A

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

by

APPROVAL SHEET

This

is submitted in partial fulfillment of the requirements
for the degree of

Author:

Advisor:

Advisor:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

Efficient collection, evaluation and deployment of large scale deep learning models in low resource natural language processing scenarios

by

Debajyoti Datta

A Dissertation Presented
for the degree of
Doctor of Philosophy

Graduate Supervisory Committee:

Michael Porter, Chair
Donald E. Brown
Laura Barnes
Jennifer Chiu
Nikolaos Sidiropoulos

University of Virginia

April 12th 2023

ABSTRACT

Deep learning in natural language processing revolutionized low-resource domains like education and healthcare with approaches like transfer learning and prompting-based methods. Large language models can generalize to new tasks in low-resource environments; however, domain-specific data collection often beats generalized data for a given model size.

My dissertation aims to take the advances in deep learning and natural language processing and apply them in the context of low-resource domains like education and healthcare. Through this work, we have collected data using model-assisted labeling by improving the annotation speed by up to 33% and efficiently devised a method to determine examples near the decision boundary of deep learning-based classifiers.

Evaluation of Deep Learning models is tricky because seemingly small perturbations to the input data (for example changing the articles in a sentence) can cause the classification label to change. We proposed a differential-geometry-based approach to find examples that are most and least susceptible to this change.

We also explored how to train and deploy transformer-based models in educational scenarios efficiently. We proposed a tensorized adapter approach (using tensor decomposition-based methods) that reduced the number of tunable parameters of Large Language Model without a drop in performance.

ACKNOWLEDGMENTS

After I began my Ph.D. in 2019, the world experienced the COVID-19 pandemic. It took an enormous toll on human lives and affected every corner of the world.

However, amidst all the chaos and uncertainty, I was fortunate to have supportive collaborators, advisors, family, and friends who helped me navigate these unprecedented times. Their guidance and support have been invaluable, and I am grateful for their unwavering assistance.

Dr. Donald E Brown for being incredibly supportive during my Ph.D. It is tough to emphasize how he has supported every crazy idea I wanted to pursue. I would also like to thank Dr. Laura Barnes for the countless brainstorming and discussion sessions at UVa.

I would also like to thank my collaborators and advisors, Dr. Nikolaos Sidiropoulos and Dr. Tom Fletcher, for introducing me to the fascinating world of Tensors and Differential Geometry. Dr. Jennifer Chiu, Dr. Jim Bywater, and Dr. Ginger Watson for introducing me to the fantastic world of education research and its broad impact on society.

I am eternally grateful to my parents, Dr. Nitai Datta and Arundhati Datta, who exemplified the importance of asking the right questions and solving problems that matter. My sister Noel Datta, for continuously encouraging and supporting me in this journey.

I would also like to thank my friends Trishala Neeraj, Shashwat Kumar, and Soumee Guha for their constant help and support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

The last few years have seen unprecedented improvements in Natural Language Processing and artificial intelligence. These improvements led to significant progress in adjacent fields like education and healthcare that use natural language processing. A lot of these improvements have resulted from modeling improvements using optimal training methods and modern architectures. Improvement in model performance has resulted mainly from larger models (models with up to a trillion parameters). However, model architectures and larger models only address part of the challenge in the application of NLP systems. Low resource domains still have a "data" problem. Specific tasks in a field like question answering in the context of math education require labeled data for math education. Generic data collected in another domain is often not sufficient to achieve the desired performance [6, 7].

However collecting large datasets for a task is non-trivial. The cost of collecting and labeling datasets is non-trivial and designing effective systems to collect data can improve not just the amount of data collected but also the quality of the data. In our work, we prototype a system for a mathematical training scenario for classroom instructions.

The nature and quality of classroom instruction is highly correlated to teachers' ability to rehearse effective teaching strategies. Utilizing research-based teaching practices increases teacher effectiveness, confidence, and retention along with improving student achievements. High-fidelity, AI-based simulated classroom systems enable teachers to rehearse and get feedback on specific pedagogical skills. One primary challenge is that current conversational agents (CA) can have task-oriented conversations,

however more varied dialogue-oriented conversations such as that between a teacher and student for a domain-specific task (like a mathematical scenario) can be difficult to model. In the first chapter we present a high-fidelity, AI-based classroom simulator to help teachers rehearse research-based mathematical questioning skills. The system relies on advances in deep-learning uncertainty quantification and natural language processing while acknowledging the limitations of CAs for specific pedagogical needs.

A second challenge in low resource data scenarios is understanding failure modes. Recent findings indicate how larger models are not necessarily better but use shortcuts [8] to improve metrics like accuracy. Understanding what leads to failure modes in specific tasks will improve interpretability and reliability of NLP systems. A growing body of evidence has suggested that metrics like accuracy overestimate the classifier's generalization ability. Several state of the art Natural Language Processing (NLP) classifiers like BERT and LSTM rely on superficial cue words (e.g., if a movie review has the word "romantic", the review tends to be positive), or unnecessary words (e.g., learning a proper noun to classify a movie as positive or negative). One approach to test NLP classifiers for such fragilities is analogous to how teachers discover gaps in a student's understanding: by finding problems where small perturbations confuse the student. While several perturbation strategies like contrast sets or random word substitutions have been proposed, they are typically based on heuristics and/or require expensive human involvement. In the second chapter, using tools from information geometry, we propose a principled way to quantify the fragility of an example for an NLP classifier. By discovering such fragile examples for several state of the art NLP models like BERT, LSTM, and CNN, we demonstrate their susceptibility to meaningless perturbations like noun/synonym substitution, causing their accuracy to drop down to 20 percent in some cases. Our approach is simple, architecture agnostic and can be used to study the fragilities of text classification models.

A critical challenge in using Large Language Models is the deployability of these models in low resource scenarios. Since a multi-stage dialogue system requires multiple copies of the model, our goal is to reduce the total amount of memory required for a dialogue system. In this work, we use tensor decomposition based approaches to improve task regularization and parameter efficiency of fine-tuned models (reducing the number of copies of the transformer models). Adapters, Compactors, and BitFit are parameter-efficient approaches to fine-tuning large language models by freezing the entire language model and adding task-specific, fully-connected layers. The goals are two-fold: Parameter efficiency and improved task regularization. Tensorized layers (CP, Tucker, TensorTrain, and TensorRing) are more efficient and have also been shown to act as regularizers. One natural extension is using Tensorized layers to parameterize the adapter layers and simultaneously exploit parameter efficiency and regularization ability. Our work extensively studies Tensorized layers that can naturally exploit the low-rank structure of the adapter layers and provide a drop-in replacement for Adapters. We empirically investigate the different tensor decomposition approaches and the impact of tensor ranks in the adapter layers.

Chapter 2

OZ BASED DATA COLLECTION SYSTEM

2.1 Introduction

Despite ample evidence showing that deliberate practice can improve teachers' mathematical questioning, teachers are rarely given opportunities to rehearse these kinds of questioning strategies in pre-service or in-service settings due to a variety of constraints in teacher preparation programs. However, computer-based systems can provide ways for pre-service and in-service teachers to practice and receive feedback on mathematical questioning skills. This chapter presents the development of an AI-based classroom teaching system (ACTS) designed to help teachers rehearse mathematical questioning strategies that leverages advances in conversational agent (CA) development. In particular, this chapter describes the use of a human expert working with the computer-based system in a supervisor-type role to step in and keep the conversation going when the CA may fail. We also show how AI assisted data labeling for CA's can improve the speed and efficiency of data collection. The goal of the system is to simultaneously collect data for conversational agent components, while maintaining a coherent conversation and relying on state of the art advances in natural language processing systems. This chapter reports on the development and user testing of the ACTS system.

2.2 Related Works

Conversational agent evaluation is tricky because different systems have different evaluation requirements. [9] showed the limitations with automatic evaluations of

dialogue systems often are poor evaluations of dialogue systems. Interactive evaluations [10] where both the user of the system evaluates the system is critical specifically in scenarios like teacher education because one critical component of this evaluation is the user's perception of students understanding of the concepts. Our evaluation metrics for the dialogue system comprise of both interactive evaluation components and automated evaluation components [10].

Uncertainty has been recently explored in deep learning for classification tasks either for out of data distributions [11] and also for distributions shifts [12]. However, Active Dropout [13] is a cheap and effective way to add uncertainty to classification tasks, and we use this approach to add uncertainty to our dialogue acts module.

Including multiple modalities is critical in scenarios in teaching since it relies on visual aspects. Engaging with images has been [14] has been explored for conversations, and they show the effectiveness of using an image as a modality for interaction.

Data collection and data labeling are the primary bottlenecks in applying deep learning methodologies in new domains. A paradigm similar to weak supervision proposed by [15] used deep learning to understand annotator accuracy. They also showed the robustness of deep learning to label noise [16]. However, these approaches scale with large datasets and only show promising results when large-scale datasets are available, which is not often the case in education. Moreover, with larger datasets, inconsistencies are so common that models rely on shortcut learning [8] for classification tasks. The largest and the most popular dataset in deep learning is riddled with inconsistencies of various kinds [17], and there are social and political consequences of using them [18]. In this work, we wanted to collect a dataset only for the evaluation of IQA since it is a well-established metric for teacher evaluation and our goal was to make it efficient to label and annotate using domain experts over crowd workers.

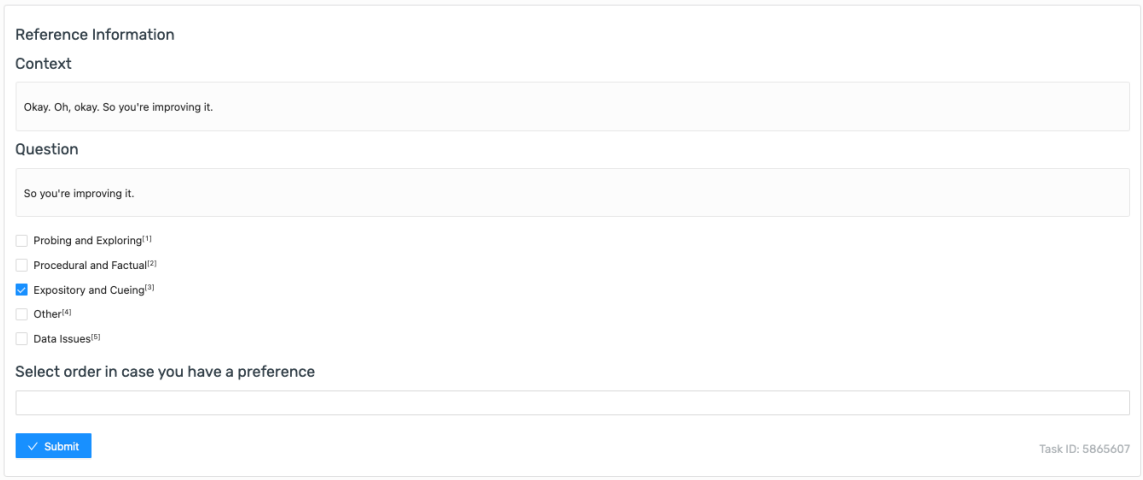

Figure 2.1: With the weak supervision rules, we trained a DistilBERT [1] model for category classification. The annotators had access to this screen pre-filled with one of the category labels (Probing and Exploring, Procedural and Factual, and Expository and Cueing and Other) before annotators labeled. The annotators have the option of agreeing with the label or choosing a different option. The annotators could also select multiple options and indicate their order preference if they were unsure. This information was not used to train the models but helps build more robust annotation guidelines for subsequent iterations of data collection.

2.3 Current Challenges

Current challenges to CA development in educational contexts include:

- **Model and Data Limitations (MDL)**: Deep learning models (models that are used in most modern CAs) can fail catastrophically because they work by exploiting spurious relationships [19] within data sets which is known as shortcut learning [8].
- **User Expectation Limitations (UEL)**: The "deep gulf of evaluation" [20] exists because CA systems lack meaningful feedback regarding the systems

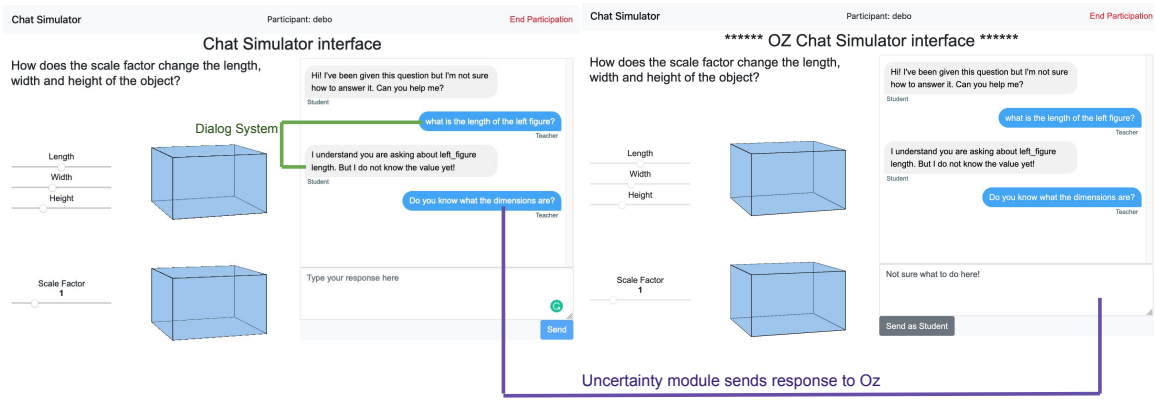

Figure 2.2: **The ACTS system has chat functionality and a dynamic representation of the mathematical task. The expert user has access to the Supervisor Interface to the right. The dialogue system responds when it is certain about the dialogue acts and entities. When the uncertainty thresholds are met, it sends the prompt to the expert user. The expert user can then type in the response as a student, and it will appear on the prompt (left). This prevents conversations from breaking because of the failure of ML pipelines and components.**

intelligence and capability. This mismatch of user expectations and current technology capabilities can lead to a mis-application of CAs and a lack of confidence that results in an avoidance of use and deployment for complex tasks or sensitive activities.

- **Limitations of Pretrained Models**: Generic models like BERT [21] do not readily generalize to specific domains. To address this, education communities need education-specific pre-trained models, like the biomedical community [7].

- **Limitations in data collection practices:** The primary mode of data collection in education contexts for domain-specific scenarios often relies on text transcriptions from noisy classroom videos. This coupled with privacy concerns

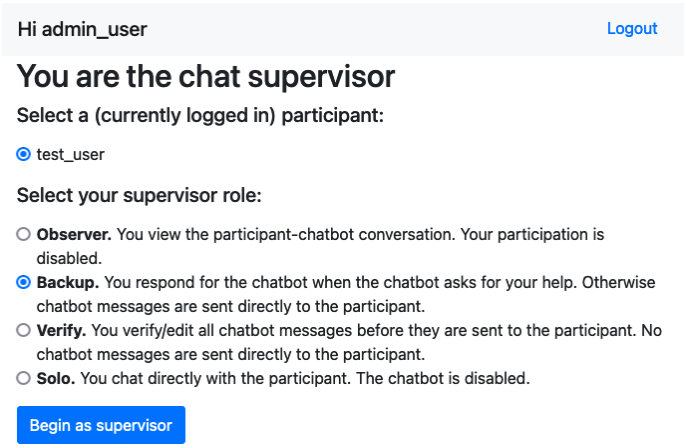

Figure 2.3: **The system enables data collection both during evaluation of the natural language processing modules and building scenarios for a new system. The "Verify" mode enables data collection for new scenarios while the "Backup" mode can redirect control to the human when a language component fails.**

makes data collection for addressing MDL non-trivial.

2.4 System Description

The **MDL** and **UEL** are difficult to meet in CA's for any system and especially for a CA in the context of education. In order to design useful systems, we need to acknowledge that current advances in dialogue systems make having fluent conversations over multiple turns non-trivial. We designed a dialogue system to specifically address user expectation failure and model and data failures. At it's core, the system minimizes negative user experiences by incorporating uncertainty modeling. When a dialogue system component or a system fails, we redirect control to a supervisor whose task is to bring the conversation back on track. We rely on recent developments in deep learning and uncertainty estimation for each subcomponent of the dialogue

Technique		Annotator	n	Annotator Accuracy	Agreement (kappa)	Model-Assisted Agreement	Time M(SD) (seconds)	*p-value*
A	Classical Labeling	Both	1730	0.82	0.52	-	15.2(42.1)	<0.001
		A1	864	0.89		-	13.2(37.8)	
		A2	866	0.74		-	17.3(45.9)	
B	WS - MAL	Both	3983	0.84	0.61	0.70	10.4(32.7)	
		B1	1994	0.89		0.80	7.1(18.3)	
		B2	1989	0.79		0.60	13.6(42.3)	

Table 2.1: Performance comparison of annotators between traditional supervised labeling and weak-supervision + model-assisted labeling approach. The gold accuracy table refers to anonymized annotators accuracy compared to gold labels generated by expert teachers with significant experience in evaluating IQA metrics.

system and build an interaction pipeline of user + system + supervisor that can pass the system's control from the dialogue system to the supervisor based on the failure. The uncertainty modules prevent the conversation from derailing due to a failure of one or more of the sub-components. The supervisor is present during the user engagement with the application acting within the role as an expert user (someone familiar with the scenario and the system's limitations) with the ability to send a response back to the system as a "Student". While perfect conversations in task-specific dialogue systems are challenging to achieve in complex scenarios, adding an expert in the loop prevents conversations from derailing. This also enables better user satisfaction and long-term data collection for better modeling or never-ending learning [22] scenarios.

2.5 Conclusion and Future Work

Initial user studies demonstrate an improvement of the user experience of our conversational agent despite the limitations of a complex, low-data educational sce-

nario by including uncertainty module elements and allowing for human-in-the-loop interactions. Even with the uncertainty modules, uncertainty quantification in deep learning is not robust [23]. Also, uncertainty in deep learning has been explored mostly for classification tasks but dialogue systems have many other tasks (entity recognition, turn-taking). Thus accuracy of uncertainty estimates also vary based on the stage of the dialogue system. In the future, we are planning to explore two lines of work: determining approaches so that we can jointly model uncertainty of all the stages together as a more robust uncertainty metric and controlling the amount of supervisor involvement as we collect more data in these training scenarios. We believe that these kinds of discrete, domain-specific web-based simulated classroom systems can provide needed opportunities to help pre-service and in-service teachers rehearse specific pedagogical strategies.

Chapter 3

MODEL EVALUATION

3.1 Introduction

NLP classifiers have achieved state of the art performance in several tasks like sentiment analysis [24], semantic entailment [25], and question answering [26]. Despite their successes, several studies have pointed out issues in features learnt by such classifiers. [27] discovered that several high performing NLP models were using trivial features like vagueness and negation to perform classification. [8] performed several experiments to discover that "romantic" movies tend to be classified as positive movie reviews due to the presence of unnecessary words like proper nouns, a phenomena they called shortcut learning. Even models like BERT [28] rely on superficial cue words like "not" to infer the line of argumentation. Strategies like this enable models to perform prediction without inherently using the semantic meaning of the sentence [29]. Simple attributes like word lengths are also exploited by models for prediction [30].

In order to address these issues, several perturbation based approaches have been proposed. Contrast Sets [31] and Counterfactual examples [32] get human annotators to perform minimal token substitutions to construct challenging test sets for the classifier. While useful in addressing biases, manual curation of datasets are often time consuming and require extensive efforts. Using unsupervised training to mitigate issues related to shortcut learning has not been successful in practice [29].

While interesting, most traditional formulations treat this input embedding space as flat, thus reasoning that the gradient of the likelihood in the input space gives us the direction that causes the most significant change in likelihood. If we were, however,

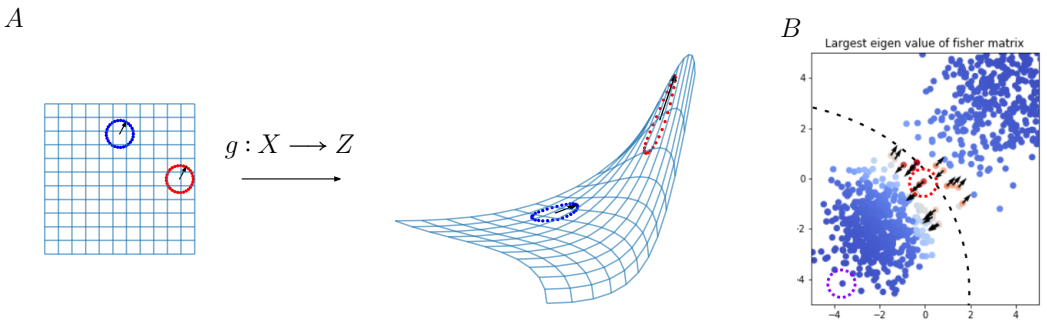

Figure 3.1: **A**: A neural network can be considered as a mapping g between the sentence manifold X on the left (represented as a Euclidean space for simplification) and the statistical manifold of probability distribution over outputs Z. The fisher metric defines a particular Riemannian metric over this manifold. Let us consider an ϵ ball around two sentences x_1 (blue circle) and x_2 (red circle). As we can see from the distortion of the blue circle, for x_1, any local perturbation can only result in a small change over the models output probability distribution which and vice versa for red circle. The eigenvalue of the fisher matrix quantifies this local distortion. **B**: We train a neural network to learn a decision boundary (black) to separate two gaussians and color each data point by the local distortion (blue implying low distortion, red implying higher distortion). As we can see from the colors, perturbing a point close to the decision boundary (red circle, same as red circle in A) results in a larger change over the model's output probability distribution than a point away from the boundary (blue circle, same as blue circle in A).

to consider the discrete likelihood of the class probabilities as the model output and the input as a pullback of this output, the space of output probability distributions is certainly non-linear, and the Euclidean distance metric no longer suffices. We show this schematically in Figure 1A, where the transformation g maps elements of the sentence manifold X to a non-linear statistical manifold Z. A natural distance

metric to consider on this manifold is the Fisher Information Metric, which along with being a Hessian of the KL divergence, is also a useful distance measure between probability distributions. Furthermore, the Fisher Information Metric is invariant to transformations like changing the model architecture, provided the likelihood remains the same. We thus use the eigenvalues of the Fisher matrix to discover high fragility regions in the statistical manifold. In regions with high λ_{max}, small perturbations can cause large changes in the output probability distribution (Figure 1A, red circle). Linguistically, this corresponds to the classifier being susceptible to meaningless perturbations like noun, synonym substitutions. Similarly, in regions with low λ_{max} no local perturbation can affect the classifier, resulting in the classifier being resilient upto 20% percent of word substitutions.

Our main contribution can thus be summarized as the following. 1. We propose a second order statistic, the log of the largest eigenvalue of Fisher matrix in order to capture linguistic fragilities. 2. We extensively establish the empirical relationship between λ_{max} and success probability of random word substitutions. To the best of our knowledge, this is the first work analyzing properties of the fisher metric to understand classifier fragility in NLP. The rest of the chapter is organized as follows: In Section 2, we summarize related work. In Section 3, we discuss our approach of computing the FIM and the gradient-based perturbation strategy. In Section 4, we discuss the results of the eigenvalues of FIM in synthetic data and sentiment analysis datasets with BERT and CNN. We also do extensive quantitative evaluations using 4 other text datasets. Finally, in Section 5, we discuss the implications of studying the eigenvalues of FIM for evaluating NLP models.

Table 3.1: **Top row**: Substituting even a single word for fragile examples (large λ_{max}) causes BERT to change the predicted sentiment. **Bottom Row**: Robust examples (small λ_{max}), however, retain positive sentiment despite multiple substitutions of positive words with negative words.

Perturbed sentiment	**Word substitutions**
Positive → Negative fragile example (λ_{max} =0.78)	OK, I kinda like the idea of this movie. I'm in the age demographic, and I kinda identify with some of the stories. Even the sometimes tacky and meaningful dialogue seems realistic, and in a different movie would have been forgivable.¡br /¿¡br /¿I'm trying as hard as possible not to trash this movie like the others did, but it's robust when the filmmakers were trying very hard.¡br /¿¡br /¿The editing in this movie is terrific! Possibly the **best** → **worst** editing I've ever seen in a movie! There are things that you don't have to go to film school to learn, leaning good editing is not one of them, but identifying a bad one is.¡br /¿¡br /¿Also, the shot... Oh my God the shots, just fantastic! I can't even go into the details, but we sometimes just see random things popping up, and that, in conjunction with the editing will give you the most exhilirating film viewing experience.¡br /¿¡br /¿This movie being made on low or no budget with 4 cast and crew is an excuse also. I've seen short films on youtube with a lot less artistic integrity! ...
Positive → Positive robust example (λ_{max} =0.55)	This is the **best and most** original show seen in years. The more I watch it the more I **fall in love with** → **hate** it. The cast is **excellent** → **terrible** , the writing is **great** → **bad**. I personally **loved** → **hated** every character. However, there is a character for everyone as there is a good mix of personalities and backgrounds just like in real life. I believe ABC has done a great service to the writers, actors and to the potential audience of this show, to cancel so quickly and not advertise it enough nor give it a real chance to gain a following. There are so few shows I watch anymore as most TV is awful . This show in my opinion was right down there with my favorites Greys Anatomy and Brothers and Sisters. In fact I think the same audience for Brothers and Sisters would hate this show if they even knew about it.

3.2 Related Work

In NLP, improperly trained machine learning models (for instance in the presence of limited/biased data) for classification often rely on spurious statistical patterns of the text and use *shortcut* for learning to classify. These can range from annotation artifacts [27,33,34], spelling mistakes [35], or new test conditions that require world knowledge [36].

Another issue of language recently has been that static benchmarks (e.g., GLUE by [5]) tend to saturate quickly because of the availability of ever-increasing compute and harder benchmarks are needed to evaluate NLP models (e.g., SuperGlue [37]). A more sustainable approach to this is the development of moving benchmarks. One notable initiative in this area is the Adversarial NLI [38], but most of the research community hardly validate their approach against this sort of moving benchmark. In the Adversarial NLI dataset, the authors propose an iterative, adversarial human-and-model-in-the-loop solution which makes models robust by training the model iteratively on difficult examples. In approaches like never-ending learning, models improve and test sets get difficult over time [22]. A moving benchmark is necessary since we know that improving performance on a constant test set may not generalize to newly collected datasets under the same condition [39,40]. Therefore, it is essential to find fragile examples in a more disciplined way. Approaches based on geometry have recently started gaining traction in computer vision literature. [41] studied universal statistics of the eigenvalues of the Fisher matrix to conclude that the parameter landscape is flat in most dimensions but very strongly distorted in others. [42] studied the connections between the Fisher metric and natural gradient. [43] used a similar approach for understanding adversarial examples in images, a formulation we extend to language in our work.

Table 3.2: **Top row**: In fragile examples synonym or change of name, changes classifier label. **Bottom row**: In robust examples, despite multiple simultaneous antonym substitutions, the classifier sentiment does not change.

Perturbed sentiment	**Word substitutions**
Positive → Negative fragile example (λ_{max} =5.25)	Going into this movie, I had heard good things about it. Coming out of it, I wasn't really amazed nor disappointed. Simon Pegg plays a rather childish character much like his other movies. There were a couple of laughs here and there– nothing too funny. Probably my **favorite → preferred** parts of the movie is when he dances in the club scene. I totally gotta try that out next time I find myself in a club. A couple of stars here and there including: Megan Fox, Kirsten Dunst, that chick from X-Files, and Jeff Bridges. I found it quite amusing to see a cameo appearance of Thandie Newton in a scene. She of course being in a previous movie with Simon Pegg, Run Fatboy Run. I see it as a toss up, you'll either enjoy it to an extent or find it a little dull. I might add, **Kirsten Dunst → Nicole Kidman, Emma Stone, Megan Fox, Tom Cruise, Johnny Depp, Robert Downey Jr.** is adorable in this movie. :3
Negative → Negative robust example (λ_{max} =0.0008)	I missed this movie in the cinema but had some idea in the back of my head that it was worth a look, so when I saw it on the shelves in DVD I thought "time to watch it". Big mistake!¡br /¿¡br /¿A long list of stars cannot save this turkey, surely one of the **worst → best** movies ever. An **incomprehensible → comprehensible** plot is **poorly → exceptionally** delivered and **poorly → brilliantly** presented. Perhaps it would have made more sense if I'd read Robbins' novel but unless the film is completely different to the novel, and with Robbins assisting in the screenplay I doubt it, the novel would have to be an **excruciating → exciting** read as well.¡br /¿¡br /¿I hope the actors were well paid as they looked embarrassed to be in this waste of celluloid and more lately DVD blanks, take for example Pat Morita. Even Thurman has the grace to look uncomfortable at times.¡br /¿¡br /¿Save yourself around 98 minutes of your life for something more worthwhile, like trimming your toenails or sorting out your sock drawer. Even when you see it in the "under $5" throw-away bin at your local store, resist the urge!

3.3 Computation of the KL divergence

Consider an η perturbation in the sentence space. In that case the classifier probability changes as show in Equation (3.1):

$$KL(p(y|x)||p(y|x+\eta)) \tag{3.1}$$

$$= E_{p(y|x)} \log \frac{p(y|x)}{p(y|x+\eta)} = E_{p(y|x)} \log p(y|x) - E_{p(y|x)} \log p(y|x+\eta) \tag{3.2}$$

Performing a taylor expansion of 2^{nd} term

$$E_{p(y|x)} \log p(y|x+\eta) = E_{p(y|x)}((\log p(y \mid x) + \eta^\top \nabla \log p(y|x) + \eta^\top \nabla^2 \log p(y|x)) \tag{3.3}$$

The first order term:

$$\eta^\top \nabla \log p(y|x)$$

The Hessian is:

$$\eta^\top \nabla^2 \log p(y|x)$$

Now the first order term vanishes because:

$$E_{p(y|x)} \eta^\top \nabla \log p(y|x)\Big) - \tag{3.4}$$

$$= \eta^\top \sum_y p(y|x) \nabla_y \log p(y|x) - \tag{3.5}$$

$$= \eta^\top \sum_y p(y|x) \left(\frac{1}{p(y|x)} \cdot \nabla_y p(y|x) \right) - \tag{3.6}$$

$$= \eta^\top \sum_y \nabla_y p(y|x) - \tag{3.7}$$

Since gradient is a linear operator, we can interchange summation ($\sum$) and (∇).

$$= \text{ eg: } \nabla_{y_1} p\left(y_1 \mid x\right) + \nabla_{y_2} p\left(y_2 \mid x\right)$$

$$= \nabla\left(p\left(y_1 \mid x\right) + p\left(y_2 \mid x\right)\right) = \nabla \sum_{y} p(y \mid x) = \nabla \cdot 1 = 0$$

Thus the first order term vanishes and the final equation becomes:

$$= E_{p(y|x)} \log p(y|x) - E_{p(y|x)} \log p(y)x\Big) - 0$$

$$- \ E_{p(y|x)} \eta^{\top} \nabla^2 \log p(y|x)\Big)\, \eta$$

After ignoring the higher order terms we finally get:

$$= \eta^{\top} \left(E_{p(y|x)} - \nabla^2 \log p(y|x)\right) \eta$$

And $\nabla^2 \log p(y|x)$ is the expectation of the Hessian.

Thus we only need to compute the following term

$$= \eta^{\top} G \eta$$

where

$$G = E_{p(y|x)} - \nabla_x^2 log p(y|x)$$

Consider the manifold of all possible sentence embeddings X. We show this schematically in Figure 1A, left. Although, We represent this as a euclidean space for simplicity, in the general setting this manifold could be non linear. Let $x \epsilon X$ be a sentence on this manifold, and y be the label vector corresponding to this sentence. The neural network $p(y|x)$ maps each sentence to a probability distribution over y. The set of all such probability distributions forms a statistical manifold Z (Figure 1A, right), Depending on the properties of the neural network, an η change in x (red circle/blue circle) can result in a large change in $p(y|x)$. We seek to quantify this change by measuring the KL divergence between the two original and η perturbed sentence.

By studying the eigenvalues of the fisher matrix, we can quantify the local distortion. As we see in Figure 1, the red sentence has a larger λ_{max}, resulting in a large change in the prob distribution. The blue sentence has a much smaller λ_{max}, meaning that perturbations around this sentence are less likely to affect $p(y|x)$ and thus the model accuracy.

After getting the eigenvalues of the FIM, we can use the largest eigenvalue λ_{max} to quantify how fragile an example is to linguistic perturbation. We propose the following procedure to calculate λ_{max}

Algorithm 1 Algorithm for estimating fragility of an example

Input: x: Sentence representation, f: Neural Network

Output: λ_{max}

Calculate probability vector :

1: $p = f(x)$

Calculate Jacobian of log probability w.r.t x

2: $J = \nabla_x logp$

Duplicate probability vector along rows to match J's shape

3: $p_c = duplicate(p, J.dim[0])$

Compute the FIM

4: $G = p_c J J^T$

Perform eigendecomposition to get the eigenvalues

5: $\lambda_s, v_s = eigendecomposition(G)$

6: **return** $max(\lambda_s)$

3.3.1 Models and Datasets

We used multiple model architectures to understand the implications of FIM. Convolutional Neural Networks [44], LSTM [45], Fasttext [46] and BERT [28]. For

datasets, we used IMDB [24], AG_NEWS [47], Sogou News [47] and Yelp Review Polarity [47]. [1]

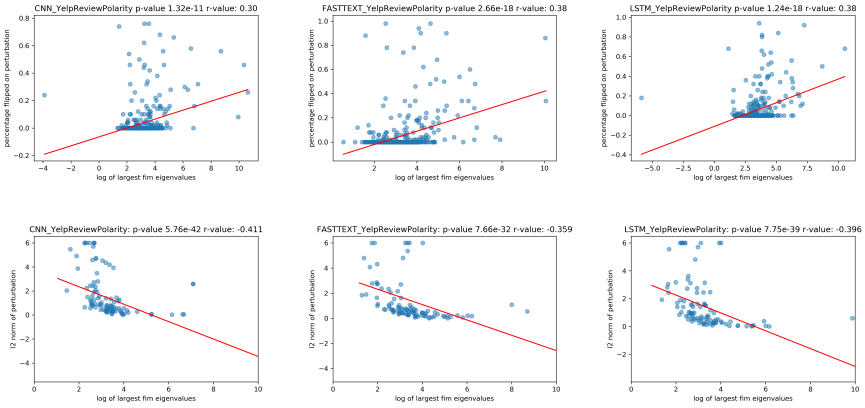

Figure 3.2: **Top row**: Correlation between λ_{max} and probability of a random word substitution substitution p_{flip} for CNN, LSTM and FastText classifiers. **Bottom row**: Correlation between λ_{max} and minimum perturbation strength to flip classifier output for CNN, LSTM and FastText classifiers. The linear relationship of λ_{max} with both p_{flip} and minimum perturbation strength demonstrates that λ_{max} captures perturbation sensitivity in both embedding space and word substitutions.

3.4 Discussion and Results

We now quantitatively and qualitatively explore how the Fisher Information metric relates to properties like accuracy and investigate it's feasibility in finding examples that are susceptible to perturbations.

3.4.1 λ_{max} reflects distances from the decision boundary

We first investigate the FIM properties by training a neural network on a synthetic mixture of gaussians dataset. The parameters of the two gaussians are $\mu_1 = [-2, -2]$

[1] More details about models and datasets can be found here: https://arxiv.org/abs/2010.07212

and $\mu_2 = [3.5, 3.5]$. The covariances are $\Sigma_1 = eye(2)$ and $\Sigma_2 = [[2., 1.], [1., 2.]]$ We train a 2-layered network to separate the two classes from each other. We use algorithm 1 to compute λ_{max} for each datapoint, and use it to color the points. We also plot the eigenvector for the top 20 points.

As seen by the gradient of the colors in Figure 1B, the points with the largest λ_{max} tend to lie close to the decision boundary. These points (red circle) are indicative of how fragile the example is to the neural network since a small shift along the eigenvector can cause a significant change in the KL divergence between the probability distribution of the original and new data points. For points away from the boundary (blue circle), there is minimal effect of local perturbations.

3.4.2 FIM captures resilience to lingustic perturbations

In this section we explore relationship of FIM with linguistic perturbations (synonym/antonym substitutions, noun replacements), token level substitutions (random word choice, nearest neighbors in GloVE embeddings) and embedding perturbations.

λ_{max} is correlated with fragility to word substitutions

In order to investigate the relationship between λ_{max} and linguistic fragility, we perform the following experiment: We randomly sample 1000 examples from Yelp ReviewPolarity dataset. For each sampled example, we try a batch of word substitutions based on nearest neighbours in glove embedding space. In each substitution attempt, we try to flip between 10 to 20 percentage of words at a time, and calculate the percentage of successful flips which change the classifier prediction p_{flip}. We also calculate the λ_{max} for these examples using the procedure outlined in Algorithm 1.

As we see from Figure 2 and Table 4, On YelpReviewPolarity, we observe r values of 0.3, 0.38 and 0.38 for CNN, FastText and LSTM respectively. It's interesting to note

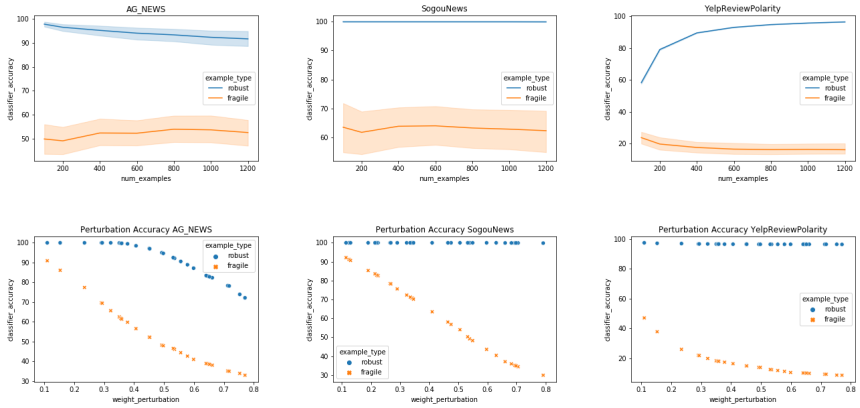

Figure 3.3: **Top row**: We sample n low λ_{max} (robust) examples and n high λ_{max} (fragile) examples and plot classification accuracy as a function of n. Robust examples retain high accuracy when token substitutions are performed whereas accuracy on fragile examples is significantly lower. In other words, high λ_{max} examples are much more fragile (p_{flip} ~ 0.5 on AG_NEWS, p_{flip} ~ 0.38 on SogouNews, p_{flip} ~ 0.85 on YelpReviewPolarity) as opposed to low λ_{max} examples (p_{flip} ~ 0.05 on AG_NEWS, p_{flip} ~ 0 on SogouNews, p_{flip} ~ 0.1 on YelpReviewPolarity). **Bottom row**: Similar to above, we sample n robust and fragile examples and plot the classifier accuracy vs the norm of perturbation. High λ_{max} examples are significantly more fragile to perturbations, with even small perturbations (weight of 0.5 on AG_NEWS, 0.5 on SogouNews, 0.1 on YelpReviewPolarity) can cause the classifier accuracy to plummet near 40-50 percent. The low λ_{max} examples have nearly 100% accuracy for these perturbation strengths.

that although FIM is defined in a continuous space, we observe a linear relationship with success probability of token substitutions, which are discrete. This could be indicative of the fact that flipping a small number of tokens in a large sentence is akin to an ϵ perturbation in embedding space, which gets captured by λ_{max}.

λ_{max} captures strength of the minimal sufficient perturbation

We were also interested in investigating the relationship between λ_{max} and norm of the smallest vector (oriented along the largest eigenvector e_{max}) which can cause the classifier to flip it's prediction. We perform the following experiment: We randomly sample 500 examples from YelpReview. For each of these examples, we calculate λ_{max} and e_{max} and perturb the sentence embedding with a strength of η, where

$$x_{flip} = x_{orgin} + \eta \frac{e_{max}}{\|e_{max}\|}$$

By performing binary search on η, we can determine the minimum perturbation strength sufficient to flip the classifier prediction.

As evident from Figure 2 and Table 5, we obtain r values of -0.41, -0.36 and -0.39 for CNN, FastText and LSTM, respectively. These correlations suggest that λ_{max} captures the perturbability of a sentence, with lower λ_{max} examples requiring much larger perturbation strengths in order to flip their prediction.

3.4.3 High λ_{max} examples cause substantial drop in accuracy

In order to study the effect of λ_{max} on classification accuracy, we sort the examples by λ_{max} and pick n low and high λ_{max} examples. We then plot the classifier accuracy as a function of number n. We repeat this procedure 6 times with n ranging from 200 to 1200.

As we see from Figure 3, low λ_{max} examples retain accuracies of 90-100 percent across all 3 datasets. They are much more resilient to perturbations. In other words, high λ_{max} examples are much more fragile, with much higher probabilities of random word substitutions causing misclassifications ($p_{flip} \sim 0.5$ on AG_NEWS, $p_{flip} \sim 0.38$ on SogouNews, $p_{flip} \sim 0.85$ on YelpReviewPolarity) as opposed to low λ_{max} examples ($p_{flip} \sim 0.05$ on AG_NEWS, $p_{flip} \sim 0$ on SogouNews, $p_{flip} \sim 0.1$ on YelpReviewPolarity).

Table 3.3: Statistics of correlation between random word substitution success probability p_{flip} and λ_{max}.

Dataset	**Architecture**	**p-value**	**r-value**
YelpReviewPolarity	CNN	1.32e-11	0.30
YelpReviewPolarity	FastText	2.66e-18	0.38
YelpReviewPolarity	LSTM	1.24e-18	0.38
AG_NEWS	CNN	1.43e-11	0.24
AG_NEWS	FastText	3.81e-16	0.21
AG_NEWS	LSTM	2.36e-10	0.26
SogouNews	CNN	7.32e-8	0.22
SogouNews	FastText	2.22e-15	0.23
SogouNews	LSTM	4.21e-18	0.31

As we see from the trend with n, this relationship is stable across different sample sizes of low and high.

We also investigated the effect of perturbation strength along largest eigenvector e_{max} on classifier accuracy. In figure 3b, we pick n low and high fim examples and plot classifier accuracy as a function of perturbation strength. As we see from the difference between the orange and blue curves in Figure 3, part B, Similar to above, we sample n robust and fragile examples and plot the classifier accuracy vs the norm of perturbation. High λ_{max} examples are significantly more fragile to perturbations, with even small perturbations (weight of 0.5 on AG_NEWS, 0.5 on SogouNews, 0.1 on YelpReviewPolarity) can cause the classifier accuracy to plummet near 40-50 percent. Low λ_{max} examples on the other hand, are remarkably resilient, with classifier accuracy remaining near 100 percent at these perturbation strengths.

Table 3.4: Statistics of correlation between min sufficient perturbation strength and λ_{max}.

Dataset	Architecture	p-value	r-value
YelpReviewPolarity	CNN	5.76e-42	-0.411
YelpReviewPolarity	FastText	7.66e-32	-0.359
YelpReviewPolarity	LSTM	7.75e-39	-0.396
AG_NEWS	CNN	6.38e-30	-0.31
AG_NEWS	FastText	6.24e-28	-0.30
AG_NEWS	LSTM	5.87e-29	-0.27
SogouNews	CNN	7.84e-32	-0.26
SogouNews	FastText	7.43e-32	-0.19
SogouNews	LSTM	7.72e-39	-0.21

3.4.4 Qualitative exploration of fragilities

We also qualitatively explore the nature of linguistic fragilities for both high and low λ_{max} examples. We sample high and low λ_{max} examples from the two tails of the λ_{max} distributions. We then perform several token level substitutions: synonym, antonym substitutions, noun, and article substitutions. The tokens for substitutions were selected using word attribution score from Integrated Gradients ([48]). Integrated gradients assign importance scores to words by the network and provide a more methodical approach to word substitutions than random word substitutions for qualitative evaluations.

As seen in Table 3.2, either replacing favorite with preferred or "Kirsten Dunst" with any of the listed actors/actresses suffices to change the classifier's prediction. Note that "Megan Fox's" name appears in the same review in the previous sentence. Similarly in Table **??**, it's sufficient to replace "exciting" with either an antonym ("boring" or

"uninteresting") or a synonym ("extraordinary" or "exceptional"). However, for robust examples, despite trying to replace four or more high attribution words simultaneously with antonyms, the predicted sentiment did not change. Substitutions include "good" to "bad", "unfunny" to "funny", "factually correct" to "factually incorrect". Even though the passage included words like "boredom", a word that is generally associated with a negative movie review, the model did not assign it a high attribution score. Consequently, we did not try to substitute these words for testing robustness or fragility of word substitutions.

For our models, fragile examples have a mix of positive and negative words in a movie review. The models also struggled with examples of movies that selectively praise some attributes like acting (e.g., "Exceptional performance of the actors got me hooked to the movie from the beginning") while simultaneously use negative phrases (e.g., "however the editing was horrible"). Fragile examples also have high token attributions associated with irrelevant words like "nuclear," "get," and "an." Thus substituting one or two words in fragile examples change the predicted label of the classifier. Similarly, easier examples have clearly positive reviews (e.g., "Excellent direction, clever plot and gripping story"). Combining integrated gradients with high λ_{max} examples can yield insights into NLP models' fragility.

Several interesting differences emerge within the four models: CNN, Fasttext and LSTMs, as a consequence of being trained on less amount of data are fragile to substitutions like Noun substitutions. On the other hand, BERT models are robust to these sorts of substitutions because of their pretraining. High FIM BERT examples are more robust to meaningless changes, but single word substitutions still cause classifier prediction to change compared to low λ_{max} examples.

3.5 Discussion and Conclusion

As evident from our experiments above, λ_{max} is correlated with susceptibility to linguistic perturbations. Furthermore, λ_{max} is directly correlated to the minimum perturbation needed to flip the classifier prediction. Finally, we show a stark difference in the response of low and high λ_{max} examples, with the high λ_{max} examples, perturbed examples being susceptible to meaningless perturbations (e.g., noun/synonym substitutions).

These experiments have several interesting ramifications: Firstly, it's risky to over rely on accuracy while studying the generalizability of NLP classifiers. Most of the classifiers we used in our experiments attain really high accuracy on the test set despite failing simple sanity checks (e.g., invariance to synonym substitutions). It's thus important to identify the examples which are most susceptible to perturbations and test their resilience. The Fisher information provides us with a theoretically motivated framework to address this issue. By extracting the high λ_{max} examples and perturbing them by using glove based synonym substitutions, we can construct a rolling test set for our NLP classifiers. Furthermore, by mining only the high λ_{max} examples, FIM provides NLP practitioners with an interactive way to perturb and explore the flaws of their classifiers, something which would be extremely tedious to do on the entire dataset otherwise.

In this chapter, we introduced a method to discover the highly fragile examples for an NLP classifier. our method discovered in several state of the art classifiers like BERT, CNN and LSTM. Furthermore, our experiments shed some light on the links between geometry of NLP classifiers and their linguistic and embedding space perturbability. Our method provides NLP practitioners with both an automated an interactive human in the loop framework to better understand their models.

There are several interesting extensions. Firstly, it would be interesting to decode the optimal perturbation vector e_{max} associated with λ_{max} as a sentence. We are developing several approaches based on finding token substitutions which maximize this dot product with e_{max} in order to address this. It will also be interesting to study the link between purely geometrical properties like distance to the decision boundary and linguistic resilience. Finally, we are also interested in measuring the norm of token substitutions in embedding space to understand the relationship between the two.

Chapter 4

DEPLOYMENT CHALLENGES

4.1 Introduction

Fine-tuning pre-trained large language models (PLMs) in downstream tasks in Natural Language Processing (NLP) is the standard approach for achieving state-of-the-art results. This approach involves fine-tuning all the weights of the models (often billions of parameters) and is sample inefficient and unstable [49, 50]. Moreover, each task requires fine-tuning a separate model, which causes significant storage overheads and deployment costs and hinders adaptability to real-world scenarios. In order to address this parameter-inefficiency limitation, recently, there has been a significant number of approaches to improve fine-tuning the minimum set of parameters to achieve performance on par or better than full fine-tuning methods. The most prominent early approach, adapters [3], involves adding task-specific modules between layers of pre-trained networks. Since then, this approach has been adapted to other task-specific modules like only adapting the bias terms [51], or projection matrices [52]. [52] showed that fine-tuning by an arbitrary random linear projection into a smaller subspace could often be at par with full fine-tuning performance. The approach of using adapters is efficient as it exploits the over-parameterizations but fails to account for the intrinsic low-rank subspace dimensionality of the intermediate layers. In this work, we want to focus on extreme parameter reduction by using tensorized layers as a drop-in replacement to adapter-based approaches. This approach has two key benefits: Efficient tensorization of the pre-trained model varies based on network architectures and is often not transferrable across architectures. However, by only using tensorized

layers as a drop-in replacement of Adapter layers, this compression method can be used in any pre-trained model. Secondly, tensorization of layers naturally provides regularization and helps fine-tuning for downstream tasks, where the number of points is orders of magnitude smaller than the number of fine-tuned parameters.

Tensors can be viewed as multilinear mappings (i.e., functions), which are higher-order generalizations of linear mappings represented by matrices. Essential components of modern deep learning like convolutions and attentions naturally fit the notion of tensor mappings. Tensor methods help mitigate the curse of dimensionality without discarding the data's or model's original structure. Tensorized Layers (CP, Tucker, TensorTrain, and TensorRing) help reduce the number of parameters and improves generalization across tasks and domains [53,54]. [55] discuss tensor-based approaches in modern statistical learning and [56] specifically discuss the applications in computer vision and deep learning. Tensors are particularly important in understanding the theoretical aspects. Recent work has shown that tensor ranks can serve as a measure of predictor complexity and understand implicit regularization in deep learning [57].

Tensorized decompositions have been used to compress the weights of the neural network layers and improve the speed of inference [58–60]. A natural extension to using adapters and exploiting the low-rank structure of the weight matrices is to analyze the performance of fine-tuning with tensorization of adapter layers. Since adapter layers are already parameter efficient, tensorization will reduce the parameters further and naturally address the problems stemming from the curse of dimensionality. At its core, tensorization replaces a neural network layer with an approximate and structured low-rank form. This "parametric" form, i.e it's *shape* determines design trade-offs between storage capacity, accuracy and parameter efficiency. Furthermore, tensorized layers can be fine-tuned for a given task as a drop-in replacement for many layers like linear layers [61], embedding layers [62] and convolutional layers [59], by

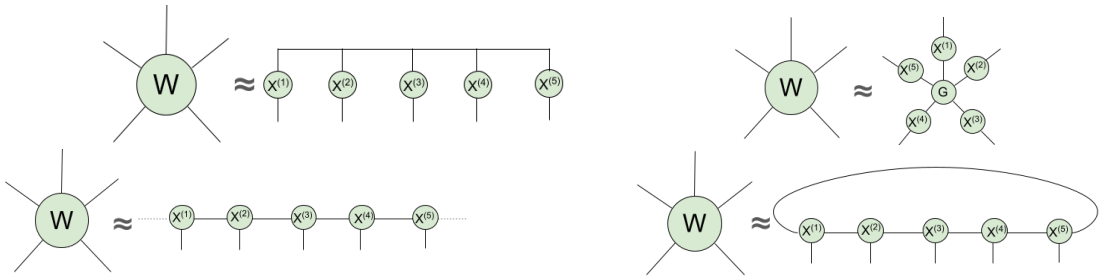

Figure 4.1: Tensor Network Diagrams of Tensor Decomposition Approaches. CP decomposition (Top Left), Tucker Decomposition (Top Right), Tensor Train Decomposition (Bottom Left) and Tensor Ring Decomposition (Bottom Right)

learning the projection factors by back-propagation along with the rest of the network parameters.

Tensorized layers add regularization ability while being parameter efficient because projection factors naturally deal with the curse of dimensionality. [63] explored tensorizing pre-trained language models, but efficient approaches to tensorizations are non-trivial. For example, [64] showed that merge ordering of the decomposition factors could determine the computational complexity for TensorRing Decomposition. Unlike matrices, tensors can be decomposed in different ways giving rise to different notions of rank. For most tensor learning problems, deciding the tensorization and the decomposition is non-trivial. It is also non-trivial to determine the tradeoff between minimizing the number of parameters and minimizing the corresponding loss function [65]. However, while previous work has proposed different variations of adapter modules [52, 66], tensorized layers have not been used to explore the parameter efficiency and downstream performance improvement of the models. In recent work, Compacter [66] used low-rank matrices for parameter efficiency in the adapter layers. However, unlike matrices, tensors have many representative properties that likely lead to robustness. Firstly, it has been shown that tensor-tensor representation of

an equal dimensional spanning space is superior to its matrix counterpart regarding representation efficiency [67]. TT-ranks provide much better flexibility than the matrix rank when applied at the same compression level.

Current methods that use parameter efficient approaches for fine-tuning PLMs have often shown robustness with fewer parameters. While these isolated approaches like using the bias terms [51], adapters [3], and Compacters [66] have shown some amount of performance improvements, the success of the robustness aspect remains somewhat mysterious. Instead of proposing adaptations to internal layer structure, we empirically study the impact of tensors as adapter layers. The advantage of tensors is many folds. Theoretical guarantees around uniqueness, low-rank structures, and regularization abilities make tensors appropriate for analysis across various structures. Since it has already been shown that adapter layers are efficient for downstream language tasks, this paper aims to isolate the regularization ability of adapter layers with tensor-based methods. We do a detailed comparison of different tensor decomposition methods to understand the impact on performance. Since it is non-trivial to decide the decomposition approach and the rank, we do an empirical study across ranks and different decomposition methods. We notice that tensor-based decompositions are more stable than matrix decompositions for adapter layers.

The goal is to systematically evaluate the following questions across a variety of datasets and models.

- What is the impact of rank in the different tensor compression approaches on the performance metric?
- How does the parameter efficiency change as the number of layers in transformer models increases?
- Is there a significant performance difference between the different tensor decomposition-

based approaches for compression?

- Use tensor based approach to deploy IQA question classification model to reduce cost of model inference without increasing speed of inference.

It is noted that several recent works focus on network compression using tensors. For example, [66] proposes a new architecture based on low-rank matrices. [68] directly uses matrix factorization for compressing pre-trained language models. Tensor decomposition naturally guarantees stable convergence properties and is superior to the matrix counterparts in terms of representative efficiency in terms of the number of parameters [67]. Secondly, the other line of work involves tensorizing various components of the pre-trained language model [69]. Since this is architecture-specific, we aim to only isolate the impact of performance in terms of rank and parameter efficiency by using tensor decomposition in the adapter layers. This generalizes to other multi-layer architectures with no modification (See implementation details in Section 5.1). Also, unlike previous work, which highlights isolated compression approaches, we focus on a detailed empirical study of the different compression approaches and the impact of rank in the compression.

We summarize key findings from the tensor-based compression approaches as a generalized adapter module.

- Tensor-based compression approaches are more stable than the corresponding matrix-based compression approaches.

- With increasing rank (more parameters), accuracy improves. However, the improvement is marginal and low ranks like (2,4) often provide an optimal balance of the number of parameters and performance.

- Unlike past work in compression that often shows one decomposition is significantly superior to the other, we notice that with the increasing number of layers, there is minimal difference in the performance of the different decomposition algorithms. Thus the decomposition approach can be chosen based on tradeoffs like compute efficiency or inference speed.

4.2 Related Work

Tensors have also been used to understand theoretical properties of deep learning like expressivity [70] and generalizability [71]. Compression of deep networks using tensor decomposition has been the primary intersection point of tensors and deep learning. The low-rank structure of the weight matrices means that compressing them without a significant loss in metrics like accuracy will improve inference time and enable deployment in resource-constrained hardware. A large body of this research has focussed on compressing convolutional neural networks (CNNs). These include [72, 73] who proposed low-rank approximations of the convolutional neural networks. However, since it was not entirely clear which decomposition is better for the different kinds of architectures, [74] proposed an approach specific to CNNs. Following the previous work, [75] introduced a tensors factorization framework for efficient multi-dimensional convolutions of higher-order CNNs for emotion estimation.

However, the biggest gain in most of these architectures comes from compressing the fully connected layers since this is often the bottleneck layer (generally contains the largest number of parameters compared to the other layers of the network). Also, the fully connected layer is flattened after the convolutional layers or the transformer layers before classification or generation tasks, thus losing the multimodal information and the hierarchical structure it may have learned. In fact, for the well-known VGG-19 network [76] 80% of the parameters come from the fully connected layer.

The popular approaches that have been used for compressing the fully connected layer are Tucker Decomposition by [77], Tensor Train decomposition by [78] and Block Term Decomposition by [79]. All these show the possibilities in deep learning compression algorithms. Since Transformer is the dominant architecture in modern deep learning algorithms, an in-depth study of the compressibility of transformers has been a recent focus of modern approaches. Some known approaches are quantization, where 32 bit parameters are replaced with a binary parameters (which can compress upto 32 times) [80, 81], knowledge distillation where a new model is trained from scratch [1], or approaches that are computationally heavy like the neural architecture search [82]. Since we use adapter layers to fine-tune the network on downstream tasks, this approach is generalizable like adapters while being parameter efficient.

Transformers have been compressed using Matrix [83] and Tensor Based decomposition methods [84]. [84] showed the promise of designing efficient transformer architectures, and [85] showed how Tensor Train architectures could be used to fine-tune the language model for compression simultaneously. The most recent work in this area [63] also focuses on compressing pre-trained transformer models using Tensor-based methods. However, none of these works compare the different tensor decomposition approaches; thus, comparing and evaluating the tensor decomposition approaches and the impact of tensor rank on parameter efficiency and performance during fine-tuning is difficult.

4.3 Adapter Compression Approach

Parameterizing fully connected layers using tensor decompositions was first proposed by [58]. The weights of fully-connected layers are represented as matrices, and tensor decompositions cannot be applied directly. Therefore, the weight matrix needs to be reshaped to obtain a higher-order tensor. The appropriate shape for reconstruc-

tion is still an open question and is non-trivial. Decisions about decompositions can also determine the computational complexity and efficiency of constructing the weight matrix from the decomposition factors.

Specifically, consider an input matrix $\mathbf{W}$ of size $I \times J$, whose dimensions can be expressed as $I = I_1 \times I_2 \times \cdots \times I_N$ and $J = J_1 \times J_2 \times \cdots \times J_N$. Hence, $\mathbf{W}$ is tensorized first by reshaping it to a higher-order tensor of size $I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_N$. Next, by permuting the dimensions and reshaping it again, an N^{th} order tensor of $I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N$ is obtained. This tensor is then compressed using any of the tensor decomposition methods as shown in Figure 4.1. The approximated weights are reconstructed from that low-rank factorization during inference and are then reshaped back into a matrix. A detailed comparison of the decomposition and corresponding factors can be found in Table 4.1.

4.4 Gradient Updates

A common approach to reducing parameters in transformer networks is to add intermediate layers within a transformer network and freezing the rest of the network. Thus the newly added intermediate layers are part of the gradient updates. The update equation is as follows.

$$\theta_{new} = \psi(\theta_{old})$$

In the fine-tuning stage, adding intermediate layers increases the number of computations to produce the final output. The added layers in the case of Adapters [3] are multiple fully connected layers, consequently leading to a significant increase in new matrix multiplications. This approach consequently leads to an increase in inference times. While speeding up this process could be achieved through other optimization, like int-8 quantization, we have left that for future work.

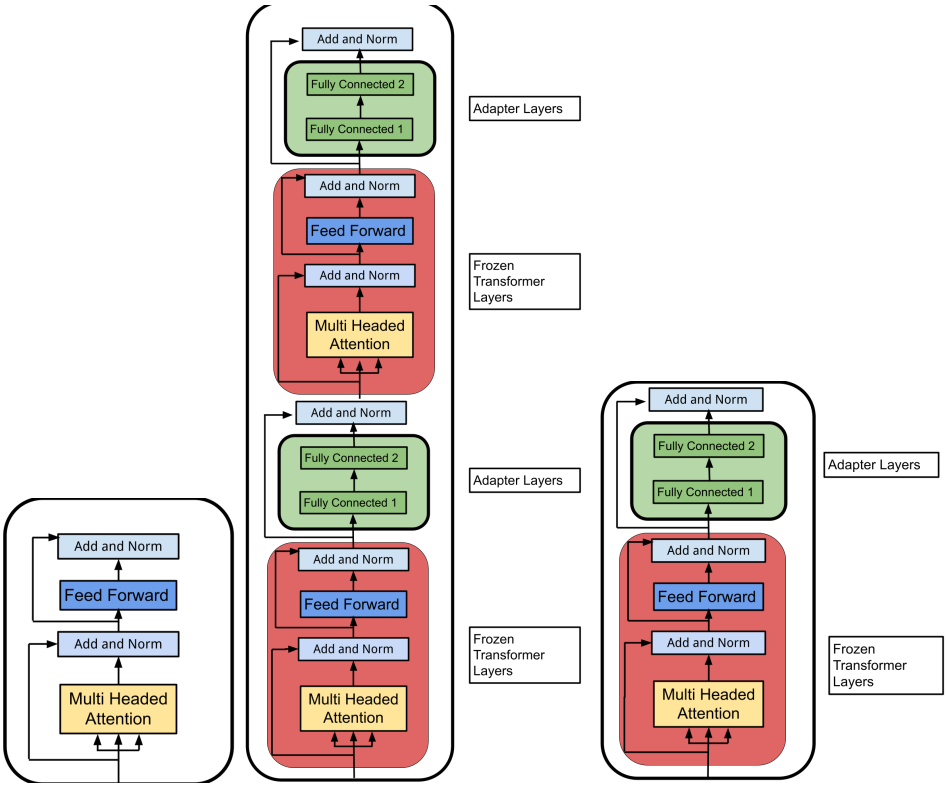

Figure 4.2: **Existing Work:** (Left) Original Transformer [2], (Center) Adapters [3], (Right) Parameter Efficient Adapters [4]

A second challenge of this approach is that these fully connected layers need to be added to the network, and this approach is non-trivial. The adapters have to serve as bottleneck layer to facilitate the gradient transfer from other layers but also needs to be updatable in the fine-tuning stage.

In order to address some of the challenges we proposed the following adaptations.

$$\theta_{new} = \theta_{old} + \phi$$

ϕ is **only** getting updated during the fine-tuning stage. This prevents added matrix multiplications, and it is only matrix additions. Additions are significantly faster than matrix multiplications.

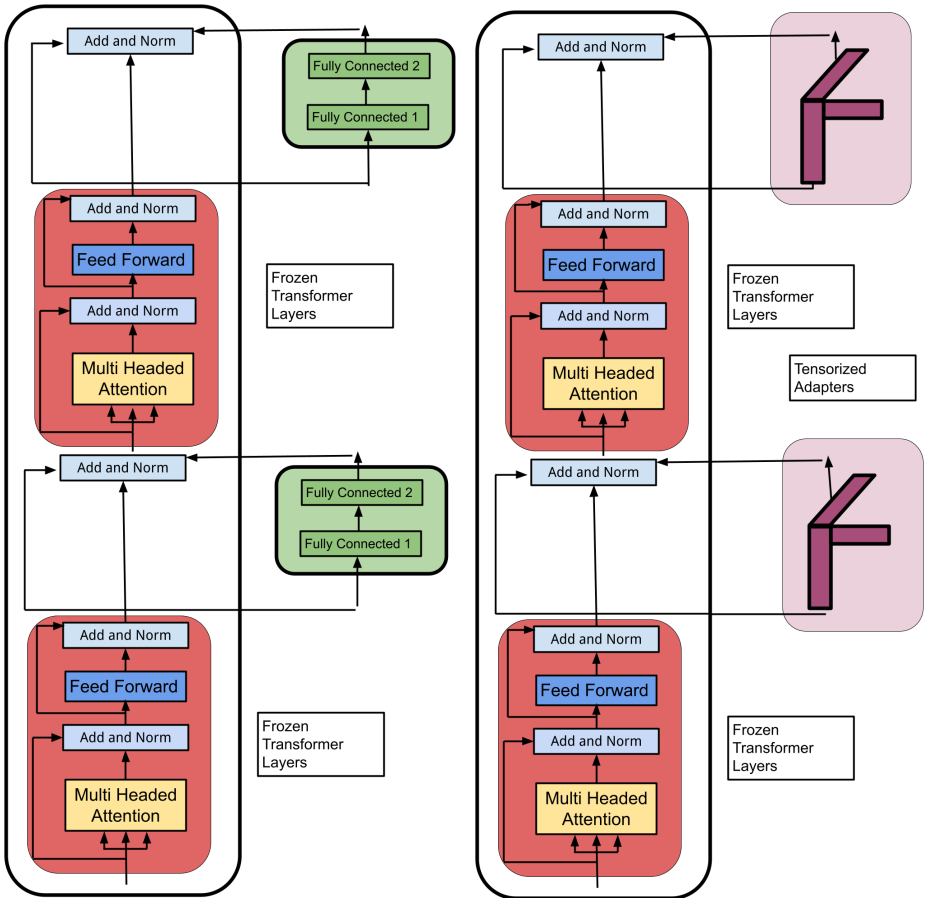

Figure 4.3: **Our Work** Left: Adding Adaptation Layers (Right) Tensorized Adapters (Adding tensorized networks improve model efficiency while adding regularization.)

A second advantage of this approach is that because the computations are additions, arbitrary and complex computations can be redistributed to a different device. In this context, ϕ, could be any function that can be computed in parallel in a different device. This parallel computation will not lead to an increase in inference times.

4.5 Adaptation of tensor based approach to improve inference speed

Adapters [3] and the family of parameter efficient training had one critical drawback. While the number of parameters that were fine-tuned was less, it did not lead to decrease in inference speed because the model parameters were all used during inference.

(In other words, the number of matrix multiplications increased because of the added parameters and all the components needed to reside in a Graphical Processing Unit. In essence the update equation changes from:

$$\theta_{new} = \psi(\theta_{old})$$

to

$$\theta_{new} = \theta_{old} + \phi$$

If θ is mXn, $\psi(\theta_{old})$ is also mXn. In adapters, new layers are added inside the transformers. While this method is efficient in solving multiple tasks, by design this is architecture specific and different models require different adaptations to be able to solve multiple tasks simultaneously. This approach also meant different parts of the network needed different kinds of tensor compression methods and embedding layers (which often is very parameter heavy) could not be compressed simultaneously across architectures.

In both the adaptations, our goal is to be able to approximate the tensors in the weight matrix. In the second approach, the gradients are independently computed for the parameters and thus the inference speed is not affected since the matrix multiplication for the adaptation part 4.3 happens parallel to the forward network computation of the original transformer. This approach is also thus agnostic to newer architectures of transformers as long as there are independent sub-component of the transformer blocks.

Instead of using only fully connected layers, (left figure of Figure 4.3), we use tensor decomposition based methods for the gradient updates for both approaches.

Thus finally the forward pass changes from:

$$H_l = W_l x + \Delta W x$$

to:

$$H_l = W_l x + \mathcal{T} x$$

Here $\mathcal{T}$ is any representation of Tensor Decomposition Based Approach which is discussed in the following section.

In Figure 4.3, we used CP decomposition and thus $\mathcal{T}$ can be represented using the following equation:

$$\mathcal{T} = \sum_{\alpha=1}^{r} \mathbf{u}_\alpha^{(1)} \circ \cdots \circ \mathbf{u}_\alpha^{(d)}, \tag{4.1}$$

4.6 Tensor Decomposition Methods

The general approach behind all the tensor decomposition is to use the weight matrix tensor and one of the following tensor decomposition algorithms for Compression. Figure 4.1 shows the tensor network diagram of each decomposition.

4.6.1 CP decomposition

The CPD or cannonical polyadic decomposition [86] aims to represent a dth-order tensor $\mathcal{T}$, in our case it is the weight matrix, by a sum of rank-one tensors. Determining the number of rank-1 components is an NP-hard problem.

$$\mathcal{T} = \sum_{\alpha=1}^{r} \mathbf{u}_\alpha^{(1)} \circ \cdots \circ \mathbf{u}_\alpha^{(d)}, \tag{4.2}$$

where each rank-1 Tensor is represented by an outer product of d vectors. It can be also written in the element-wise form given by

$$T(i_1, \ldots, i_d) = \left\langle \mathbf{u}_{i_1}^{(1)}, \ldots, \mathbf{u}_{i_d}^{(d)} \right\rangle, \tag{4.3}$$

where $\langle \cdot, \ldots, \cdot \rangle$ denotes an inner product of a set of vectors, i.e., $\mathbf{u}_{i_k}^{(k)} \in \mathbb{R}^r, k = 1, \ldots, d$.

4.6.2 Tucker decomposition

The Tucker decomposition [87] aims to represent a dth-order tensor $\boldsymbol{\mathcal{T}}$ by a multilinear product between a core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ and factor matrices $\mathbf{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}, k = 1, \ldots, d$, which is expressed by

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}^{(1)} \times_2 \cdots \times_d \mathbf{U}^{(d)} = [\![\boldsymbol{\mathcal{G}}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}]\!]. \tag{4.4}$$

4.6.3 TT decomposition

The tensor train decomposition [88] aims to represent a dth-order tensor $\boldsymbol{\mathcal{T}}$ by a sequence of cores $\boldsymbol{\mathcal{G}}_k, k = 1, \ldots, d$, where the first core $\mathbf{G}_1 \in \mathbb{R}^{n_1 \times r_2}$ and the last core $\mathbf{G}_d \in \mathbb{R}^{r_d \times n_d}$ are matrices while the other cores $\boldsymbol{\mathcal{G}}_k \in \mathbb{R}^{r_k \times n_k \times r_{k+1}}, k = 2, \ldots, d-1$ are 3rd-order tensors. Specifically, TT decomposition in the element-wise form is expressed as

$$T(i_1, \ldots, i_d) = \mathbf{g}_1(i_1)^T \mathbf{G}_2(i_2) \cdots \mathbf{G}_{d-1}(i_{d-1}) \mathbf{g}_d(i_d), \tag{4.5}$$

where $\mathbf{g}_1(i_1)$ is the i_1th row vector of $\mathbf{G}_1$, $\mathbf{g}_d(i_d)$ is the i_dth column vector of $\mathbf{G}_d$, and $\mathbf{G}_k(i_k), k = 2, \ldots, d-1$ are the i_kth lateral slice matrices of $\boldsymbol{\mathcal{G}}_k$.

4.6.4 Tensor Ring Model

The Tensor Ring decomposition [89] is very similar to the Tensor Train decomposition except that it represents a higher order tensor by a sequence of 3rd-order tensors that are multiplied circularly. Specifically, let $\boldsymbol{\mathcal{T}}$ be a dth-order tensor of size $n_1 \times n_2 \times \cdots \times n_d$, denoted by $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, TR representation is to decompose it into a sequence of latent tensors $\boldsymbol{\mathcal{Z}}_k \in \mathbb{R}^{r_k \times n_k \times r_{k+1}}, k = 1, 2, \ldots, d$, which can be expressed in

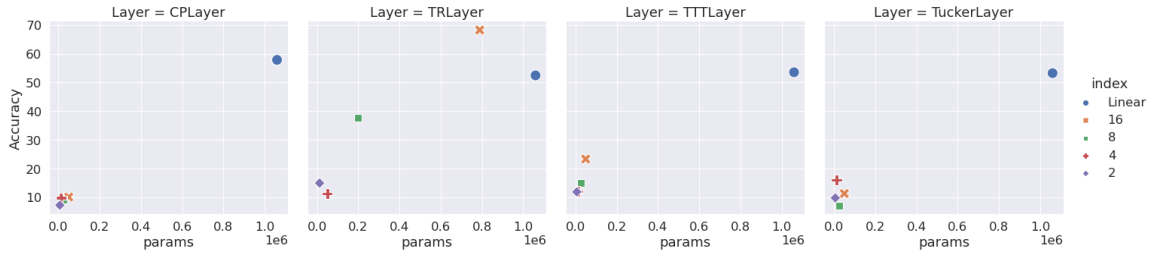

Figure 4.4: Performance of the 4 types of Tensor Decomposition Models using the MNIST dataset. The network is a 2-layered, fully connected network with dimensions (1024, 1024) and (1024, 10). The experiment's goal was to understand and evaluate the impact of tensor decomposition algorithms on simple feedforward networks.

an element-wise form given by

$$\begin{aligned} T(i_1, i_2, \ldots, i_d) &= \mathrm{Tr}\left\{\mathbf{Z}_1(i_1)\mathbf{Z}_2(i_2)\cdots\mathbf{Z}_d(i_d)\right\}, \\ &= \mathrm{Tr}\left\{\prod_{k=1}^{d}\mathbf{Z}_k(i_k)\right\}. \end{aligned} \tag{4.6}$$

$T(i_1, i_2, \ldots, i_d)$ denotes $(i_1, i_2, \ldots, i_d)$th element of the tensor. $\mathbf{Z}_k(i_k)$ denotes the i_kth lateral slice matrix of the latent tensor $\boldsymbol{\mathcal{Z}}_k$, which is of size $r_k \times r_{k+1}$. Note that any two adjacent latent tensors, $\boldsymbol{\mathcal{Z}}_k$ and $\boldsymbol{\mathcal{Z}}_{k+1}$, have an equivalent dimension r_{k+1} on their corresponding mode. An important property of Tensor Ring decomposition is that the last latent tensor $\boldsymbol{\mathcal{Z}}_d$ is of size $r_d \times n_d \times r_1$, i.e., $r_{d+1} = r_1$. This fact makes it different from the Tensor Train decomposition and ensures that the product of these matrices is a square matrix resulting in numerous important numeric properties of Tensor Ring Decomposition. For simplicity, TR decomposition in the tensor form, can also be expressed as:

$$\boldsymbol{\mathcal{T}} = \sum_{\alpha_1,\ldots,\alpha_d=1}^{r_1,\ldots,r_d} \mathbf{z}_1(\alpha_1, \alpha_2) \circ \mathbf{z}_2(\alpha_2, \alpha_3) \circ \cdots \circ \mathbf{z}_d(\alpha_d, \alpha_1), \tag{4.7}$$

	Type	Rank	Factors	TotalParams	HidLayer	NumLayer	Compression
0	TR	4	[(4, 24, 4), (4, 32, 4), (4, 24, 4), (4, 32, 4...	21504	768	12	329.143 x
1	TR	8	[(8, 24, 8), (8, 32, 8), (8, 24, 8), (8, 32, 8...	86016	768	12	82.2857 x
2	TT	4	[(1, 24, 4), (4, 32, 4), (4, 24, 4), (4, 32, 1...	13440	768	12	526.629 x
3	TT	8	[(1, 24, 8), (8, 32, 8), (8, 24, 8), (8, 32, 1...	48384	768	12	146.286 x
4	CP	4	[(24, 4), (32, 4), (24, 4), (32, 4)] * 12	5376	768	12	1316.57 x
5	CP	8	[(24, 8), (32, 8), (24, 8), (32, 8)] * 12	10752	768	12	658.286 x
6	Tucker	4	[(24, 4), (32, 4), (24, 4), (32, 4)] * 12	5376	768	12	1316.57 x
7	Tucker	8	[(24, 8), (32, 8), (24, 8), (32, 8)] * 12	10752	768	12	658.286 x

Table 4.1: Comparison of different tensor decomposition approaches for ranks 4 and 8 compared to a linear layer, a key component of the adapter layer.

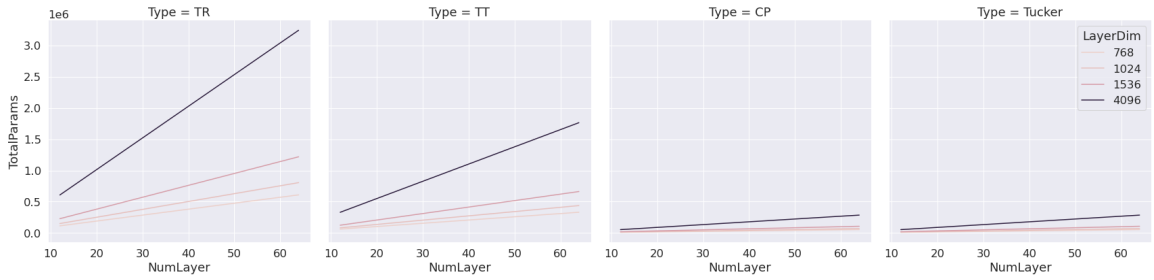

Figure 4.5: The number of parameters grows with increasing layer dimensions and the number of layers. Since modern transformer architectures grow significantly in terms of the number of layers and the layer dimensions, TensorTrain and Tensor Ring models grow faster than CP and Tucker decompositions.

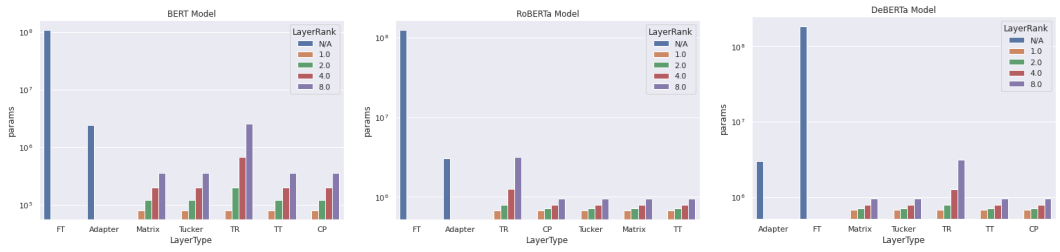

Figure 4.6: Comparison of the number of parameters for 3 models (BERT, RoBERTa, and DeBERTa) with fully fine-tuned and adapter parameters for the STSB task in the GLUE Benchmark [5].

LayerType	**Model**	**Rank**	A/MRPC	A/RTE	S/STSB	F1/QQP	Params
Adapter	bert	-	0.884	0.695	0.787	0.720	2.42M
	deberta-v2	-	0.884	0.684	0.854	0.715	3.01M
	roberta	-	0.884	0.637	0.793	0.713	3.01M
TensorRing	bert	2.0	0.881	0.677	0.754	0.712	63K
		4.0	0.881	0.673	0.749	0.719	132K
	deberta-v2	2.0	0.884	0.681	0.764	0.716	655K
		4.0	0.884	0.684	0.801	0.711	724K
	roberta	2.0	0.884	0.645	0.757	0.715	654K
		4.0	0.884	0.619	0.724	0.716	723K
TensorTrain	bert	2.0	0.881	0.677	0.754	0.712	54.1K
		4.0	0.881	0.673	0.750	0.709	79.1K
	deberta-v2	2.0	0.884	0.681	0.799	0.711	646K
		4.0	0.884	0.684	0.829	0.715	671K
	roberta	2.0	0.884	0.645	0.703	0.713	645K
		4.0	0.884	0.612	0.797	0.720	670K
CP	bert	2.0	0.881	0.677	0.750	0.705	51.6K
		4.0	0.881	0.673	0.751	0.709	63.2K
	deberta-v2	2.0	0.884	0.681	0.769	0.721	644K
		4.0	0.884	0.684	0.733	0.717	655K
	roberta	2.0	0.884	0.648	0.758	0.722	642K
		4.0	0.884	0.616	0.759	0.718	654K
Tucker	bert	2.0	0.881	0.677	0.749	0.697	51.8K
		4.0	0.881	0.673	0.748	0.717	66K
	deberta-v2	2.0	0.884	0.681	0.783	0.713	644K
		4.0	0.884	0.684	0.810	0.706	658K
	roberta	2.0	0.884	0.645	0.804	0.718	642K
		4.0	0.884	0.616	0.788	0.717	657K

Table 4.2: Metrics on the GLUE benchmark for the different decomposition algorithms and ranks. No decomposition-based approach is significantly superior to the other. Even though many recent works relied on Tensor Train decomposition for most of the experiments, in this dataset, we show that both CP and Tucker Decomposition perform equally well compared to Tensor Train and Tensor Ring decomposition.

4.7 Experiments

In order to illustrate the difference in accuracy, a simple two-layer neural network is used for MNIST classification as a toy example. The goal was to understand the impact of tensorization and regularization as a substitute for the linear layer. We used MNIST instead of language tasks because language networks always have an embedding layer, and it is difficult to isolate the impact of the fully connected layer. Figure 4.4 shows that tensor ring decomposition as a drop-in replacement to linear layer performs better than the other decomposition-based approaches. Even though [64] had shown that Tensor Ring decomposition is more expressive than TT for the same intermediate rank, this performance gain disappears when multiple adapters are used for compression in Transformer based architectures Table ??.

All experiments were performed in A6000 GPU from NVIDIA, which has a 48GB GDDR6 RAM. The experiments used consistent random seeds set using both the numpy and the torch options. The models and the datasets were loaded from the Huggingface Transformers library [90] and the Huggingface Datasets library [91]. Since it is difficult to fine-tune models with different network architectures, we fixed the number of epochs for fine-tuning to 3 as recommended in the official evaluation scripts for the GLUE [5] tasks in the library. This means, technically, the actual performance could be sub-optimal; however, it enables fair comparison between each of the Tensor Decomposition Algorithms when trained similarly. Three models were used for the experiments BERT [21], RoBERTa [92] and DeBERTa [93] and DistilBERT [1] model. Details of the hyper-parameters and the runs can be found in the Appendix.

4.7.1 Implementation Details

Each adapter layer was added after a *SelfOutput* Layer. The Huggingface transformers library [94] was used, and a detailed hyper-parameter description can be found in the Appendix. Each adapter layer is composed of a down-projection layer and an up-projection layer which are generally linear layers of shape *(hidden dimension, adapter dimension)* and *(adapter dimension, hidden dimension)*. We used the same hyper-parameters across all the tensor decomposition algorithms, and the models are trained based on the official transformer library. The datasets library was used for the datasets used in the experiment [95]. The results reported are the evaluation accuracy.

4.7.2 Models

BERT Base

BERT [21] is a pretrained on the English language using the masked language modeling objective and the next sentence prediction objective. The model used in our experiment is 'BERT-base-uncased,' which does not distinguish between upper case and lower case letters. The training paradigm relied on self-supervision, essentially eliminating the need for using labeled datasets for training. The original model was trained on four cloud TPU (Tensor Processing Units) for one million steps with a batch size of 256. The adam optimizer was used, and details of the exact hyperparameters can be found in the original paper.

RoBERTa Base

RoBERTa [92] was originally a replication study of BERT and showed significant improvements over BERT by only altering the hyperparameters. The conclusion from RoBERTa was that BERT was significantly undertrained and could match or even

exceed the performance of BERT in many tasks. However, since the community still heavily relies on BERT for all benchamrks we use both BERT and RoBERTa for fair performance comparison.

Since [51] showed that trends between the *base* and the *large* models consistently remained the same, we used the *base* models for all our experiments.

DeBERTa-V2

This model is an improvement upon BERT and RoBERTa. Disentangled Attention helps provide richer information for each word, and an enhanced mask decoder is used to incorporate absolute positions in the decoding layer. Compared to RoBERTa large, DeBERTa was trained on half the training data and performed consistently better on the GLUE benchmark.

The DistilBERT

[1] model is a faster and a smaller version of the BERT model that uses the same training corpus but uses teacher-student distillation. Here the BERT base model was the teacher and in addition to the BERT loss functions it has a cosine embedding loss and distillation loss. Since the model is smaller than BERT this results in faster inference without a significant decrease in accuracy.

4.7.3 Datasets

We evaluate the tensor decomposition algorithms on the subset of the GLUE [5] tasks similar to other compression papers. The performance across datasets for compression remained consistent in [3] and [51]; thus, the subset was chosen based on the training time required. Some datasets like WNLI were excluded since BERT models do not outperform the majority baseline. Also, since the QNLI test set

Table 4.3: Teacher question categories and their alignment with terminology used in other frameworks

Model Name	**Accuracy**	**Runtime**	**Hidden Size**	**Vocab Size**
bert-base	0.75	6.3532	1024	50272
roberta-base	0.76	5.6464	768	50265
microsoft/deberta-v3-base	0.75	7.8258	768	128100
distilbert-base-uncased	0.74	1.5125	768	30522

benchmark has been updated since the original release, it has not been included for comparison.

4.7.4 Performance for IQA classification

The tensorized adapter approach is significant since it is an efficient approach to deploy Large Language Models in low resource scenarios. Our tensorized adapters as was shown in table ?? performed at par with existing fully connected approach. We evaluated it on the IQA question classification task and also noted the runtimes below. Because of the efficient distribution of adapter computation and the network computation this approach can be generalized to any transformer based models for language tasks.

4.8 Parameter Growth, Rank and Accuracy

As the rank increases, the number of parameters increases, and the performance metrics should naturally improve. However, lower ranks act as regularizers and often lead to performance improvements, as seen in some decompositions. Also, unlike previous approaches that have consistently shown the advantages of TensorTrain

and TensorRing [64] based decompositions, the performance improvements observed through Tucker and CP decompositions are comparable ??. As the number of layers increase, the difference between the decomposition performances reduce. However, it is worth noting that Tensor Ring, followed by Tensor Train decompositions, increases the number of parameters with layers (Figure: 4.5. However, there is minimal difference in accuracy between these approaches. The parameter growth can also be seen in Figure 4.6 for the STSB task in the GLUE Benchmark [5].

4.9 Conclusion

We have shown results of tensor decomposition techniques that could often benefit the fine-tuning process in traditional transformers. Our goal was to address previous approaches to transformer compression through Tensor methods by tensorized transformer layers. We proposed a simple adapter replacement to compress transformers and transfer across multiple tasks. Existing work [3, 4] focussed on reducing the task specific parameters they increased the speed of inference. Our work keeps the inference speed constant by adding the tensorized adapters parallel to the transformer layer. We also empirically studied the different decomposition algorithms and the ranks of the decomposition.

We showed an efficient approach to compress transformer layers (both embedding and self attention layers) by splitting up the gradient computation of the forward pass. This work has significant potential to reducing carbon footprint of the models without any increase in inference speed. This also opens up approaches in the future to improve upon existing multi-task learning approaches by using tensorized adapters for each task. We demonstrate the memory efficiency of our approach by deploying it in a real world use case which cost 5x lesser than traditional deployment approaches.

Our goal is to open-source our implementations of all the code for the decompo-

sitions. It is also worth highlighting that we use *torch.nn.Module* to implement the decomposition algorithms, thus enabling any multi-layer Transformer models to use it as a drop-in replacement of transformer layers.

Chapter 5

CONCLUSION

This dissertation explored various aspects of deep learning in natural language processing and its applications in low-resource domains like education and healthcare. Our work focused on data collection, evaluation of deep learning models, and efficient training and deployment of transformer-based models. We proposed novel approaches to address the challenges in these areas and demonstrated their effectiveness through experiments and evaluations.

We showed that model-assisted labeling can significantly improve the annotation speed, making data collection more efficient. Our approach to finding examples near the decision boundary of deep learning-based classifiers can help identify challenging cases and improve model performance. Our differential-geometry-based approach to evaluating deep learning models can identify examples that are most and least susceptible to small perturbations in input data, providing valuable insights into the model's behavior. Our approach highlighted the seemingly simple failure modes of large language models, and this approach can extend to other languages and datasets.

Finally, our tensorized adapter approach to training and deploying transformer-based models can reduce the number of tunable parameters without sacrificing performance, making these models more efficient and practical for educational scenarios.

There are several directions for future work based on the findings of this dissertation. First, our model-assisted data collection approach can be extended to other low-resource domains and different data types (images or time series data). Further research can investigate optimizing the model's performance using the collected data. Second, our differential-geometry-based approach to evaluating deep learning models

can be applied to other natural language processing tasks beyond classification, such as question-answering and machine translation. Finally, our tensorized adapter approach to training and deploying transformer-based models can be further optimized to improve efficiency and performance. Future research can explore using other tensor decomposition-based methods or hybrid techniques to reduce the parameters further while maintaining high accuracy. Additionally, the effectiveness of these approaches can be evaluated on larger models to assess their scalability and generalizability. Tensorized layers can also provide insights into the low-rank structure of intermediate-weight matrices, which in turn can help interpretability and efficient architecture design for larger models.

References

[1] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[3] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.

[4] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," *arXiv preprint arXiv:2007.07779*, 2020.

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

[6] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[7] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.

[8] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut Learning in Deep Neural Networks," *arXiv:2004.07780 [cs, q-bio]*, Apr. 2020, arXiv: 2004.07780. [Online]. Available: http://arxiv.org/abs/2004.07780

[9] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," *arXiv preprint arXiv:1603.08023*, 2016.

[10] S. E. Finch and J. D. Choi, "Towards unified dialogue system evaluation: A comprehensive analysis of current evaluation protocols," in *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st

virtual meeting: Association for Computational Linguistics, Jul. 2020, pp. 236–245. [Online]. Available: https://www.aclweb.org/anthology/2020.sigdial-1.29

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2016.

[12] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *arXiv preprint arXiv:1906.02530*, 2019.

[13] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[14] K. Shuster, S. Humeau, H. Hu, A. Bordes, and J. Weston, "Engaging image captioning via personality," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 516–12 526.

[15] M. Guan, V. Gulshan, A. Dai, and G. Hinton, "Who said what: Modeling individual labelers improves classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[16] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," *arXiv preprint arXiv:1705.10694*, 2017.

[17] V. U. Prabhu and A. Birhane, "Large image datasets: A pyrrhic win for computer vision?" *arXiv preprint arXiv:2006.16923*, 2020.

[18] K. Crawford and T. Paglen, "Excavating ai: the politics of images in machine learning training sets," *Excavating AI*, 2019.

[19] A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme, "Hypothesis only baselines in natural language inference," *arXiv preprint arXiv:1805.01042*, 2018.

[20] D. A. Norman, "Cognitive artifacts," *Designing interaction: Psychology at the human-computer interface*, vol. 1, no. 1, pp. 17–38, 1991.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[22] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[23] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," *arXiv preprint arXiv:2002.06470*, 2020.

[24] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

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

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

[27] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith, "Annotation Artifacts in Natural Language Inference Data," *arXiv:1803.02324 [cs]*, Apr. 2018, arXiv: 1803.02324. [Online]. Available: http://arxiv.org/abs/1803.02324

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

[29] T. Niven and H. Y. Kao, "Probing neural network comprehension of natural language arguments," in *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*. Association for Computational Linguistics (ACL), 2020, pp. 4658–4664.

[30] A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme, "Hypothesis Only Baselines in Natural Language Inference," in *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 180–191. [Online]. Available: https://www.aclweb.org/anthology/S18-2023

[31] M. Gardner, Y. Artzi, V. Basmova, J. Berant, B. Bogin, S. Chen, P. Dasigi, D. Dua, Y. Elazar, A. Gottumukkala, N. Gupta, H. Hajishirzi, G. Ilharco, D. Khashabi, K. Lin, J. Liu, N. F. Liu, P. Mulcaire, Q. Ning, S. Singh, N. A. Smith, S. Subramanian, R. Tsarfaty, E. Wallace, A. Zhang, and B. Zhou, "Evaluating NLP Models via Contrast Sets," *arXiv:2004.02709 [cs]*, Apr. 2020, arXiv: 2004.02709. [Online]. Available: http://arxiv.org/abs/2004.02709

[32] D. Kaushik, E. Hovy, and Z. C. Lipton, "Learning the difference that makes a difference with counterfactually-augmented data," *arXiv preprint arXiv:1909.12434*, 2019.

[33] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6904–6913.

[34] D. Kaushik and Z. C. Lipton, "How much reading does reading comprehension require? a critical investigation of popular benchmarks," *arXiv preprint arXiv:1808.04926*, 2018.

[35] R. T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," *arXiv preprint arXiv:1902.01007*, 2019.

[36] M. Glockner, V. Shwartz, and Y. Goldberg, "Breaking nli systems with sentences that require simple lexical inferences," *arXiv preprint arXiv:1805.02266*, 2018.

[37] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," in *Advances in Neural Information Processing Systems*, 2019, pp. 3261–3275.

[38] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, "Adversarial nli: A new benchmark for natural language understanding," *arXiv preprint arXiv:1910.14599*, 2019.

[39] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" *arXiv preprint arXiv:1902.10811*, 2019.

[40] S. Beery, G. Van Horn, and P. Perona, "Recognition in Terra Incognita," 2018, pp. 456–473. [Online]. Available: http://openaccess.thecvf.com/content_ECCV_2018/html/Beery_Recognition_in_Terra_ECCV_2018_paper.html

[41] R. Karakida, S. Akaho, and S.-i. Amari, "Universal statistics of fisher information in deep neural networks: Mean field approach," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1032–1041.

[42] S.-i. Amari, R. Karakida, and M. Oizumi, "Fisher information and natural gradient learning in random deep networks," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 694–702.

[43] C. Zhao, P. T. Fletcher, M. Yu, Y. Peng, G. Zhang, and C. Shen, "The adversarial attack and detection under the fisher information metric," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5869–5876.

[44] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[45] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

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

[47] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[48] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3319–3328.

[49] M. E. Peters, S. Ruder, and N. A. Smith, "To tune or not to tune? adapting pretrained representations to diverse tasks," *arXiv preprint arXiv:1903.05987*, 2019.

[50] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

[51] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[52] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.

[53] Y. Yang and T. Hospedales, "Deep multi-task representation learning: A tensor factorisation approach," *arXiv preprint arXiv:1605.06391*, 2016.

[54] A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "Incremental multi-domain learning with network latent tensor factorization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 470–10 477.

[55] W. W. Sun, B. Hao, and L. Li, "Tensors in modern statistical learning," *Wiley StatsRef: Statistics Reference Online*, pp. 1–25, 2014.

[56] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor methods in computer vision and deep learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

[57] N. Razin, A. Maman, and N. Cohen, "Implicit regularization in tensor factorization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8913–8924.

[58] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[59] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.

[60] M. Astrid and S.-I. Lee, "Cp-decomposition with tensor power method for convolutional neural networks compression," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 115–118.

[61] J. Kossaifi, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar, "Tensor contraction layers for parsimonious deep nets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 26–32.

[62] O. Hrinchuk, V. Khrulkov, L. Mirvakhabova, E. Orlova, and I. Oseledets, "Tensorized embedding layers for efficient model compression," *arXiv preprint arXiv:1901.10787*, 2019.

[63] B. Wang, Y. Ren, L. Shang, X. Jiang, and Q. Liu, "Exploring extreme parameter compression for pre-trained language models," in *International Conference on Learning Representations*, 2021.

[64] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, "Wide compression: Tensor ring nets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9329–9338.

[65] M. Hashemizadeh, M. Liu, J. Miller, and G. Rabusseau, "Adaptive learning of tensor network structures," *arXiv preprint arXiv:2008.05437*, 2020.

[66] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[67] M. E. Kilmer, L. Horesh, H. Avron, and E. Newman, "Tensor-tensor algebra for optimal representation and compression of multiway data," *Proceedings of the National Academy of Sciences*, vol. 118, no. 28, 2021.

[68] M. B. Noach and Y. Goldberg, "Compressing pre-trained language models by matrix decomposition," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 2020, pp. 884–889.

[69] Y. Ren, B. Wang, L. Shang, X. Jiang, and Q. Liu, "Exploring extreme parameter compression for pre-trained language models," *arXiv preprint arXiv:2205.10036*, 2022.

[70] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," *arXiv: Neural and Evolutionary Computing*, 2015.

[71] J. Li, Y. Sun, J. Su, T. Suzuki, and F. Huang, "Understanding generalization in deep learning via tensor methods," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 504–515.

[72] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *Advances in neural information processing systems*, vol. 27, 2014.

[73] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," *arXiv preprint arXiv:1511.06067*, 2015.

[74] K. Hayashi, T. Yamaguchi, Y. Sugawara, and S.-i. Maeda, "Exploring unexplored tensor network decompositions for convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[75] J. Kossaifi, A. Toisoul, A. Bulat, Y. Panagakis, T. M. Hospedales, and M. Pantic, "Factorized higher-order cnns with an application to spatio-temporal emotion estimation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6059–6068, 2020.

[76] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[77] J. Kossaifi, Z. C. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *ArXiv*, vol. abs/1707.08308, 2020.

[78] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *NIPS*, 2015.

[79] J. Ye, G. Li, D. Chen, H. Yang, S. Zhe, and Z. Xu, "Block-term tensor neural networks," *Neural networks : the official journal of the International Neural Network Society*, vol. 130, pp. 11–21, 2020.

[80] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu, "Ternarybert: Distillation-aware ultra-low bit bert," *arXiv preprint arXiv:2009.12812*, 2020.

[81] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. Lyu, and I. King, "Binarybert: Pushing the limit of bert quantization," *arXiv preprint arXiv:2012.15701*, 2020.

[82] C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou, "Bert-of-theseus: Compressing bert by progressive module replacing," *arXiv preprint arXiv:2002.02925*, 2020.

[83] M. B. Noach and Y. Goldberg, "Compressing pre-trained language models by matrix decomposition," in *AACL*, 2020.

[84] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song, "A tensorized transformer for language modeling," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[85] P. Liu, Z.-F. Gao, W. X. Zhao, Z. Y. Xie, Z.-Y. Lu, and J. rong Wen, "Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators," in *ACL*, 2021.

[86] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, pp. 283–319, 1970.

[87] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966.

[88] I. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, pp. 2295–2317, 2011.

[89] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," *ArXiv*, vol. abs/1606.05535, 2016.

[90] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[91] Q. Lhoest, A. Villanova del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall, J. Davison, M. Šaško, G. Chhablani, B. Malik, S. Brandeis, T. Le Scao, V. Sanh, C. Xu, N. Patry, A. McMillan-Major, P. Schmid, S. Gugger, C. Delangue, T. Matussière, L. Debut, S. Bekman, P. Cistac, T. Goehringer, V. Mustar, F. Lagunas, A. Rush, and T. Wolf, "Datasets: A community library for natural language processing," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 175–184. [Online]. Available: https://aclanthology.org/2021.emnlp-demo.21

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

[93] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.

[94] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[95] Q. Lhoest, A. V. del Moral, Y. Jernite, A. Thakur, P. von Platen, S. Patil, J. Chaumond, M. Drame, J. Plu, L. Tunstall *et al.*, "Datasets: A community library for natural language processing," *arXiv preprint arXiv:2109.02846*, 2021.