creating multimodel models.

While there is more work to be done in this field, I hope my findings will inspire further research in this challenging area of machine learning.

## 7.1 Contributions

My literature review in **Chapter 2** provides an extensive overview of areas shared across my contribution chapters: Fundamental NLP tasks, long document datasets and models, as well as multitask model architectures and task reformulation.

**Chapter 3** explores the robustness of models trained on the decaNLP multitask benchmark. These models used task reformulation to cast 10 NLP tasks in terms of question answering. I gather a new dataset of paraphrased multitask prompts, PQ-decaNLP, which created via crowd-workers. This diagnostic dataset is crucially annotated with the paraphrase phenomena that were used to transform it from the original decaNLP question the models were trained upon. This allows me to measure how decaNLP models fail under paraphrasing and importantly what kind of paraphrasing. I then test methods for improving the robustness of the models via both directly finetuning on my PQ-decaNLP dataset, and by exploiting the multitask nature of the model, adding a paraphrase detection/generation task. I show that while directly training on the paraphrased questions improves performance, knowledge from additional paraphrasing tasks does not transfer to increased robustness to paraphrasing.

I then explore the creation of multitask models which can operate on long documents. In **Chapter 4**, I create a new benchmark which spans a wide variety of NLP tasks and importantly uses inputs with lengths much greater than previous work. I take commonly used datasets and augment, expand or replicate their methodology in order to create a benchmark where every document is over 10,000 tokens in length. I devise a series of baselines for each task using both a standard transformer and efficient transformer model. I show that models with increased context length are better able to solve the tasks presented, suggesting these tasks require models to handle long contexts.

**Chapter 5** improves on these baselines by building a model based upon task-decomposition.

By decomposing a large task into several smaller sub-tasks, we can solve a wide variety of common NLP tasks, without any task-specific logic outside the model. I show how a variety of NLP problems can be reformulated to work within this framework, and find that the model is able to achieve similar performance to the baseline model while adopting a task-agnostic approach, and outperforms it on 2 of the tasks: Style Change Detection and Translation. Due to the ability to inspect intermediate outputs, I am able to analyse why the model fails under different circumstances. Crucially, this model performs the same procedure irrespective of the task, allowing for multitask learning and the addition of new tasks simply by formatting the data.

**Chapter 6** explores a visual parallel of language modelling - visual modelling. I create a dataset of simple physics simulation videos and explore how a variety of models can predict next frames if provided with a partial sequence. I pair each simulation with a test task to investigate what knowledge pretraining on visual modelling induces, finding that for at least some of the tasks the knowledge is helpful for solving downstream tasks. Finally, I explore how just as question answering and language modelling can be used as super-tasks for NLP, visual modelling can act as a supertask uniting NLP and vision. I convert a sentiment classification task into a video format and show that a visual model can solve this task with 82% accuracy. I additionally show how the task-decomposition model introduced in Chapter 5 could be extended into this modality.

## 7.2 Limitations & Future Work

Although limitations and future work have been discussed throughout each of the preceding chapters, here I discuss them in more detail and in context of the entire thesis.

### 7.2.1 Chapter 3: Bigger and Better Models

Since the work on Chapter 3 was conducted, there have been many developments in multi-task learning, significantly in the use of powerful language models (such as GPT3, ChatGPT) which show impressive zero-shot capability across many NLP tasks. Rather than specific prompt questions, users can interact with these models in a more naturalistic way. However, like the task-prompt question models examined in this thesis, there are still failures. Future work should extend my methodology, investigating how robust these models are to different kinds of paraphrasing of the prompt. Additionally, many of these models can be shown a new task by providing examples in the prompt. Variations in the number, order, and variety of these examples would provide interesting insights into how these models perform.

### 7.2.2 Chapter 4: More Tasks

Future work should also take the ideas from Chapter 4 of extending and filtering datasets to increase their length, and apply it to more NLP tasks.

As computational hardware continues to evolve, becoming more powerful and efficient, it's crucial that our benchmarks evolve in tandem to remain challenging. MuLD, designed to provide a challenging task for existing long document models, should not become obsolete as hardware capabilities increase. Therefore, the length of the documents in the benchmark could be progressively increased to ensure that they continue to challenge the state-of-the-art models. This will not only push the boundaries of what current models can do, but also drive innovation in developing even more efficient algorithms and architectures.

An intriguing possibility is that many search problems could be reframed as information extraction tasks from long documents. These 'long documents' could be artificially constructed by concatenating a database of shorter documents. For example:

**Legal Search** Instead of searching through multiple legal documents to find relevant case law or statutes, a single concatenated 'super-document' could be created. Information extraction techniques could then be applied to find the relevant sections.

**Scientific Research** Researchers often need to sift through multiple papers to gather information on a specific topic. Concatenating these papers into a single document could allow for more efficient information extraction via question answering or summarization.

**News Aggregation** In the context of news or social media, one could concatenate various articles or posts related to a specific event to form a single document. Information extraction could then be used to summarize the event or identify key viewpoints.

By approaching these search problems as information extraction tasks in a single document, we could leverage the power of long-document handling capabilities of efficient long-document models to improve performance on these tasks. However, it's important to consider whether ever constructing longer documents is the correct paradigm. Constructing longer documents may lead to long 'super-documents' with contradictory information - a problem future models will have to solve. Length increases must also be meaningful, and it may be more fruitful to consider other paradigms, such as retrieval for certain kinds of data.

### 7.2.3 Chapter 5: Problems with single-pass models

In Chapter 5, I discuss the limitation of my long document model which processes the document in a single pass, only looking at each chunk once. This causes problems for some tasks, namely multi-hop question answering, where it is not always clear whether a section of the document is useful until a later section is read.

To overcome this limitation, we need a model which can arbitrarily decide which parts of the documents to read and re-read. This approach would allow the model to make decisions about which parts of the document to consider and how to navigate between them based on the content of the document it has already read. This mimics how humans solve long document problems by first evaluating the context and making decisions about what information is needed and where in the document it is likely to be found.

Inspired by how webGPT (Nakano et al., 2021) browses the web, we could devise a set of actions the model can take at each stage:

- Forward/Backward - Moves to the next or previous chunk in the document
- Store - Copies a fragment of text from the current chunk into a temporary scratch area which can be recalled
- Find - Finds the next instance of a specific term
- Output - Outputs directly. Used for the final answer (in the case of classification), or outputting continuously as in translation.

The model can be trained to replicate human behaviour at each stage. This approach would be a more flexible and adaptable solution which would improve performance not just at multihop question answering, but a range of other complex tasks. Additionally, it removes many of the assumptions made when forming the data, instead relying on replicating human approaches to the tasks. It does however require recording the sequence of actions needed to solve a task, which may require an extensive human labelling effort.

The style of approach needed here highlights the need to broaden our focus beyond machine learning and consider how difficult problems such a planning in classical Artificial Intelligence

(AI). The extent to which planning abilities emerge from large language models is an interesting area of future work.

### 7.2.4 Visual Modelling: A Multimodel "Supertask"

In chapter 6, I introduced the concept of visual modelling and showed that it can serve as a useful pretraining task for downstream video tasks. I also introduced the idea of encoding text within videos, as a possible way of using visual modelling as a "supertask" uniting vision and NLP.

Although in chapter 6, I explore a very simple reformulation (one letter per frame) on an easy NLP task, I show that it is indeed possible to solve NLP tasks via visual modelling. With a sufficiently capable model and enough resources it should be possible to pretrain a model on a large diverse video dataset (such as YouTube). Different ways of converting text into a video should also be experimented with (single words, sentences and even entire paragraphs per frame).

In this envisioned paradigm, the "visual model" would be a unified architecture capable of handling both visual and textual data seamlessly. Instead of having separate encoders for each modality, this model would employ a more integrated approach, using a single encoder for all modalities, each of which being reformulated as visual modelling. This would allow the model to learn cross-modal relationships more effectively, thereby enhancing its performance on a wide range of tasks. This would enable tasks which mix images, text, and video to be solved without separate encoders.

The potential advantages of this approach are manifold. First, a unified model would be more efficient in terms of computational resources and memory, as it would eliminate the need for separate encoders and the subsequent alignment steps. Second, it would enable more effective transfer learning, as the model would be pretrained on a rich, multimodal dataset. This would

allow it to generalize better to new tasks, whether they are purely visual, purely textual, or a combination of both.

Moreover, the concept of "visual modeling" as a "supertask" could extend beyond the scope of individual tasks to serve as a foundational technology for various applications. For example, in autonomous driving, a visual model could not only recognize and track objects but also understand road signs and even human gestures. In healthcare, it could assist in medical image analysis while also being capable of reading and interpreting medical texts.

However, there are challenges to be addressed. One of the primary concerns would be the complexity of training such a unified model. The training data would need to be carefully curated to ensure a balanced representation of both visual and textual elements.

The idea of visual modeling as a "supertask" offers an exciting avenue for future research. It promises a more integrated, efficient, and powerful approach to machine learning, capable of understanding the world through both its visual and textual aspects. By moving towards this unified framework, we can potentially unlock new capabilities and applications that are currently beyond the reach of models specialized in either vision or NLP alone.

## Epilogue

During the undertaking of this work, the field of NLP has experienced a profound transformation. The 2 key problems I contribute towards: multitask learning and long document NLP, have also seen major developments. Efficient transformer models have pushed the lengths of documents we are able to process, and prompting has become the de-facto standard for task reformulation using multitask models.

Yet many issues still remain.

While prompting is now a widely used paradigm, analysis of how these new models understand prompts and the ways they fail under paraphrasing is little explored. I hope the techniques and analysis explored in this thesis can be applied to this new paradigm. Understanding how these multitask models work and fail is vital for their integration into everyday life.

We have not yet seen the widespread adoption of long document models[1]. MetaAI's recently released open-source large language model, for example, only has a context length of 4096 tokens (Touvron et al., 2023). I hope that benchmarks and methods such as the ones described in this work help make processing novels, reports and even theses such as this possible.

Finally, as our ability to synthesise images increases, particularly with the advent of diffusion models (Croitoru et al., 2023) and other powerful techniques, I hope to see the same transformation in computer vision as prompt-based LLMs have driven in NLP. Ultimately, I believe uniting these (and other) modalities under a single model will revolutionise the field, continuing the trend of more powerful, universal, and ultimately more useful models.

[1]Things move very fast in NLP. This summary was written in the summer of 2023. By final submission in May 2024, models with context lengths exceeding 100,000 tokens now exist. The need to evaluate them properly continues.

# Bibliography

Akiba, Y. et al. (2004). 'Overview of the IWSLT evaluation campaign'. In: *Proceedings of the First International Workshop on Spoken Language Translation: Evaluation Campaign* (cit. on p. 13).

Alqahtani, S., A. Mishra and M. Diab (2020). 'A multitask learning approach for diacritic restoration'. In: *arXiv preprint arXiv:2006.04016* (cit. on p. 38).

Altheneyan, A. S. and M. E. B. Menai (2020). 'Automatic plagiarism detection in obfuscated text'. In: *Pattern Analysis and Applications* 23, pp. 1627–1650 (cit. on p. 43).

Amersfoort, J. R. van et al. (2017). 'Transformation-Based Models of Video Sequences'. In: *ArXiv* abs/1701.08435 (cit. on p. 116).

Bahdanau, D., K. Cho and Y. Bengio (2014a). 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: (cit. on p. 21).

— (2014b). 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: *CoRR* abs/1409.0473. URL: https://api.semanticscholar.org/CorpusID:11212020 (cit. on p. 20).

Banerjee, S. and A. Lavie (2005). 'METEOR: An automatic metric for MT evaluation with improved correlation with human judgments'. In: *Proceedings of the acl workshop on intrinsic*

*and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72 (cit. on p. 14).

Barrón-Cedeño, A. et al. (Dec. 2013). 'Plagiarism Meets Paraphrasing: Insights for the Next Generation in Automatic Plagiarism Detection'. In: *Computational Linguistics* 39.4, pp. 917–947 (cit. on pp. 45, 58).

Barto, A. G. and S. Mahadevan (2003). 'Recent advances in hierarchical reinforcement learning'. In: *Discrete event dynamic systems* 13.1-2, pp. 41–77 (cit. on p. 32).

Beltagy, I., M. E. Peters and A. Cohan (2020). 'Longformer: The long-document transformer'. In: *arXiv preprint arXiv:2004.05150* (cit. on pp. 27, 30, 70, 80).

Bengio, Y. et al. (2000). 'A Neural Probabilistic Language Model'. In: *J. Mach. Learn. Res.* 3, pp. 1137–1155 (cit. on p. 116).

Bezenac, E. de, A. Pajot and P. Gallinari (2018). 'Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge'. In: *ICLR* (cit. on p. 117).

Biderman, S., K. Bicheno and L. Gao (2022). *Datasheet for the Pile* (cit. on p. 98).

Bird, S. and E. Loper (July 2004). 'NLTK: The Natural Language Toolkit'. In: *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Barcelona, Spain: Association for Computational Linguistics, pp. 214–217. URL: https://aclanthology.org/P04-3031 (cit. on p. 94).

Bordes, A., S. Chopra and J. Weston (2014). 'Question Answering with Subgraph Embeddings'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics (cit. on p. 57).

Bos, J. and K. Markert (2005). 'Recognising textual entailment with logical inference'. In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 628–635 (cit. on p. 11).

Bowman, S. R. et al. (2015). 'A large annotated corpus for learning natural language inference'. In: *arXiv preprint arXiv:1508.05326* (cit. on p. 11).

Brown, P. F., J. Cocke et al. (1990). 'A statistical approach to machine translation'. In: *Computational linguistics* 16.2, pp. 79–85 (cit. on p. 12).

Brown, T., B. Mann, N. Ryder et al. (2020a). 'Language Models are Few-Shot Learners'. In: *NeurIPS*. Vol. 33, pp. 1877–1901 (cit. on p. 111).

Brown, T., B. Mann, N. Ryder et al. (2020b). 'Language models are few-shot learners'. In: *Advances in neural information processing systems* 33, pp. 1877–1901 (cit. on p. 98).

Buck, C. et al. (May 2018). 'Ask the Right Questions: Active Question Reformulation with Reinforcement Learning'. In: *Sixth International Conference on Learning Representations (ICLR)*. Vancouver, Canada. URL: https://openreview.net/forum?id=S1CChZ-CZ (cit. on p. 44).

Butcher, J. C. (1996). 'A history of Runge-Kutta methods'. In: *Applied numerical mathematics* 20.3, pp. 247–260 (cit. on p. 122).

Carion, N. et al. (2020). 'End-to-End Object Detection with Transformers'. In: *ECCV*, pp. 213–229 (cit. on p. 111).

Castrejon, L., N. Ballas and A. Courville (2019). 'Improved Conditional VRNNs for Video Prediction'. In: *ICCV* (cit. on p. 111).

Chen, M. et al. (2020). 'Generative Pretraining From Pixels'. In: *ICML* (cit. on pp. 111, 112, 115, 116, 151, 152).

Cheung, M. L. L. (2009). 'Merging corpus linguistics and collaborative knowledge construction'. PhD thesis. University of Birmingham (cit. on p. 44).

Child, R. et al. (2019). 'Generating long sequences with sparse transformers'. In: *arXiv preprint arXiv:1904.10509* (cit. on p. 31).

Cho, K. et al. (2014). 'Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. URL: http://dx.doi.org/10.3115/v1/D14-1179 (cit. on p. 21).

Cohan, A. et al. (June 2018). 'A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 615–621. URL: https://www.aclweb.org/anthology/N18-2097 (cit. on pp. 28, 78).

Collobert, R. and J. Weston (2008). 'A unified architecture for natural language processing: deep neural networks with multitask learning'. In: *ICML '08* (cit. on p. 33).

Cooley, J. W. and J. W. Tukey (1965). 'An algorithm for the machine calculation of complex Fourier series'. In: *Mathematics of computation* 19.90, pp. 297–301 (cit. on p. 31).

Cooper, R. et al. (1996). *Using the framework*. Tech. rep. Technical Report LRE 62-051 D-16, The FraCaS Consortium (cit. on p. 11).

Crawshaw, M. (2020). 'Multi-task learning with deep neural networks: A survey'. In: *arXiv preprint arXiv:2009.09796* (cit. on p. 35).

Cristianini, N. and J. Shawe-Taylor (2000). 'An Introduction to Support Vector Machines and Other Kernel-based Learning Methods'. In: URL: https://api.semanticscholar.org/CorpusID:60486887 (cit. on p. 18).

Croitoru, F.-A. et al. (2023). 'Diffusion Models in Vision: A Survey'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.9, pp. 10850–10869 (cit. on p. 166).

Dai, J. et al. (2017). 'Deformable Convolutional Networks'. In: *ICCV*, pp. 764–773 (cit. on p. 116).

Darapaneni, N. et al. (2021). 'Building a Question and Answer System for News Domain'. In: *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pp. 78–83 (cit. on p. 20).

Dayan, P. and G. E. Hinton (1992). 'Feudal reinforcement learning'. In: *Advances in neural information processing systems* 5 (cit. on pp. 31, 32, 85).

Deng, J., W. Dong et al. (2009). 'Imagenet: A large-scale hierarchical image database'. In: *CVPR* (cit. on p. 115).

Deng, L., G. Hinton and B. Kingsbury (May 2013). 'New types of deep neural network learning for speech recognition and related applications: an overview'. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8599–8603 (cit. on p. 33).

Denkowski, M. and A. Lavie (July 2011). 'Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems'. In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 85–91. URL: https://aclanthology.org/W11-2107 (cit. on pp. 81, 101).

Derczynski, L. et al. (Aug. 2017). 'SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours'. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 69–76 (cit. on p. 10).

Dernoncourt, F., M. Ghassemi and W. Chang (2018). 'A repository of corpora for summarization'. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (cit. on p. 15).

Devlin, J. et al. (June 2019). 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186 (cit. on pp. 26, 27, 43, 51, 84, 155).

Dolan, B. and C. Brockett (Jan. 2005). 'Automatically Constructing a Corpus of Sentential Paraphrases'. In: *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing (cit. on p. 43).

Donahue, J. and K. Simonyan (2019). 'Large Scale Adversarial Representation Learning'. In: *NeurIPS* (cit. on p. 115).

Dong, L. et al. (Sept. 2017). 'Learning to Paraphrase for Question Answering'. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 875–886 (cit. on pp. 44, 57).

Dosovitskiy, A. and T. Brox (2016). 'Generating Images with Perceptual Similarity Metrics based on Deep Networks'. In: *NIPS* (cit. on p. 116).

Dosovitskiy, A., L. Beyer et al. (2021). 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale'. In: *ArXiv* abs/2010.11929 (cit. on p. 111).

Duong, L. et al. (2015). 'Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser'. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 845–850 (cit. on p. 34).

El Desouki, M. I. and W. H. Gomaa (2019). 'Exploring the Recent Trends of Paraphrase Detection'. In: *International Journal of Computer Applications* 975, p. 8887 (cit. on p. 43).

Fader, A., L. Zettlemoyer and O. Etzioni (Aug. 2013). 'Paraphrase-Driven Learning for Open Question Answering'. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1608–1618 (cit. on p. 57).

Farazi, H. and S. Behnke (2019). 'Frequency Domain Transformer Networks for Video Prediction'. In: *ArXiv* abs/1903.00271 (cit. on p. 111).

Farazi, H., J. Nogga and S. Behnke (2021). 'Local Frequency Domain Transformer Networks for Video Prediction'. In: *arXiv preprint arXiv:2105.04637* (cit. on p. 111).

Fyodorov, Y., Y. Winter and N. Francez (2000). 'A natural logic inference system'. In: *Proceedings of the 2nd Workshop on Inference in Computational Semantics (ICoS-2)* (cit. on p. 11).

Gidiotis, A. and G. Tsoumakas (2020). 'A divide-and-conquer approach to the summarization of long documents'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28, pp. 3029–3040 (cit. on p. 32).

Giorgos, O. et al. (1999). 'Decision trees and NLP: A case study in POS tagging'. In: *Proceedings of annual conference on artificial intelligence (ACAI)* (cit. on p. 18).

Girshick, R. (2015). 'Fast R-CNN'. In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448 (cit. on p. 33).

Glorot, X., A. Bordes and Y. Bengio (2011). 'Deep sparse rectifier neural networks'. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 315–323 (cit. on p. 19).

Gomes, S. R. et al. (2017). 'A comparative approach to email classification using Naive Bayes classifier and hidden Markov model'. In: *2017 4th international conference on advances in Electrical Engineering (ICAEE)*. IEEE, pp. 482–487 (cit. on p. 18).

Gomez, A. N. et al. (2017). 'The reversible residual network: Backpropagation without storing activations'. In: *Advances in neural information processing systems*, pp. 2214–2224 (cit. on p. 30).

Graves, A. (2013). 'Generating sequences with recurrent neural networks'. In: *arXiv preprint arXiv:1308.0850* (cit. on p. 20).

Guan, J. et al. (2021). 'LOT: A Benchmark for Evaluating Chinese Long Text Understanding and Generation'. In: *arXiv preprint arXiv:2108.12960* (cit. on p. 28).

Haffner, P., G. Tur and J. Wright (2003). 'Optimizing SVMs for complex call classification'. In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)*. Vol. 1, pp. I–I (cit. on p. 18).

Hashimoto, K. et al. (Sept. 2017). 'A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks'. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 1923–1933. URL: https://www.aclweb.org/anthology/D17-1206 (cit. on p. 38).

He, K. et al. (2016). 'Deep Residual Learning for Image Recognition'. In: *CVPR*, pp. 770–778 (cit. on p. 118).

Hermann, K. M. et al. (2015). 'Teaching machines to read and comprehend'. In: *Advances in neural information processing systems* 28 (cit. on p. 16).

Hoare, C. A. R. (July 1961). 'Algorithm 64: Quicksort'. In: *Commun. ACM* 4.7, p. 321 (cit. on p. 31).

Hochreiter, S., Y. Bengio et al. (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies* (cit. on p. 24).

Hochreiter, S. and J. Schmidhuber (1997). 'Long short-term memory'. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 21).

Honnibal, M. et al. (2020). 'spaCy: Industrial-strength Natural Language Processing in Python'. In: (cit. on p. 94).

Horé, A. and D. Ziou (2010). 'Image Quality Metrics: PSNR vs. SSIM'. In: *ICPR*, pp. 2366–2369 (cit. on p. 127).

Howard, J. and S. Ruder (July 2018). 'Universal Language Model Fine-tuning for Text Classification'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 328–339 (cit. on p. 51).

Hunt, E. et al. (2019). 'Machine learning models for paraphrase identification and its applications on plagiarism detection'. In: *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, pp. 97–104 (cit. on p. 43).

Hutsebaut-Buysse, M., K. Mets and S. Latré (2022). 'Hierarchical reinforcement learning: A survey and open research challenges'. In: *Machine Learning and Knowledge Extraction* 4.1, pp. 172–221 (cit. on p. 32).

Ide, N. (2008). 'The American national corpus: Then, now, and tomorrow'. In: *Selected Proceedings of the 2008 HCSNet Workshop on Designing the Australian National Corpus: Mustering Languages, Summerville, MA. Cascadilla Proceedings Project* (cit. on p. 12).

Imtiaz, Z. et al. (2020). 'Duplicate questions pair detection using siamese malstm'. In: *IEEE Access* 8, pp. 21932–21942 (cit. on p. 44).

Isozaki, H. et al. (2010). 'Automatic evaluation of translation quality for distant language pairs'. In: *Proceedings of the 2010 conference on empirical methods in natural language processing*, pp. 944–952 (cit. on p. 14).

Iyyer, M. et al. (2018). 'Adversarial Example Generation with Syntactically Controlled Paraphrase Networks'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics (cit. on p. 44).

Jing, H. (2002). 'Using hidden markov modeling to decompose human-written summaries'. In: *Computational linguistics* 28.4, pp. 527–543 (cit. on p. 16).

Karatsuba, A. A. and Y. P. Ofman (1962). 'Multiplication of many-digital numbers by automatic computers'. In: *Doklady Akademii Nauk*. Vol. 145. 2. Russian Academy of Sciences, pp. 293–294 (cit. on p. 31).

Keskar, N. S. et al. (2019). 'Unifying Question Answering, Text Classification, and Regression via Span Extraction'. In: *arXiv: Computation and Language*. URL: https://api.semanticscholar.org/CorpusID:203701451 (cit. on p. 40).

Keung, P. et al. (2020). 'The Multilingual Amazon Reviews Corpus'. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* (cit. on p. 154).

Kitaev, N., L. Kaiser and A. Levskaya (2020). 'Reformer: The Efficient Transformer'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rkgNKkHtvB (cit. on pp. 28–31, 70).

Kočiský, T. et al. (Dec. 2018). 'The NarrativeQA Reading Comprehension Challenge'. In: *Transactions of the Association for Computational Linguistics* 6, pp. 317–328. URL: http://dx.doi.org/10.1162/tacl_a_00023 (cit. on pp. 27, 28, 72, 75, 76, 80, 81, 100, 101).

Kocmi, T. et al. (2022). 'Findings of the 2022 conference on machine translation (WMT22)'. In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pp. 1–45 (cit. on p. 14).

Koehn, P. (2005). 'Europarl: A parallel corpus for statistical machine translation'. In: *Proceedings of machine translation summit x: papers*, pp. 79–86 (cit. on p. 13).

Koehn, P. and C. Monz (2006). 'Manual and automatic evaluation of machine translation between european languages'. In: *Proceedings on the Workshop on Statistical Machine Translation*, pp. 102–121 (cit. on p. 13).

Kryściński, W. et al. (2021). 'Booksum: A collection of datasets for long-form narrative summarization'. In: *arXiv preprint arXiv:2105.08209* (cit. on p. 86).

Kuehne, H. et al. (2011). 'HMDB: A large video database for human motion recognition'. In: *ICCV*, pp. 2556–2563 (cit. on pp. 122, 125).

Kwiatkowski, T. et al. (2019). 'Natural questions: a benchmark for question answering research'. In: *Transactions of the Association for Computational Linguistics* 7, pp. 453–466 (cit. on p. 17).

Lang, K. (1995). 'Newsweeder: Learning to filter netnews'. In: *Machine learning proceedings 1995*. Elsevier, pp. 331–339 (cit. on p. 10).

Lecun, Y. et al. (1998). 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 113, 125).

LeCun, Y. (1985). 'Une procedure d'apprentissage ponr reseau a seuil asymetrique'. In: *Proceedings of Cognitiva 85*, pp. 599–604 (cit. on p. 20).

Levesque, H. J., E. Davis and L. Morgenstern (2011). 'The Winograd Schema Challenge'. In: *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. URL: https://api.semanticscholar.org/CorpusID:15710851 (cit. on p. 18).

Li, Y., Y. Song et al. (2016). 'TGIF: A New Dataset and Benchmark on Animated GIF Description'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4641–4650 (cit. on p. 116).

Li, Z., X. Jiang et al. (2019). 'Decomposable neural paraphrase generation'. In: *arXiv preprint arXiv:1906.09741* (cit. on p. 44).

Li, Z., N. B. Kovachki et al. (2021). 'Fourier Neural Operator for Parametric Partial Differential Equations'. In: *ICLR* (cit. on p. 117).

Lin, C.-Y. (July 2004). 'ROUGE: A Package for Automatic Evaluation of Summaries'. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81. URL: https://aclanthology.org/W04-1013 (cit. on pp. 14, 81, 101).

Lison, P., J. Tiedemann and M. Kouylekov (2018). 'OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora'. In: *LREC* (cit. on pp. 13, 76).

Liu, D., Y. Yan et al. (2021). 'GLGE: A New General Language Generation Evaluation Benchmark'. In: *FINDINGS* (cit. on pp. 28, 70).

Liu, N. F., M. Gardner et al. (2019). 'Linguistic Knowledge and Transferability of Contextual Representations'. In: *NAACL* (cit. on p. 129).

Long, J., E. Shelhamer and T. Darrell (2015). 'Fully convolutional networks for semantic segmentation'. In: *CVPR* (cit. on p. 118).

MacCartney, B. and C. D. Manning (2008). 'Modeling semantic containment and exclusion in natural language inference'. In: *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 521–528 (cit. on p. 11).

Manek, G. and J. Kolter (2020). 'Learning Stable Deep Dynamics Models'. In: *NeuIPS* (cit. on p. 117).

Maron, M. E. (June 1961). 'Automatic Indexing: An Experimental Inquiry'. In: *J. ACM* 8.3, pp. 404–417. URL: https://doi.org/10.1145/321075.321084 (cit. on p. 18).

McCann, B. and N. S. Keskar (2019). 'The Natural Language Decathlon: Multitask Learning as Question Answering'. In: Presentation given at the Bay Area Research in NLP and ML group. URL: https://www.youtube.com/watch?v=dJJkMDC82r8 (cit. on p. 42).

McCann, B., N. S. Keskar et al. (June 2018). 'The Natural Language Decathlon: Multitask Learning as Question Answering'. In: URL: http://arxiv.org/abs/1806.08730v1 (cit. on pp. 42, 51, 53, 55, 56, 58, 59, 67).

Mel'čuk, I. (1992). 'Paraphrase et lexique: la théorie Sens-Texte et le Dictionnaire explicatif et combinatoire'. In: *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques III. Les Presses de l'Université de Montréal*, pp. 9–58 (cit. on p. 43).

Mikolov, T. et al. (2010). 'Recurrent neural network based language model'. In: *INTERSPEECH* (cit. on p. 116).

Minervini, P. and S. Riedel (2018). 'Adversarially Regularising Neural'. In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Association for Computational Linguistics (cit. on p. 44).

Mishra, S. et al. (2021). 'Cross-task generalization via natural language crowdsourcing instructions'. In: *arXiv preprint arXiv:2104.08773* (cit. on p. 43).

Misra, I. et al. (2016). 'Cross-stitch networks for multi-task learning'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3994–4003 (cit. on p. 36).

Moratanch, N. and S. Chitrakala (2017). 'A survey on extractive text summarization'. In: *2017 international conference on computer, communication and signal processing (ICCCSP)*. IEEE, pp. 1–6 (cit. on p. 15).

Müller, M. et al. (2018). 'A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation'. In: *arXiv preprint arXiv:1810.02268* (cit. on pp. 14, 77, 82, 87, 94).

Nachum, O. et al. (2019). 'Why does hierarchy (sometimes) work so well in reinforcement learning?' In: *arXiv preprint arXiv:1909.10618* (cit. on p. 32).

Nakano, R. et al. (2021). 'Webgpt: Browser-assisted question-answering with human feedback'. In: *arXiv preprint arXiv:2112.09332* (cit. on p. 163).

Nallapati, R., B. Zhou, C. Gulcehre et al. (2016). 'Abstractive text summarization using sequence-to-sequence rnns and beyond'. In: *arXiv preprint arXiv:1602.06023* (cit. on p. 16).

Nallapati, R., B. Zhou and M. Ma (2016). 'Classify or select: Neural architectures for extractive document summarization'. In: *arXiv preprint arXiv:1611.04244* (cit. on p. 16).

Nallapati, R., B. Zhou, C. N. dos Santos et al. (2016). 'Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond'. In: *Conference on Computational Natural Language Learning*. URL: https://api.semanticscholar.org/CorpusID:8928715 (cit. on p. 20).

Nigam, K. et al. (2000). 'Text classification from labeled and unlabeled documents using EM'. In: *Machine learning* 39, pp. 103–134 (cit. on p. 10).

Oord, A. van den, Y. Li and O. Vinyals (2018). 'Representation Learning with Contrastive Predictive Coding'. In: *ArXiv* abs/1807.03748 (cit. on p. 115).

Oord, A. van den, O. Vinyals and K. Kavukcuoglu (2017). 'Neural Discrete Representation Learning'. In: *NIPS* (cit. on p. 115).

Oprea, S. et al. (2020). 'A Review on Deep Learning Techniques for Video Prediction'. In: *IEEE TPAMI* PP (cit. on pp. 111, 112, 117, 141, 142).

Over, P., H. Dang and D. Harman (2007). 'DUC in context'. In: *Information Processing & Management* 43.6, pp. 1506–1520 (cit. on p. 15).

Pang, R. Y. et al. (2021). 'QuALITY: Question Answering with Long Input Texts, Yes!' In: *arXiv preprint arXiv:2112.08608* (cit. on pp. 29, 72).

Papineni, K. et al. (July 2002). 'Bleu: a Method for Automatic Evaluation of Machine Translation'. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. URL: https://aclanthology.org/P02-1040 (cit. on pp. 14, 81, 101).

Pascanu, R., T. Mikolov and Y. Bengio (2013). 'On the difficulty of training recurrent neural networks'. In: *International conference on machine learning*. Pmlr, pp. 1310–1318 (cit. on p. 24).

Peters, M. E., M. Neumann, M. Iyyer et al. (2018). 'Deep contextualized word representations'. In: *Proc. of NAACL* (cit. on pp. 26, 51).

Peters, M. E., M. Neumann, L. Zettlemoyer et al. (2018). 'Dissecting Contextual Word Embeddings: Architecture and Representation'. In: *EMNLP* (cit. on p. 129).

Quinlan, J. R. (1993). 'Programs for machine learning'. In: *C4.5*. URL: https://cir.nii.ac.jp/crid/1573387449987904000 (cit. on p. 18).

Radev, D., E. Hovy and K. McKeown (2002). 'Introduction to the special issue on summarization'. In: *Computational linguistics* 28.4, pp. 399–408 (cit. on p. 15).

Radford, A. et al. (2019a). 'Language Models are Unsupervised Multitask Learners'. In: (cit. on pp. 1, 42).

Radford, A. et al. (2019b). 'Language models are unsupervised multitask learners'. In: *OpenAI Blog* 1.8, p. 9 (cit. on p. 115).

Raffel, C. et al. (2020). 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer'. In: *Journal of Machine Learning Research* 21.140, pp. 1–67 (cit. on pp. 27, 40, 51, 56, 61, 79).

Rajpurkar, P., R. Jia and P. Liang (July 2018). 'Know What You Don't Know: Unanswerable Questions for SQuAD'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 784–789. URL: https://aclanthology.org/P18-2124 (cit. on p. 17).

Rajpurkar, P., J. Zhang et al. (2016). 'SQuAD: 100,000+ Questions for Machine Comprehension of Text'. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. URL: http://dx.doi.org/10.18653/v1/D16-1264 (cit. on p. 17).

Rakhimov, R. et al. (2021). 'Latent Video Transformer'. In: *VISIGRAPP* (cit. on p. 111).

Ranchod, P., B. Rosman and G. Konidaris (2015). 'Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning'. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 471–477 (cit. on p. 32).

Rangel, F. et al. (2020). 'Overview of the 8th author profiling task at pan 2020: Profiling fake news spreaders on twitter'. In: *CEUR workshop proceedings*. Vol. 2696. Sun SITE Central Europe, pp. 1–18 (cit. on p. 10).

Ranzato, M. et al. (2014). 'Video (language) modeling: a baseline for generative models of natural videos'. In: *ArXiv* abs/1412.6604 (cit. on pp. 111, 115, 116, 152).

Rei, R. et al. (2020). 'COMET: A neural framework for MT evaluation'. In: *arXiv preprint arXiv:2009.09025* (cit. on p. 14).

Ribeiro, M. T., S. Singh and C. Guestrin (2018). 'Semantically Equivalent Adversarial Rules for Debugging NLP models'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics (cit. on p. 44).

Ronneberger, O., P. Fischer and T. Brox (2015). 'U-Net: Convolutional Networks for Biomedical Image Segmentation'. In: *MICCAI* (cit. on p. 118).

Ruder, S. (June 2017). 'An Overview of Multi-Task Learning in Deep Neural Networks'. In: URL: `http://arxiv.org/abs/1706.05098v1` (cit. on pp. 33, 35).

Ruder, S. et al. (2017). 'Sluice networks: Learning what to share between loosely related tasks'. In: *arXiv preprint arXiv:1705.08142* (cit. on p. 39).

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986a). 'Learning representations by back-propagating errors'. In: *nature* 323.6088, pp. 533–536 (cit. on p. 20).

— (1986b). 'Learning representations by back-propagating errors'. In: *nature* 323.6088, pp. 533–536 (cit. on p. 21).

Rush, A. M., S. Chopra and J. Weston (Sept. 2015). 'A Neural Attention Model for Abstractive Sentence Summarization'. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 379–389. URL: https://aclanthology.org/D15-1044 (cit. on p. 15).

Sanh, V., T. Wolf and S. Ruder (2018). 'A Hierarchical Multi-task Approach for Learning Embeddings from Semantic Tasks'. In: *arXiv preprint arXiv:1811.06031* (cit. on p. 38).

Scao, T. L. and A. M. Rush (2021). 'How many data points is a prompt worth?' In: *arXiv preprint arXiv:2103.08493* (cit. on p. 43).

See, A., P. J. Liu and C. D. Manning (2017). 'Get To The Point: Summarization with Pointer-Generator Networks'. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics (cit. on p. 55).

Shaham, U. et al. (2022). *SCROLLS: Standardized CompaRison Over Long Language Sequences* (cit. on pp. 28, 29).

Shi, X. et al. (2017). 'Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model'. In: *NeurIPS*. Vol. 30 (cit. on p. 117).

Singh, S. P. (1992). 'Transfer of learning by composing solutions of elemental sequential tasks'. In: *Machine learning* 8, pp. 323–339 (cit. on pp. 31, 85).

Skowron, M. et al. (2016). 'Automatic identification of character types from film dialogs'. In: *Applied Artificial Intelligence* 30.10, pp. 942–973 (cit. on pp. 75, 76).

Soomro, K., A. Zamir and M. Shah (Dec. 2012). 'UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild'. In: *CoRR* (cit. on p. 116).

Srivastava, N., E. Mansimov and R. Salakhutdinov (2015). 'Unsupervised Learning of Video Representations using LSTMs'. In: *ICML* (cit. on pp. 113, 125).

Stamatatos, E. et al. (2018). 'Overview of PAN 2018: Author identification, author profiling, and author obfuscation'. In: *Experimental IR Meets Multilinguality, Multimodality, and Interac-*

*tion: 9th International Conference of the CLEF Association, CLEF 2018, Avignon, France, September 10-14, 2018, Proceedings 9*. Springer, pp. 267–285 (cit. on p. 10).

Stein, B. et al. (2009). '3rd PAN workshop on uncovering plagiarism, authorship and social software misuse'. In: *25th annual conference of the spanish society for natural language processing (SEPLN)*, pp. 1–77 (cit. on p. 10).

Stiennon, N. et al. (2020). 'Learning to summarize with human feedback'. In: *Advances in Neural Information Processing Systems* 33, pp. 3008–3021 (cit. on p. 99).

Strøm, E. (2021). 'Multi-label Style Change Detection by Solving a Binary Classification Problem'. In: *CLEF* (cit. on pp. 80, 100).

Sulem, E., O. Abend and A. Rappoport (2020). 'Semantic structural decomposition for neural machine translation'. In: *Proceedings of the ninth joint conference on lexical and computational semantics*, pp. 50–57 (cit. on p. 32).

Sutskever, I., O. Vinyals and Q. V. Le (2014). 'Sequence to Sequence Learning with Neural Networks'. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf (cit. on p. 20).

Svore, K., L. Vanderwende and C. Burges (2007). 'Enhancing single-document summarization by combining RankNet and third-party sources'. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 448–457 (cit. on p. 16).

Tay, Y. et al. (2021). 'Long Range Arena : A Benchmark for Efficient Transformers'. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=qVyeW-grC2k (cit. on pp. 27, 28, 72).

Tomar, G. S. et al. (2017). 'Neural paraphrase identification of questions with noisy pretraining'. In: *arXiv preprint arXiv:1704.04565* (cit. on p. 44).

Tompson, J. et al. (2017). 'Accelerating Eulerian Fluid Simulation with Convolutional Networks'. In: *ICML*. Sydney, NSW, Australia, pp. 3424–3433 (cit. on p. 117).

Tong, S. and D. Koller (2001). 'Support vector machine active learning with applications to text classification'. In: *Journal of machine learning research* 2.Nov, pp. 45–66 (cit. on p. 10).

Touvron, H. et al. (2023). 'Llama 2: Open foundation and fine-tuned chat models'. In: *arXiv preprint arXiv:2307.09288* (cit. on p. 166).

Turing, A. M. (Oct. 1950). 'Computing Machinery and Intelligence'. In: *Mind* LIX.236, pp. 433–460 (cit. on p. 18).

Unterthiner, T. et al. (2019). 'FVD: A new Metric for Video Generation'. In: *ICLR* (cit. on p. 127).

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer (cit. on p. 18).

Vaswani, A. et al. (2017). *Attention Is All You Need* (cit. on pp. 22, 23, 25, 118).

Vila, M. (2013). 'Paraphrase Scope and Typology. A Data-Driven Approach from Computational Linguistics'. PhD thesis. Universitat de Barcelona (cit. on pp. xiii, 45).

Vila, M., M. Bertran et al. (Mar. 2015). 'Corpus annotation with paraphrase types: new annotation scheme and inter-annotator agreement measures'. In: *Language Resources and Evaluation* 49.1, pp. 77–105 (cit. on pp. 45, 50, 59).

Vila, M., M. A. Martí and H. Rodríguez (2014). 'Is This a Paraphrase? What Kind? Paraphrase Boundaries and Typology'. In: *Open Journal of Modern Linguistics* 04.01, pp. 205–218 (cit. on pp. 43, 59).

Vila Rigat, M. et al. (2011). 'Paraphrase concept and typology. A linguistically based and computationally oriented approach'. In: *Procesamiento del lenguaje natural* 46, pp. 83–90 (cit. on p. 45).

Vilca, G. C. V. and M. A. S. Cabezudo (2017). 'A study of abstractive summarization using semantic representations and discourse level information'. In: *Text, Speech, and Dialogue: 20th*

*International Conference, TSD 2017, Prague, Czech Republic, August 27-31, 2017, Proceedings 20*. Springer, pp. 482–490 (cit. on p. 15).

Wang, A., Y. Pruksachatkun et al. (2019). 'SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems'. In: *arXiv preprint 1905.00537* (cit. on pp. 28, 70).

Wang, A., A. Singh et al. (2018). 'GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding'. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics (cit. on pp. 28, 70).

Wang, B. (May 2021). *Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX*. `https://github.com/kingoflolz/mesh-transformer-jax` (cit. on pp. 27, 98).

Wang, R., R. Walters and R. Yu (2021). 'Meta-Learning Dynamics Forecasting Using Task Inference'. In: *ArXiv* abs/2102.10271 (cit. on p. 117).

Wang, S., B. Z. Li et al. (2020). 'Linformer: Self-attention with linear complexity'. In: *arXiv preprint arXiv:2006.04768* (cit. on pp. 30, 31, 70).

Wang, Y., J. Wu et al. (2020). 'Probabilistic Video Prediction From Noisy Data With a Posterior Confidence'. In: *CVPR* (cit. on p. 116).

Williams, A., N. Nangia and S. R. Bowman (2017). 'A broad-coverage challenge corpus for sentence understanding through inference'. In: *arXiv preprint arXiv:1704.05426* (cit. on pp. 11, 12).

Wong, S.-M. J. and M. Dras (2011). 'Exploiting parse structures for native language identification'. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1600–1610 (cit. on p. 10).

Wu, J., L. Ouyang et al. (2021). *Recursively Summarizing Books with Human Feedback* (cit. on pp. 33, 86, 92, 98, 101, 102).

Wu, J., J. J. Lim et al. (2016). 'Physics 101: Learning Physical Object Properties from Unlabeled Videos'. In: *BMVC* (cit. on p. 117).

Xie, E. et al. (2021). 'SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers'. In: *arXiv preprint arXiv:2105.15203* (cit. on pp. 118, 120, 121).

Yan, W. et al. (2021). *VideoGPT: Video Generation using VQ-VAE and Transformers* (cit. on pp. 111, 112, 116).

Yang, X., X. Zhu et al. (2019). 'Enhancing unsupervised pretraining with external knowledge for natural language inference'. In: *Advances in Artificial Intelligence: 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019, Kingston, ON, Canada, May 28–31, 2019, Proceedings 32*. Springer, pp. 413–419 (cit. on p. 43).

Yang, Y. and T. Hospedales (2017). 'Trace Norm Regularised Deep Multi-Task Learning'. In: *Workshop track -ICLR 2017* (cit. on p. 34).

Yang, Y., W.-t. Yih and C. Meek (2015). 'WikiQA: A Challenge Dataset for Open-Domain Question Answering'. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 2013–2018 (cit. on p. 17).

Yang, Z., P. Qi et al. (2018). 'HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering'. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (cit. on pp. 17, 86).

Yao, Y. et al. (2021). 'CUGE: A Chinese Language Understanding and Generation Evaluation Benchmark'. In: *ArXiv* abs/2112.13610 (cit. on pp. 28, 70).

Yilmaz, M. and A. Tekalp (2021). 'DFPN: Deformable Frame Prediction Network'. In: *ArXiv* abs/2105.12794 (cit. on p. 116).

Zangerle, E. et al. (2020). 'Overview of the Style Change Detection Task at PAN 2020.' In: *CLEF 2020* (cit. on p. 11).

— (2021). 'Overview of the Style Change Detection Task at PAN'. In: (cit. on p. 77).

Zeiler, M. D. and R. Fergus (2014). 'Visualizing and Understanding Convolutional Networks'. In: *ECCV* (cit. on p. 118).

Zhang, D., X. Chen and W. S. Lee (2005). 'Text classification with kernels on the multinomial manifold'. In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 266–273 (cit. on p. 10).

Zhang, R., P. Isola et al. (2018). 'The Unreasonable Effectiveness of Deep Features as a Perceptual Metric'. In: *CVPR*, pp. 586–595 (cit. on p. 127).

Zhang, T., V. Kishore et al. (2019). 'Bertscore: Evaluating text generation with bert'. In: *arXiv preprint arXiv:1904.09675* (cit. on p. 14).

Zhang, Y., A. Ni et al. (2021). 'Summˆ n: A multi-stage summarization framework for long input dialogues and documents'. In: *arXiv preprint arXiv:2110.10150* (cit. on p. 32).

Zhang, Z., Z. Han et al. (2021). 'Style Change Detection Based On Writing Style Similarity'. In: *PAN at CLEF 2021*, pp. 2208–2211 (cit. on pp. 80, 86, 100).

Zhang, Z., J. Yang and H. Zhao (2020). 'Retrospective Reader for Machine Reading Comprehension'. In: *AAAI Conference on Artificial Intelligence*. URL: https://api.semanticscholar.org/CorpusID:210920624 (cit. on p. 17).

Zhou, Y., H. Dong and A. El Saddik (2020). 'Deep Learning in Next-Frame Prediction: A Benchmark Review'. In: *IEEE Access* 8, pp. 69273–69283 (cit. on pp. 111, 117).

Zhou Wang et al. (2004). 'Image quality assessment: from error visibility to structural similarity'. In: *TIP* 13.4, pp. 600–612 (cit. on p. 127).

Ziemski, M., M. Junczys-Dowmunt and B. Pouliquen (2016). 'The united nations parallel corpus v1. 0'. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 3530–3534 (cit. on p. 13).

# Appendix A

# MQAN Error Examples

I present examples of when the MQAN model makes errors when given paraphrased questions. These are sampled from across the outputs of the standard MQAN model when run on the PQ-decaNLP test set (without training on paraphrase data).

| | |
|---|---|
| **IWSLT** | unanswerable |
| | price range; |
| | price range; dontcare |
| | senate, objected |
| | eight hundred thirty lines of information |
| **CNN/DM** | race organizers |
| | A ‘hero’ |
| | 1958 world cup |
| | best of the rest |
| | blueprint |
| **MNLI** | invasion |
| | Uh |
| | “ “ what |
| | pleasure |
| | “” “” |
| **SST** | one |
| | swim |
| | review |
| | treat |
| | movies |
| **WOZ** | unanswerable |
| | . |
| | ; |
| **WikiSQL** | Der table hat einen player |
| | Select date, date, caption„ >, min, >, >, >, >, >, >, >, >, >, >, >, >, >, > |
| | Der table hat eine ds division |
| | Der table hat sich in peru, spezies in peru,... |
| | the table hat tournament, 1988, 1989,... |
| **MWSC** | boring |
| | it |
| | a toy flag in the highest tower |
| | unanswerable |

# Appendix B

# PQ-decaNLP examples

I present examples from our PQ-decaNLP dataset of paraphrased decaNLP questions.

## MultiNLI

The string *sent* will be replaced with the hypothesis in the multiNLI example.

```
If our Hypothesis is: "{sent}", would this be entailment, neutral, or contradiction?
The hypothesis "{sent}" could be considered a a) contradiction b) entailment or c)
    neutral?
For the hypothesis: "{sent}", would this be considered an entailment, contradiction, or
     neutral hypothesis?
Is the hypothesis "{sent}" an example of entailment, contradiction or neutral?
For the hypothesis "{sent}", is it classified as entailment, neutral or contradiction?
If the hypothesis is "{sent}", would it be neutral, entailment or contradiction?
Given the choices of contradiction, entailment, or neutral -- what is the hypothesis:
    "{sent}".
```

```
Is the fact that "{sent}" -- an entailment, neutral, or contradiction?
Ascertain whether the hypothesis, "{sent},"  is an entailment, neutral, or a
    contradiction.
Is the hypothesis "{sent}" an entailment, neutral, or contradiction relationship?
The hypothesis is that "{sent}". Is this: 1: entailment 2: neutral or 3: contradiction.
Is it entailment, neutral, or contradiction that "{sent}"?
Is the hypothesis "{sent}", entailment, neutral or contradiction?
Determine if the hypothesis "{sent}" can be described as entailment, neutral or
    contradiction.
Tell me if the "{sent}" hypothesis would be contradiction, entailment or neutral?
For the hypothesis, "{sent}", which one applies? contradiction, entailment, or neutral?
Does an entailment, neutral, or contradiction best describe the hypothesis: "{sent}"?
Hypothesis: "{sent}" - which of the following fits best: contradiction, entailment, or
    neutral?
For the hypothesis: "{sent}" does the term entailment, neutral, or contradiction fit?
Does the hypothesis: "{sent}", show a neutral, entailment, or contradiction
    relationship?
Given a hypothesis: "{sent}", is it entailment, neutral, or contradiction?
Judge this hypothesis: "{sent}" -- is it entailment, neutral, or contradiction?
Suppose "{sent}" is an hypothesis, then which of these applies: entailment, neutral, or
     contradiction?
Would entailment, neutral, or contradiction be most accurate for the hypothesis "{sent
    }"?
Hypothesis: "{sent}". Choose: 1) entailment; 2) neutral; 3) contradiction.
Hypothesis: "{sent}". Determine if this is entailment, neutral, or contradiction.
Given that the hypothesis is "{sent}", does this sound like entailment, neutral or
    contradiction?
"{sent}" (Tell me if this hypothesis is an example of an entailment, is neutral or is a
     contradiction).
Provided with a hypothesis: "{sent}", Establish whether this is entailment,
    contradiction, or is neutral.
entailment, neutral, or contradiction -- Which applies to the hypothesis: "{sent}"?
```

## Schema

{mainQuestion} Let me know if it's {choice1} or {choice2}.

{mainQuestion} Tell me if it's {choice1} or {choice2}.

{mainQuestion} Select whether {choice1} or {choice2} is valid.

{mainQuestion} Figure out whether {choice1} is true or if {choice2} is.

{mainQuestion} Is {choice1} or {choice2} correct?

{mainQuestion} Is it {choice1} or is it {choice2}?

{mainQuestion} Decide between {choice1} and {choice2}.

{mainQuestion} Between {choice1} and {choice2}, say which is right.

{mainQuestion} Is it {choice1} or {choice2} that should be selected?

{mainQuestion} Does {choice1} or {choice2} answer it? Choose one.

{mainQuestion} Between {choice1} and {choice2}, decide on one.

{mainQuestion} Could it be {choice1} or {choice2}?

{mainQuestion} If you had to pick one answer, would it be {choice1}, or would it be {choice2}?

{mainQuestion}, {choice1} or possibly {choice2}?

{mainQuestion} Choose: i. {choice2} ii. {choice1}.

{mainQuestion} It is either {choice1} or {choice2}. Decide.

{mainQuestion} {choice1} and {choice2} are your two options.

{mainQuestion} {choice1} or {choice2}? Make up your mind.

{mainQuestion} One of two options: Either A: {choice1}, or B: {choice2}.

Is it {choice1} or {choice2} that answers: {mainQuestion}

{mainQuestion} Would you say it is {choice1} or {choice2}?

{mainQuestion} Is the correct response {choice1} or {choice2}?

{choice1} or {choice2} -- {mainQuestion}

{mainQuestion} Is {choice1} or {choice2} the one answer?

Does {choice2} or {choice1} answer {mainQuestion}?

{mainQuestion} -- Options: {choice1}, {choice2}?

{mainQuestion} Will the answer be {choice1} or {choice2}?

{mainQuestion} Make a decision between {choice1} or {choice2}.

Your options are {choice1} or {choice2} for the question: {mainQuestion}

{mainQuestion} {choice1} is an option. {choice2} is the other.

## Sentiment

Is the review for this movie positive or is it negative?

Would this review be described as positive or negative?

Is this review a more positive or a more negative one?

Could this be a positive review or could it be a negative review?

Does the review tend to be positive or negative?

Would this review be mainly negative or positive?

Find out for me if they say the movie's review is positive or negative.

Is this review more positive or negative?

Is this a negative or positive review?

Is this a positive review or a negative review?

Is there a negative or positive opinion in the review?

Was the tone of the review more positive or negative?

Is this review positive or negative?

Would you characterize this review as positive or negative?

Is this a review that would be read as positive or negative?

Is the overall review more positive or negative?

Was the text positive or negative?

I think I want to know whether the review is positive or negative.

How should we consider the review: negative or positive?

Would you say that the review is negative or positive?

Is the review positive or negative?

Is the review positive, or is it negative?

Tell me whether the review is positive or negative.

Can you tell me if the review is positive or negative?

Is this a negative or a positive review?

Would this review be considered negative or positive?

Tell me if the overall review is more positive or negative?

Is this review positive or is this review negative?

Is this review (1) positive (2) negative?

Class the opinion of the review as negative or positive.

## Summarization

What is the synopsis?

Give a brief description of the article.

What's the main idea of the article?

What's the gist of the article?

Give me a brief synopsis of the article.

Tell me the main points of the article.

Give me a debriefing.

Compact all of that information into a short summary.

What was the main point?

What should the reader take from this?

What's the point of the text?

How can it be summed up?

Tell me in a few words what the article is about.

Briefly tell me what the main points of the article are.

Provide a synopsis.

What's the TL;DR?

Find the main ideas.

What is the idea that covers the whole story?

What are the important points of the article?

How can we summarize the important points of the article?

What is a condensation of the article?

Briefly, what does the article say?

Find a short version of the article.

What are the highlights of the article?

The article could be summarized by saying what?

What are the main ideas of the reading?

How would you shorten this?

Can you sum that up?

In a nutshell?

What is the key point?

## Translation

*source* and *target* are placeholders which are replaced with the source and target languages (English and German in our experiments) at runtime.

```
Convert from {source} to {target}.
Translate from the {source} language to {target}
Convert from the {source} language to the {target} language
{target} version of this {source} sentence is?
Provide a {target} translation for the following {source} phrase.
How would the sentence be written if it was changed to {target} from {source}?
What are the quotes of the talks translated to {target} in place of the {source}?
How do I say this {source} in {target}?
How is this {source} sentence said in {target}?
How do I write the {source} in {target}?
What does this {source} sentence mean in {target}?
What is the {target} translation for this {source} phrase?
Make the language {target} instead of {source}
Translating {source} to {target} will result in what phrase?
Give me the {target} translation of this {source} phrase.
What is the {target} translation for this {source}.
I would like to translate from {source} to {target}.
I am looking to translate from {source} to {target}.
What is the interpretation of {source} to {target}?
Translate what is being said from {source} to {target}.
Tell me the translation of what is being said from {source} to {target}.
If you were speaking {target} how would you translate the {source}?
Please convert the quotes from {source} to {target}.
Convert the given quotes from {source} into {target}.
Interpret the {source} words into {target}.
How does this translate from {source} to {target}?
Can this be translated from {source} to {target}?
Translate the text from {source} to {target}.
Let it be translated from {source} to {target}
```

```
From {source} to {target}, what will the translation be?
```

## WikiSQL

```
Can I get a translation from English to SQL?
Can I find the translation from English to SQL?
Could you tell me what the translation is from English to SQL?
Convert this from English into SQL.
Tell me what this English sentence converts to in SQL.
I need to know what this English is converted to in SQL.
Convert the English sentence to SQL statement.
Please translate this from English to SQL.
What is the translated SQL statement from it's English statement?
What is the best translation of these English words in SQL?
How can I find this English line in Structured Query Language?
What is the SQL equivalent of the English?
Translate from English to SQL.
English to SQL translation.
Create an SQL translation from an English sentence.
Convert from English to SQL.
Alter this from English to SQL.
Make this English SQL instead.
I need to know how to say in SQL what is in English.
Change the query from English into SQL.
Change the statement to SQL from English.
Transform English into SQL with the same meaning.
Transfer any meaning from English to SQL.
Replace the English concepts with their SQL equivalents.
Tell me the translation from English to SQL.
Explain to me the conversion from English to SQL.
Tell me the conversion from English to SQL.
Tell me what the English sentence means in SQL.
```

Change the text from English to Structured Query Language.

When it is English how do you say it in SQL?

## WoZ

Can you tell me the changes in state?

Please tell me the changes in state.

Give the dialogue state change.

What is the change in the intention of the user?

The change in state is what?

I need to know what the change in state is.

Give the change in state.

Give any variations in state

How has the state varied?

How has the dialogue state changed?

How has the user's dialogue state changed?

How has the state changed?

Find the difference in state.

Find any movement in dialogue state?

Where are the changes in state?

How has the state changed?

Give me the details about the change in state.

Tell me the change in state.

What has changed in the state?

How did the dialogue state alter?

What changed about the state?

Discover the adjustment in dialogue state.

What's the change in the state?

Detail the change in state.

Where is the change in state?

Are you able to find a change in state?

What is the change in user intention.

```
How has the state changed?
Provide the differences in the new state.
For the state, find the change.
```

## Annotated Examples

Here I show a few examples of the paraphrase typology annotation which was applied to the dataset:

**original**: What is the summary ?

**paraphrase**: Give a brief description of the article .

**original**: Is this review negative or positive

**paraphrase**: Is this review more positive or negative .

**original**: What is the change in state ?

**paraphrase**: How has the state varied ?

**Key:** Sentence modality change | Same-polarity substitution | Addition/Deletion | Order | Synthetic/Analytic

# Appendix C

# MuLD Benchmark Examples

I present examples from the MuLD benchmark to give the reader a sense of the tasks included. Ellipsis is added for brevity):

## NarrativeQA

**Input:**

```
How is Oscar related to Dana?
[...]EXT.  MANHATTAN ISLAND - DAY
A high AERIAL SHOT of the island features the Statue of Liberty prominently in the
foreground then TRAVELS ACROSS the harbor, OVER the Battery and Lower Manhattan to
Greenwich Village[...]
                                                DANA
                                        (exasperated)
                              Frank, do you think you could give me a hand
                              with these bags?
```

FRANK

I'm not a doorman, Miss Barrett. I'm a building superintendent[...]

**Output:**

her son

## HotpotQA

**Input:**

Were Scott Derrickson and Ed Wood of the same nationality? Doctor Strange is a 2016 American superhero film based on the Marvel Comics character of the same name. Produced by Marvel Studios and distributed by Walt Disney Studios Motion Pictures, it is the 14th film in the Marvel Cinematic Universe (MCU)[...]
Scott Derrickson (born July 16, 1966) is an American filmmaker. He is best known for directing the films The Exorcism of Emily Rose, Sinister, Deliver Us from Evil, and Doctor Strange[...]
Edward Davis Wood Jr. (October 10, 1924 - December 10, 1978) was an American filmmaker, actor, and author[...]

**Output:**

Yes

## Style Change

**Input:**

```
John’s stomach had a new home again. It now settled into his throat, painfully dry for
    want of the man before him.
"Which story would you like to hear, my dear?" John’s fingers looped in and around the
    curls so delicately, a lesser effort wouldn’t have moved the hair at all. He
    thought of the beautiful mind beneath these curls. Once storing every memory like
    a computer and now weakened with age- but always beautiful.
John gulped down a lump of panic that crawled up his throat. What did he know? "Right,
    but I still reckon I’ll keep pretending you don’t."
"John, when we need to pretend to be a couple, things like spontaneous physical contact
     are expected to maintain-"
"Don’t mention it," Sherlock responded with a soft passion he couldn’t keep out of his
    voice. The words came out in a murmur.
-" he continued, delighted when John joined in to harmonize flawlessly in the latter
    half of phrase[...]
```

**Output:**

```
1,1,1,1,1,1,1,[...],2,2,1,1,[...],3
```

# Character Types

**Input:**

```
Indiana Jones
[...]
                INDY
        No.

Barranca looks evilly at Indy’s hand upon him.
Indy releases him and smiles in a friendly way.

                INDY
        We don’t need them.
```

```
Satipo watches this confrontation with
some concern.

                BARRANCA
        I do not carry supplies.

                INDY
        We'll leave them. Once we've got it,
        we'll be able to reach the plane by
        dusk[...]
```

**Output:**

```
Hero
```

# VLSP

**Input:**

```
## Chapter 1 Basic Definitions and Concepts

### 1.1 Basics of simplicial complexes

Let @xmath , and @xmath denote the subsets of @xmath of size @xmath . A
collection @xmath of subsets of @xmath is called a (finite abstract)
simplicial complex if it is closed under inclusion, i.e. @xmath implies
@xmath . Note that if @xmath is not empty (which we will assume from now
on) then @xmath . The @xmath -th skeleton of @xmath is @xmath . The
elements of @xmath are called faces ; those in @xmath have dimension i .
The @xmath -dimensional faces are called vertices , the @xmath
-dimensional faces are called edges and the maximal faces with respect
to inclusion are called facets . If all the facets have the same
```

```
dimension, @xmath is pure . The @xmath - vector (face vector) of @xmath
is @xmath where @xmath . The dimension of K is @xmath ; e.g. a
1-dimensional simplicial complex is a simple graph. The @xmath -
polynomial of @xmath is @xmath[...]
```

**Output:**

```
This thesis focuses on algebraic shifting and its applications to f-vector
theory of simplicial complexes and more general graded posets. In particular,
several approaches and partial results concerning the g-conjecture for
simplicial spheres are presented here.
```

## OpenSubtitles

**Input:**

```
1957 was a big year.
The Russians put that Sputnik into outer space.
The Dodgers played their last game at Ebbets Field and said goodbye to Brooklyn.
That guy, he shot Frank Costello in the head and missed.
The Gallo brothers whacked Albert Anastasia in that barbershop.
It was total chaos[...]
```

**Output:**

```
1957 war ein bedeutendes Jahr.
Die Russen schossen ihren Sputnik ins All.
Die Dodgers spielten zum letzten Mal in Ebbets Field und sagten Brooklyn Adieu.
Dieser Kerl schoss auf Frank Costello und verfehlte ihn.
Die Gallo-Bruder legten Albert Anastasia beim Friseur um.
Es war totales Chaos[...]
```

# Appendix D

# Visual Modelling

## D.1 Further Model Training Details

All inputs are preprocessed to grayscale images of dimension 64x64. The pixel values are then normalised to $\in$ [0,1] by dividing all pixels by 255. A pixel-wise sigmoid activation is used to create the output image and calculate metrics and losses. The visualisations of outputs in this paper are the result of multiplying the pixel values of the outputs by 255 and rounding them.

### D.1.1 Fully Convolutional CNN

'Depth' refers to the number of broad convolution steps in both the downwards and upwards directions. My models are of depth 3 (see Figure 6.3) and the convolutions have both a kernel size of 3x3 and padding of 1. I found that varying the depth and kernel size did not yield much difference in results, and I have accordingly fixed them at their highest performing values. The linear probes (for probing frozen models) are created from:

1. the outputs of the final layer.

2. the outputs of each double convolution layer in both the downward and upward direction.

### D.1.2 Patch Transformer

As described earlier, the patch transformer is formed from two objects from the SegFormer repository[1]:

1. As an encoder, the MixVisionTransformer object with: in_chans = number of input frames, img_size = 64, sr_ratios = [1,1,1,1] (scale reduction ratios).

2. As a decoder, the SegFormerHead object with: feature_strides = [4,8,16,32], in_channels = [64,128,256,512], channels = 128, num_classes = 16, in_index = [0, 1, 2, 3], decoder_params = {“embed_dim":256}, dropout_ratio = 0.1, align_corners = False. The linear probes (for probing frozen models) are created from: **I)** the outputs of the encoder, **II)** the resized and reshaped outputs of each of the linear projection layers in the decoder, **III)** the output ‘linear_fuse’ convolution module which takes as input the concatenation the outputs described in **II)**.

[1] https://github.com/NVlabs/SegFormer

## D.2 CNN Figures

This section contains further visualisations of the experimental setup for the CNN model.

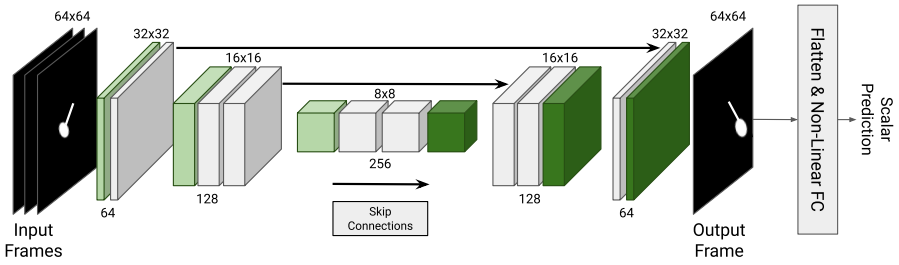



Figure D.1: Fully Convolutional 2D, depth=3, unfrozen weights test-task model. Training an unfrozen model to compare with through the linear probes (Figure D.2) did not converge. I instead append 2 non-linear fully connected layers to the flattened outputs of the model.

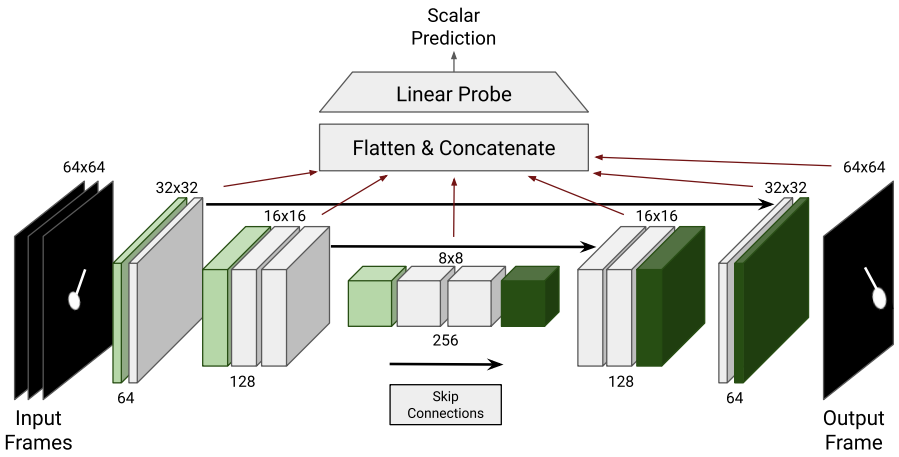



Figure D.2: Fully Convolutional 2D model. Depth=3, with linear probes on the flattened outputs of each block is sufficient to verify if information exists in the model to predict the given physical property scalar.

## D.3 Tabularisation of Results

To complement my visualisations in the main text, I share the results in table form.

| Metric | CNN | | Img Trans | | Patch Trans | |
|---|---|---|---|---|---|---|
| | **SL1** | **SSIM** | **SL1** | **SSIM** | **SL1** | **SSIM** |
| **2D Bouncing** $m = 5$ | | | | | | |
| PSNR↑ | 31.036 | 30.400 | 28.572 | 25.900 | 30.924 | 31.328 |
| SSIM↑ | 0.9717 | 0.9767 | 0.9439 | 0.9488 | 0.9714 | 0.9788 |
| L1↓ | 4.051e-3 | 8.879e-3 | 8.053e-3 | 2.009e-2 | 5.062e-3 | 8.069e-3 |
| **2D Bouncing** $m = 59$ | | | | | | |
| PSNR↑ | 28.468 | 28.446 | 25.504 | 25.915 | 24.564 | 25.007 |
| SSIM↑ | 0.9577 | 0.9641 | 0.8993 | 0.8989 | 0.8979 | 0.9534 |
| L1↓ | 6.393e-3 | 1.016e-2 | 1.443e-2 | 2.154e-2 | 1.957e-2 | 2.251e-2 |
| **3D Bouncing** $m = 5$ | | | | | | |
| PSNR↑ | 30.772 | 27.903 | 38.661 | 35.631 | 40.859 | 28.138 |
| SSIM↑ | 0.9477 | 0.9734 | 0.9847 | 0.9894 | 0.9892 | 0.9953 |
| L1↓ | 6.247e-3 | 9.284e-3 | 1.988e-3 | 2.410e-3 | 1.773e-3 | 2.839e-3 |
| **3D Bouncing** $m = 99$ | | | | | | |
| PSNR↑ | 30.571 | 28.013 | 31.443 | 33.269 | 39.060 | 26.823 |
| SSIM↑ | 0.9447 | 0.9712 | 0.9454 | 0.9778 | 0.9850 | 0.9926 |
| L1↓ | 6.468e-3 | 9.388e-3 | 4.318e-3 | 4.297e-3 | 2.337e-3 | 5.120e-3 |
| **Roller** $m = 5$ | | | | | | |
| PSNR↑ | 32.475 | 29.023 | 30.381 | 33.063 | 48.358 | 43.404 |
| SSIM↑ | 0.9914 | 0.9950 | 0.9898 | 0.9949 | 0.9998 | 0.9998 |
| L1↓ | 2.879e-3 | 3.642e-3 | 1.077e-3 | 9.768e-4 | 3.051e-4 | 3.600e-4 |
| **Pendulum** $m = 5$ | | | | | | |
| PSNR↑ | 38.941 | 36.031 | 26.246 | 36.574 | 51.085 | 43.948 |
| SSIM↑ | 0.9972 | 0.9980 | 0.9608 | 0.9988 | 0.9998 | 0.9998 |
| L1↓ | 7.231e-4 | 8.512e-4 | 3.117e-3 | 5.420e-4 | 1.492e-4 | 1.938e-4 |
| **Blocks** $m = 49$ | | | | | | |
| PSNR↑ | 31.561 | 30.365 | 41.905 | 43.402 | 51.069 | 47.371 |
| SSIM↑ | 0.9904 | 0.9922 | 0.9980 | 0.9986 | 0.9996 | 0.9996 |
| L1↓ | 5.128e-3 | 6.285e-3 | 5.585e-4 | 1.166e-3 | 2.510e-4 | 7.025e-4 |
| **Moon** $m = 5$ | | | | | | |
| PSNR↑ | 45.399 | 45.515 | 38.928 | 38.785 | 54.618 | 54.618 |
| SSIM↑ | 0.9988 | 0.9990 | 0.9930 | 0.9945 | 0.9998 | 0.9998 |
| L1↓ | 6.844e-4 | 7.699e-4 | 6.529e-4 | 8.937e-4 | 2.100e-4 | 2.100e-4 |
| **MMNIST** $m = 5$ | | | | | | |
| PSNR↑ | 17.794 | 18.097 | 13.904 | 13.089 | 17.975 | 18.237 |
| SSIM↑ | 0.8591 | 0.8641 | 0.6777 | 0.6768 | 0.8654 | 0.8700 |
| L1↓ | 2.880e-2 | 2.992e-2 | 5.607e-2 | 0.05853 | 2.784e-2 | 2.812e-2 |
| **MOCAP** $m = 5$ | | | | | | |
| PSNR↑ | 25.816 | 24.024 | 30.431 | 30.226 | 32.943 | 31.835 |
| SSIM↑ | 0.8738 | 0.8917 | 0.9523 | 0.9515 | 0.9777 | 0.9807 |
| L1↓ | 2.485e-2 | 3.495e-3 | 1.147e-2 | 1.393e-2 | 7.456e-3 | 1.241e-2 |
| **HDMB51** $m = 5$ | | | | | | |
| PSNR↑ | 22.229 | 21.675 | 22.141 | 21.673 | 23.664 | 23.191 |
| SSIM↑ | 0.7566 | 0.7633 | 0.7119 | 0.7160 | 0.8378 | 0.8431 |
| L1↓ | 4.109e-2 | 4.627e-2 | 4.649e-2 | 5.102e-2 | 3.004e-2 | 3.549e-2 |

Table D.1: Metrics between the first generated image and its respective ground truth. All metrics are reported from the best epoch of the models respective loss. The L1 metric is calculate with mean reduction. ↑ (↓) indicates higher (lower) is better.