The Applications of Meta-learners to Few-Shot Incremental Learning

LIAM WATTS
SID: 510562348

Supervisor: Dr. Caren Han
Associate Supervisor: Dr Josiah Poon

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Bachelor of Advanced Studies (Honours)

School of Computer Science
The University of Sydney
Australia

12 December 2021

Student Plagiarism: Compliance Statement

Name: Liam Watts

Signature: Watts

Date: 21/11/2021

Abstract

Current SoTA deep learning models in natural language processing have increasingly large parameter counts in the hundreds of millions. These models require increasingly large datasets to train on, while pretraining has partially addressed this expensive human labelled datasets are still necessary for fine tuning. Few-Shot learning techniques such as Meta-learning aim to reduce this expense through developing models that learn from smaller datasets. However meta-learners are typically evaluated on simulated tasks far different to the real-world problems few-shot learning aims to solve. This thesis aims to apply meta-learning to the more real-world aligned problem of class incremental learning, where a model must learn to classify new classes introduced after training is completed. Meta-learners possess the unique property of generalisation to classes unseen in training through providing a few examples of them, making them especially useful for class incremental learning. We introduce a method of simulating class incremental learning using surrogate novel classes, which can allow a meta-learner to perform the task without retraining. To take advantage of this training process we devise a meta-learner with a learnable, transformer-based aggregation function that uses information from examples of all classes to make predictions. The key contributions of this model are to consider the entire support set and query together, in addition to weighting the attention to increase the impact important samples such as the query have on inference. Overall we find this model outperforms existing baselines and that most meta-learners generalise well to the class incremental learning problem, even when not specifically trained for it. This thesis outlines the practical applications meta-learners have in class incremental learning and creates a well performing baseline for future transformer-based aggregation methods to build upon.

Acknowledgements

I want to thank my excellent supervisor Dr. Caren Han for guiding me through the research process, helping narrow down the topic of the thesis, pointing me in the right direction when needed and giving invaluable feedback. Thanks to Professor Josiah Poon for providing feedback on my presentations throughout the year and asking difficult questions that drove this thesis forward. Thanks to Kunze Wang for introducing essential NLP concepts to me. Finally I want to mention my appreciation for the entire USydNLP group for teaching so much about NLP through presentations on your research.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

The field of Natural Language Processing (NLP), concerned with the analysis of textual data, has shifted from using statistical techniques such as the Hidden Markov Model (Rabiner and Juang, 1986) to deep learning techniques such as LSTM (Hochreiter and Schmidhuber, 1997) and later Transformers (Vaswani et al., 2017). These deep learning techniques have been shown to require increasingly large amounts of training data to achieve their current SoTA results. BERT was introduced in 2018 and trained on a corpus of 2.5 billion words, GPT-3 in 2020 trained on 300 billion words. Matching the Blanks (Baldini Soares et al., 2019), the current state of the art model on the popular Few-shot learning benchmark FewRel (Han et al., 2018), achieved this by pretraining on 600 million relation pairs. Pretraining is now an established technique in NLP, popularised by Devlin et al. (2019), that aims to reduce the need for large amounts of supervised data by first training a transformer on an unsupervised task and then fine-tuning it on a smaller supervised task. Even so a substantial amount of data is needed to perform fine-tuning. FewRel, despite being a Few-Shot dataset, contains 70000 labelled samples, the labelling of which was done through crowd sourcing due to the sheer number of samples and the necessity of multiple annotators to ensure a correctly labelled dataset. This shows there is significant effort required in the labelling process, and that research into Few-shot learning techniques that facilitate learning from smaller datasets is worthwhile to reduce the need for labelling.

Few-shot learning is a wide subfield including many different techniques such as data augmentation, contrastive learning (Chen et al., 2020) in computer vision, meta-learning in general and language model prompting specifically for NLP (Wang et al., 2020). We focus on Meta-learning, which encompasses a variety of techniques that have the general goal of learning a learning policy. Meta-learning models are explicitly trained to take a few examples each from a fixed number of classes and learn how to classify new samples using episodic learning (Vinyals et al., 2016). In this way meta-learners can learn the optimal way to perform few shot learning. Meta-learning has had several successful applications in NLP (Xu et al., 2019; Bansal et al., 2020a), to the extent that FewRel's evaluation style of 5 and 10-way problems is explicitly designed for meta-learners. However solving 10-way problems, where 10 classes are selected from the 100 in FewRel, is not particularly useful in real world classification problems where classification must be done across all classes. This pattern of N-way evaluation, popularised by (Vinyals et al., 2016), is dominant among meta learning and there is a clear research gap in the application of meta-learners to full classification problems, at least in NLP.

However the question arises: why apply meta-learning to full classification problems when other few-shot learning techniques could also be utilised? Open world learning (Geng et al., 2020a) and specifically class incremental learning (Masana et al., 2020), a new paradigm where classes are gradually added to a problem, present a use-case for meta-learning. This learning paradigm is quite relevant as many real world classification problems involve such a large number of classes that continue to grow that not all of them can be represented in the dataset, for example object detection or disease classification. When new classes are identified and labelled, class incremental learning systems have the ability to incorporate them into their model without requiring the full retraining a closed world supervised learning system does. There has been some research into applying meta-learners to this problem (Xu et al., 2019; Ren et al., 2020) as meta-learners both demonstrate few-shot capability and do not require retraining as classes are added.

There are two limitations to class incremental learning that we address. Firstly most systems typically require some partial retraining effort to incorporate a new class, which results in both a more complex learning process and forgetting of older classes (Masana et al., 2020). For example bootstrapping requires training a binary classifier for each new class, and to achieve better performance may need traversal of the entire dataset to find new training examples (Gao et al., 2020). Other incremental learning techniques make use of training with exemplars of previously seen classes and new examples to incorporate new classes. Masana et al. (2020) has highlighted that these techniques are essential in managing inter-task confusion, where a model has difficulty distinguishing between separately learned sets of classes. This is as opposed to meta-learners which only require few examples of each class to from a support set and perform a prediction. This can be considered as directly using exemplars for inference, and so can address the inter-task confusion problem without retraining.

The second limitation is the lack of few shot learning ability. When a new class is discovered it is likely that only few examples of the class has been found, hence immediate incorporation of the class requires a model with few shot learning capability. Previous investigations into few shot incremental learning both fall victim to the first limitation (Gao et al., 2020; Ren et al., 2020). However as meta-learners are few-shot learners they could potentially address both limitations.

Directly using existing meta-learning architectures to perform incremental learning has severe limitations however. We examine metric learning techniques which first embed support set examples and then compute distance between them to perform classification. However these techniques encounter difficulties in incremental learning due to class overlap, support set examples from unseen classes are embedded in a way that overlaps with other classes, making it hard to separate them (Ren et al., 2020). Ren et al. (2020) address this problem through training a separate embedding for new classes and aligning the embeddings during inference. To avoid training new embeddings we develop an aggregation system that is more suited to incremental learning. Instead of manually designing this system we construct a robust learnable aggregation technique, that can learn how to fully utilise a support set to accomplish a specific type of task such as incremental learning. Our technique uses a transformer to aggregate the support set, due to the global attention this allows an example of a class to consider examples from other classes and the query to better understand the problem distribution and modify its embedding. This system could allow the aggregation method to recognize overlapping classes and separate them in aggregation. We test these meta-transformers on existing few shot NLP datasets and find performance improvements over baselines and existing incremental learning techniques.

In summary this thesis makes three contributions to the fields of meta-learning, few-shot learning and transformer architectures in general.

(1) We investigate the relatively unexplored area of applying meta-learning to class incremental learning, and argue that this is a practical application for meta-learning.
(2) We design an innovative learnt aggregation technique, exploring the use of transformers in meta-learning and developing a new relative attention mechanism that could be reused in other set-to-set transformation tasks.
(3) We present a method of training a meta-learner using surrogate novel classes to encourage robustness to unseen classes, allowing a meta-learner to perform class incremental learning without any further training.

CHAPTER 2

Literature Review

Meta Learning is a major few-shot learning method encompassing a wide variety of techniques. In this chapter we give an overview of the field, describing in detail the trends of prototypical network based models and exploring meta-learning evaluation methodologies. A common limitation of these methodologies is their use of episodic evaluation, testing a meta-learner on only a small subset of classes in the dataset. To construct a new evaluate methodology we examine the open-world learning literature. Finally model compression techniques that can address the memory limitations of BERT as a meta-learner are outlined.

2.1 Meta-Learning

2.1.1 Overview

There are a great variety of meta-learning techniques, all of which share the idea of learning a learning policy in some way. The early descriptions of this idea (Schmidhuber, 1987; Bengio et al., 1997) didn't coin the term meta-learning but were more focused on the genetic motivation, e.g. with Schmidhuber (1987) referring to their algorithm as "Meta-Evolution". They also used genetic algorithms to optimize the learning policy. Meta-learning saw more interest with the advent of deep learning and its need for large datasets, by then it and the machine learning field in general had moved entirely to gradient descent based methods. Significant inventions were made such as episodic training, where tasks are sampled from a task distribution to train a meta learner (Vinyals et al., 2016). Tasks consist of a support set containing a fixed number of classes N each with K examples, along with a query set of samples that a meta-learner must classify into the classes in the support set, this is called an N-way K-shot episode. This training method is what gives meta-learners their few shot learning ability, by restricting K to be a small number e.g. 5 we can train models to learn how to learn an entire task from just 5 examples. Meta-learning evaluation is typically done with a test set containing a disjoint set of classes never seen in training and often only a slight drop in performance is found, which shows the power of these algorithms. Vinyals et al. (2016) also had a great impact on the evaluation of meta-learning algorithms, just as it introduced episodic training it utilised the same method for evaluation, where an episode is sampled from classes not seen in training. This influential paper defined the basic methods for training and evaluation of meta-learners, and narrowed the field to focus on N-way evaluation as opposed to full classification, the standard evaluation technique in supervised learning where a model classifies across all classes in the dataset.

Meta-learning was originally developed for and evaluated on computer vision problems, the early influential techniques (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017) all evaluate on the mini-ImageNet and Omniglot dataset. It was only later that meta-learning was applied to NLP, the earliest example of few-shot classification using meta learning being Yu et al. (2018) and Han et al. (2018) making the first attempt to standardise the field by introducing a benchmark. This means that benchmark datasets are limited, with only FewRel being widely used across multiple papers, this is further discussed below.

Finally there is a debate on how to categorise meta-learning models, with Hospedales et al. (2021) outlining three main categories. Optimization meta-learning where the learning policy is optimized through gradient descent. "Black-Box" meta learning such as Mishra et al. (2018) where the support set and query are fed into an RNN or CNN model and information from the labelled support set is used to classify the query. Finally metric-learning where a comparison metric between support set and query set are learned. However other authors present different categorizations, with Geng et al. (2019) considering "Black Box" meta-learning to be a part of optimization meta-learning. This argument holds ground, as Mishra et al. (2018) outlines a technique that feeds the support set as input to a CNN, it could be argued that as the weights in the CNN have been optimized to best use the information in the support set this is equivalent to finding the optimal way to learn from the support set. Our focus is more on metric-learning, specifically prototypical networks, and hence we consider "Black-Box" learning a subset of optimization learning.

2.1.2 Optimization Meta Learning

In optimization meta-learning the goal is to learn a parameterised learning policy through gradient descent. This can take many forms, originally it consisted of literally learning an optimizer using an LSTM that would output the optimal weight updates given the gradients w.r.t loss of parameters for a model

(Andrychowicz et al., 2016). This approach has the downside of being very complex, new parameters are introduced in the optimizer and it may be prone to overfitting.

Later the field shifted to focus primarily on learning optimal initialisation parameters for all tasks as in Model Agnostic Meta-Learning (MAML) (Finn et al., 2017) and its various modifications. This approach doesn't introduce any new parameters and has the advantage of being able to reuse standard optimization algorithms such as ADAM. It also has the major advantage of being model agnostic, any model can have its optimal initialization parameters found through the algorithm of "Compute optimal parameters on a sampled task, then use loss w.r.t. the initialization parameters to optimise them". The "Black-Box" meta-learning SNAIL algorithm (Mishra et al., 2018) was published around the same time, it also learns weights that make use of a support set. However the MAML model has stood the test of time, with it and its variants being used for many NLP applications such as (Bansal et al., 2020a), likely because the model-agnostic approach allows it to be easily applied to SoTA NLP algorithms such as BERT.

The primary limitations of MAML are computational expense in computing the second order derivatives for the initialization parameters, which involves back propagation through back propagation. Additionally the fact that a MAML optimized model has a fixed number of output classes as the models it optimizes are typically traditional supervised models, meaning it cannot be used for class incremental learning. The original paper presented a method of approximating this second derivative to handle the former problem. Bansal et al. (2020a) used a variant of MAML called LEOPARD to perform transfer learning, which necessitated the creation of a model with a flexible number of output classes. This problem was addressed by meta-learning a final classification layer parameter generator in a similar way to the initialization parameters. Though their method has several limitations, primarily the fact that they use the dot-product of weights with a query sample representation as opposed to cosine similarity. Gidaris and Komodakis (2018) showed cosine similarity to be advantageous for its normalisation properties as

$$cos(v_1, v_2) = \frac{v_1}{||v_1||}\frac{v_2}{||v_2||} \tag{2.1}$$

is the dot-product normalised by the magnitude of both vectors. A variation of particular interest to us is MIML (Dong et al., 2020) which uses meta-information in the form of the class names to accelerate and reduce the impact of noise in the support set learning process. It achieves near SoTA performance on the FewRel dataset, though it's unclear whether these improvements extend to other problems and its fast-adaptation technique limits the amount of learning from a support set regardless of its size, due to the usage of attention.

Overall these models have achieved high performance in a variety of NLP tasks. MAML has the main advantage of being easily applicable to any type of model. However metric learning methods such as prototypical networks were designed to handle problems with a varying number of classes without requiring extensions such as in Bansal et al. (2020a). They also require less computation in both training and inference as they do not have to perform gradient descent to learn from the support set, instead performing a simple forward pass.

2.1.3 Metric Learning

In contrast to literally learning a learning policy these techniques implicitly learn one. That is, they learn the best way to utilise a support set, through embedding that support set into a more useful space for comparison.

The first metric-learning algorithm was Siamese networks (Koch et al., 2015), which used a CNN that outputs a feature vector for each image and calculated similarity using the L1 element-wise distance between the two vectors, that is:

$$D(v_1, v_2) = \sigma(\sum_j a_j |v_{1,j} - v_{2,j}|) \tag{2.2}$$

Where a_j are learnable parameters, this approach again has the disadvantage of not normalizing for vector magnitudes. Its main disadvantage is that training for pairwise comparison is limiting, two classes chosen at random are likely to be significantly different from each other and hence easy to differentiate, whereas the 10-way case may include one or more classes that have overlapping meaning. In addition learning from the more challenging 10-way problem could result in a more robust model.

Vinyals et al. (2016) address this challenge by introducing episodic training, where meta-learners train directly on 10-way problems. They also adopt the cosine similarity measure to compare the vector representations of images. Finally they defined the popular miniImageNet benchmark used in later research, outlining the need for harder tasks with less data to evaluate meta-learners. Despite being otherwise similar to Siamese networks in that they use a CNN to embed each image network their matchine networks model achieves significant improvements in the Omniglot task, especially in the

20-way problem which gives further evidence for the limitations of pairwise comparison. The primary limitation was the fact they didn't address techniques to take advantage of support sets with more than one example, despite testing on the 5-shot case. 5-shot learning is a more realistic use case as it is not so hard to label some extra samples to get a significant performance boost.

Snell et al. (2017) introduced prototypical networks, a popular offshoot of metric-learning that handled the support set limitation through aggregating the feature vectors of each sample in the support set. This was done through simply averaging each representation of an image given by a CNN. They also further verified the idea of "training on harder problems is better", as they found performance improvements when training on Omniglot using a 30-way episode and evaluating using the 5-way task. Prototypical networks address the main limitation of meta-learning, the poor aggregation. Though their original implementation of aggregation was lacking as an average is sensitive to noise and outliers which was improved on later (Geng et al., 2019; Gao et al., 2019).

Others have investigated different methods of handling the support set limitation. Xu et al. (2019) showed an application of metric-learners to Amazon product classification and introduced a learned voting mechanism in the form of a Bidirectional LSTM. There is a limit to how much a voting system can do as it can only take as input the similarity scores for each support set example, not the support set example's features themselves, which could make it difficult to determine which examples are outliers. Of more interest is the paper's application to Amazon product category classification. Specifically they outline the need to use an "Open World Learning" paradigm for solving problems where new classes can appear quickly and must be learnt by a model, such as in product categories. This type of problem is more commonly referred to as Open Set Recognition (OSR), and is concerned with loosening the assumption that all classes for a problem are seen in training, encompassing a variety of techniques that handle various problems such as rejecting unknown classes (identifying them as unknown) and learning to classify them (Geng et al., 2020a). Metric learning in particular presents an ideal solution to the latter problem, it has both few-shot learning ability which is needed as classes not seen in training may have few samples available and the ability to classify new classes without retraining by adding their samples to the support set. Overall little research has been done into the use of meta-learning for OSR, Liu et al. (2020) applied a variant of prototypical networks to the few-shot OSR problem but neither paper considered the full classification problem, which leaves a significant research gap.

Gupta et al. (2020) showed another way to apply metric learning to the Amazon Review Sentiment Classification (ARSC) dataset, a few-shot NLP task. This was done through fine tuning a BERT model to rate the likelihood of two sentences, a prompt such as "task 2 star books" and the review, being joined together. This is similar to the comparison approach in Koch et al. (2015) in that it is trained for and performs pairwise comparison for each label, though here this was necessary by the nature of ARSC as multi-label task that requires a binary classifier trained for each label. While the technique is simple the main contribution of this paper lies in its critique of the ARSC dataset itself, namely that its too easy. Specifically they claim the tasks across different domains e.g. "electronics reviews", "kitchen reviews" are not very different from each other as fine-tuning on the held out tasks and including a task specific prompt gives no improvement. Hence because their model can perform zero-shot classification on new tasks with little difficulty the new task is not too different to those in the training set. This would be a limitation for testing meta-learners as we desire them to be able to generalise to significantly different unseen classes. This conclusion could simply be a result of their simplistic model not fully taking advantage of fine-tuning, the mean accuracy of their model is only 89.65% and the use of MAML could improve the effectiveness of fine-tuning in this case. Nevertheless the doubts raised mean we should not use ARSC as the sole or primary benchmark in our experiments.

Matching the Blanks (Baldini Soares et al., 2019) is the most relevant example of metric-learning, using a metric-learner on BERT and a large pretraining task to achieve SoTA performance on FewRel. There are two main innovations introduced by this model aimed at improving specifically the relation extraction ability of NLP meta-learners. The first is its architecture, shown below.

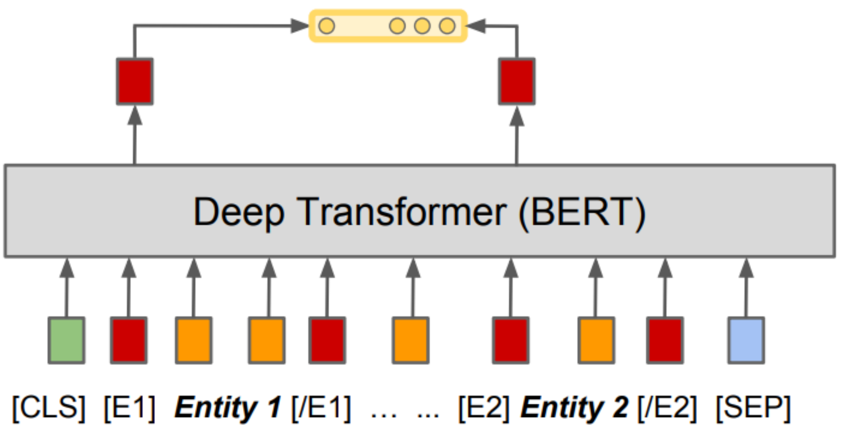


FIGURE 2.1. The entity markers + entity start state architecture of Matching the Blanks. Figure source: Baldini Soares et al. (2019) Figure 3

While the model only uses BERT, with a single linear layer on the output representation, it introduces innovations in modifying the input sequence and extracting the representation. Relation extraction is problem that involves discerning the relationship between two entities in a text. As FewRel provides the entity names along with each sentence a successful model should incorporate this information into the model somehow, so it is aware of which tokens represent the two entities. Matching the blanks achieves this through placing entity markers, shown as [E1] [/E1], in the input text to indicate to the model the position of the entity. This allows the model to understand the rest of the sentence in the context of the two entities, which should help it better discern the relationship expressed. The authors show this approach performs better than a simple positional embedding used in other papers, claiming this is because of their use of a deeper model: BERT Large with 24 layers.

The other innovation is that the representation from BERT is formed through concatenating the output at each of the two entity marker tokens. This is as opposed to the traditional method of taking output from the [CLS] token, which is often used by classification models as the [CLS] token is pretrained to perform a classification task, next sentence prediction (Devlin et al., 2019). The authors don't present any justification for this but it achieves the best results, probably it is complementary to the entity markers as their representation is calculated through attention with the rest of the sequence so contains information relevant to the entities and hence the relation. To make a prediction the similarity between each classes' support set examples and the query is calculated, using the dot product which has limitation of not being normalised as discussed above, then similarities are softmaxed. This method of using entity markers and taking the outputs from them is found to be the best performer, and is used by later models competing on the FewRel benchmark (Dong et al., 2020; Yu et al., 2020), hence we should also consider using this or developing an alternative method of entity marking.

The second major innovation relates to their pretraining task, they formulate a method of creating a large pretraining corpus without any supervision required. This is done through making the distant supervision assumption, a method previously used in relation extraction where it is assumed that if two sentences contain the same entities in the same order than they must have the same relation (Han et al., 2018). While this assumption does not always hold, e.g. Bill Gates was famously both the husband and former boss of Melinda Gates, it generally is true which is sufficient to create a useful pretraining task. To form the dataset for pretraining they crawl English Wikipedia and use an off the shelf named entity recognition (NER) model to identify the entities. Through this they form a dataset of 600 million pairs of relations half negative and half positive, they sample the relations so that common entities appear a maximum number of times to handle the bias towards popular entities, though this cannot generate relations for uncommon entities so there is a limitation. To increase the difficulty the negative samples consist largely of relations that share only one entity, so the text may share similar facts despite expressing a different relationships. Finally the entities in samples are replaced with "blanks", [MASK] tokens, so the model learns to match relations not just recognise two sentences contain the same entity. The loss is a combination of both cross entropy and masked language modelling, here the secondary objective of masked language modelling can act as a regularisation technique, requiring the model to learn classification while not forgetting the original task. This pattern of using language modelling as a secondary loss is seen across other papers including (Schick and Schütze, 2021) and could be useful in our research to add regularisation.

The key limitation of this pretraining process is that the same entities are likely to be described in similar styles across texts, and so the meta-learner may not learn to be style invariant. However overall it is effective enough to achieve high performance on FewRel without any fine-tuning. Unfortunately it is difficult to directly incorporate their pretraining, as neither the dataset nor the model weights are published and the cost to perform the NER required to reconstruct the dataset is large. Interestingly a similar pretraining approach has been used by Bansal et al. (2020b) to build a pretrained generic meta-learner for NLP, instead of blanking out entities it blanks out a single word from a sentence with the task of identifying which sentence in the support sample also had the same word blanked out. Perhaps this approach could be adopted for pretraining of the model, as it is less expensive to reconstruct and can be applied to other non relation extraction tasks

TABLE 2.1. Summary of Metric Learning Methods

Model	Summary	Key Limitation
Siamese Networks (Koch et al., 2015)	First Metric-learner, used a CNN to encode images then calculate distance	Pairwise comparison is a poor training method
Matching Networks (Vinyals et al., 2016)	Introduced episodic training, the use of cosine similarity and miniImagenet	Not suited for more than one-shot learning
Prototypical Networks (Snell et al., 2017)	Aggregate the support set representations into a prototype, then calculate distance	Averaging is a poor aggregation technique
L2AC (Xu et al., 2019)	Applies metric-learning to NLP, outlining the Open World Learning problem. Introduces a learnable voting function	The voting function is limited as it only considers one feature
BERT+prompt (Gupta et al., 2020)	Turns BERT into a few-shot learner through adding a prompt representing the task to the input	Pairwise training
Matching the Blanks (Baldini Soares et al., 2019)	Introduces techniques for labelling entities for BERT and a pretraining objective	Pretraining may result in a style-invariant model. Metric-learning's poor aggregation

Overall metric-learners are a powerful meta-learning technique, with several advantages over optimisation meta-learning. They also have the advantage of being model-agnostic, in that any model can be used to give an encoding which has allowed them to take advantage of models such as BERT. While also not being as computationally expensive to train or perform inference and having the ability to classify across a varying number of classes without retraining. The main drawback is their difficulty in performing more than 1-shot learning, Snell et al. (2017) partially addressed this through introducing prototypical networks but even then in larger shot cases MAML may achieve better performance through using gradient descent to learn from the samples. Though this is not too relevant to our research as we wish to study few-shot learning.

2.1.4 Evaluation Methods

To discuss the trend of evaluation methods of meta-learning models we first investigate the evaluation methods in each relevant paper mentioned in this review, summarising the datasets used along with the maximum N-way problem evaluated on in table 2.2.

TABLE 2.2. Meta learning models, their type along with datasets and evaluation methods (both N-way and K-shots outlined) used in meta-learning

Model	Type	Field	Datasets	Maximum N	Maximum K
(Andrychowicz et al., 2016)	Optimization	CV	MNIST, CIFAR-10, ImageNet	Full (1000-way)	N/A
(Baldini Soares et al., 2019)	Metric	NLP	FewRel, SemEval 2010 Task 8, KBP-37, TACRED	10	5
(Bansal et al., 2020a)	Optimization	NLP	GLUE, SciTail, ARSC	Full (13-way)	16
(Bansal et al., 2020b)	Optimization	NLP	CoNLL-2003, MIT-Resteraunt, ARSC, Figure-eight	Full (13-way)	32
(Chen et al., 2020)	Metric	CV	ImageNet	Full (1000-way)	
(Dong et al., 2020)	Optimization	NLP	FewRel	10	5
(Finn et al., 2017)	Optimization	CV	Omniglot, miniImageNet	20	5
(Gao et al., 2019)	Prototype	NLP	FewRel	10	5
(Geng et al., 2019)	Prototype	NLP	ARSC, ODIC	10	10
(Geng et al., 2020b)	Prototype	NLP	ODIC, MiniRCV1	10	5
(Gidaris and Komodakis, 2018)	Prototype	CV	miniImageNet, ImageNet	Full (507-way)	5
(Gupta et al., 2020)	Metric	NLP	ARSC	12	10
(Hariharan and Girshick, 2017)	Metric	CV	ImageNet	Full (507-way)	20
(Jiang et al., 2018)	Optimization	NLP	MiniRCV1	10	5
(Koch et al., 2015)	Metric	CV	Omniglot, MNIST	20	1
(Liu et al., 2020)	Prototype	CV	CIFAR10, miniImageNet	5	5
(Mishra et al., 2018)	Optimization	CV	Omniglot, miniImageNet	20	5
(Snell et al., 2017)	Prototype	CV	Omniglot, miniImageNet, Caltech-UCSD Birds	50	5
(Vinyals et al., 2016)	Metric	CV	Omniglot, miniImageNet	20	5
(Xu et al., 2019)	Metric	NLP	Amazon product descriptions	100	Full
(Yu et al., 2020)	Prototype	NLP	FewRel	10	5

In contrast to computer vision where Omniglot and miniImageNet are included in the vast majority of papers the datasets for NLP meta-learning are very scattered. The only datasets that are used across multiple NLP papers are FewRel, MiniRCV1 and ARSC (ODIC is used twice by the same authors and has data quality issues). However each of these has its own limitations, motivating the creation of a new dataset though this is outside the scope of our research.

MiniRCV1, introduced by Jiang et al. (2018), was created in a similar way to miniImageNet: through subsampling a well known dataset to create a few shot version. In this case it was created from the Reuters Corpus Volume 1 dataset (Lewis et al., 2004), consisting of 800000+ categorised news articles, where the task it to classify the category of news articles. MiniRCV1 is created through subsampling 20 documents from each of the 103 classes, with 5 documents used for training and 15 for testing. The main limitation of this dataset is that it is not published, hence we have to take our own random sample to reconstruct it, this is likely the reason it is not widely used. Only having 5 training samples also greatly restricts the experimentation possible i.e. 10-shot training is no longer possible.

The Amazon Review Sentiment Classification (ARSC) (Blitzer et al., 2007) dataset on the other hand has seen wide use in the field, is standardised and publicly available. However as we discussed in section 2.1.3 there have been doubts about the difficulty of the dataset raised by Gupta et al. (2020). It is a quite simple task and there are only 69 classes, it might be better to tackle the much harder problem presented in (Xu et al., 2019) which has over 2000 classes and is publicly available.

Of these FewRel has the highest quality so we use it as our primary benchmark, it is formed through crowdsourcing the labelling of relations drawn from Wikipedia. Each relation has been checked at least 2 times to ensure the correctness of the label so the quality is high. Finally the test set is held out by the authors, evaluation is done through submitting a model to a leaderboard. Though this has downsides, especially to our research as we wish to change the method of evaluation, it does lend credibility to the results of models evaluated on this dataset as it prevents overfitting on the test set and ensures reproducibility. Given the overall quality of this dataset we can utilise it as the primary benchmark for evaluating our model, with MiniRCV1 as a secondary benchmark to evaluate the meta-learner on a different task to relation extraction.

The introduction of fixed-way evaluation appears to have begun with Brenden et al. (2011) who introduced Omniglot and evaluated their non meta-learning technique on 20-way problems. From here onwards we refer to this setup as "episodic evaluation", following the "episodic training" terminology introduced in (Vinyals et al., 2016). The authors don't attempt to explicitly explain the usefulness of this evaluation technique, but it does have useful properties for evaluating a meta-learners classification ability. The performance of meta-learners varies depending on the support set samples randomly chosen, and so several trials should be run to find the mean performance, which episodic evaluation achieves. The other advantage is the ability to test a meta-learners' ability across multiple different problems. These advantages saw episodic evaluation adopted by later influential papers (Vinyals et al., 2016; Snell et al., 2017; Han et al., 2018), the later of which justifies their use of it by citing that almost all recent research methods use this.

That being said there has been criticism and alternatives raised by the research community. The first paper to challenge the realism of episodic training was (Hariharan and Girshick, 2017). While this paper was more focused on introducing other few-shot learning techniques it did evaluate against matching networks and prototypical networks. Of interest is the fact that their evaluation involves examining the top-5 accuracy of models on half of the ImageNet1k classes (the other half of the classes are for training). This was the first paper to argue for and evaluate meta learners on the "Full Classification" problem, which we define as the problem of performing classification across both base classes seen in training and novel classes. It shows the feasibility of applying meta-learners to this problem, given that matching networks achieve the top performance in the 1-shot case. This approach was used by others such as Gidaris and Komodakis (2018), however it failed to become as popular as episodic evaluation in computer vision and to the best of our knowledge was never used in NLP. Interestingly though other metric-learning techniques in computer vision have appeared independently that also evaluate on full classification problems, such as Contrastive Learning (Chen et al., 2020).

There is little debate about episodic evaluation in NLP, and to the best of our knowledge none have examined full classification yet. Bansal et al. (2020a) who introduced the MAML variant LEOPARD performed transfer learning with it, training a model on the GLUE benchmark and classifying across all classes in a variety of datasets. Though this doesn't make a distinction between base classes in training and novel unseen classes. Additionally both the GLUE problems and their evaluation datasets have very few classes, so the application of meta-learning techniques is questionable as they need large amounts of classes to perform episodic training. Finally, while focusing on computer vision, Wang et al. (2019) raises the issue that real world classification problems are likely to have a varying number of classes, the same motivation behind our interest in incremental learning problem. Unfortunately their evaluation doesn't test the ability of their technique to classify across all classes, and so this still isn't full classification. Overall while episodic evaluation has several advantages it is poorly aligned with real world problems, as shown by several other papers. We propose to also test our techniques on the more real-world aligned task: Full classification.

Finally we note that there have been studies, though mostly in the computer vision field, challenging the reproducibility of meta-learning. Dhillon et al. (2020) compares meta-learners to a supervised baseline, finding that a simple fine-tuning approach outperforms many other algorithms. The paper is older and hasn't been applied to NLP however it would be useful to see if these results transfer to our application, hence we should consider including a fine-tuning baseline in our evaluation. Musgrave et al. (2020) shows that several metric-learning papers have been unfairly biased in the evaluation of their technique compared to others, using several tricks to report performance improvements when in reality there are none. We are careful when evaluating our techniques to be fair, though we can have some confidence in the reported results of papers on FewRel as the testing set is hidden from researchers and evaluation is performed in an impartial manner by the authors of (Han et al., 2018).

2.2 Prototypical Networks

In this section we discuss in detail prototypical networks. Our method requires the use of metric-learning for its ability to handle a varying number classes, essential for incremental learning, hence we must choose one of the metric-learning techniques to improve on. We focus specifically on prototypical networks as opposed to other metric-learning techniques for their ability to better use a support set through their more sophisticated aggregation methods. There are three main parts of a prototypical network that can be modified, we outline these in figure 2.2.

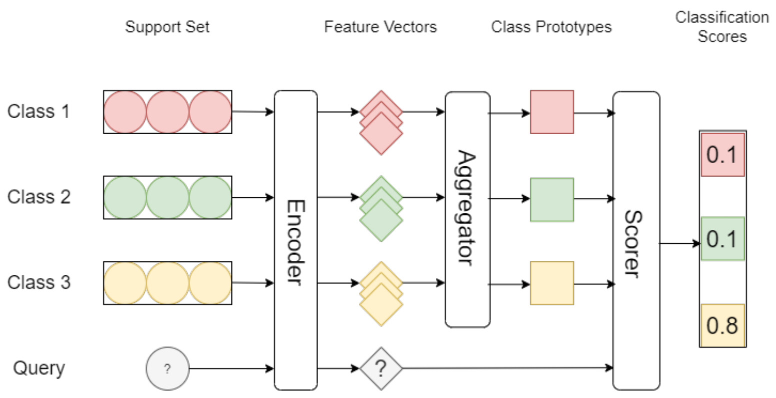


FIGURE 2.2. The standard structure of a prototypical network

The encoding module is responsible for transforming the support set and query samples into a vector representation. In recent NLP papers this is almost always done through BERT. The aggregation module takes the vectors for each support sample and combines them into a single prototype for each class, this is the module that is typically modified by new techniques. Lastly the comparison module computes a distance between the query and each class prototype, the network then makes a prediction by taking the softmax of the distances. There is also a debate surrounding the scoring module, with a partial consensus on the usage of cosine similarity though the euclidean distance is still in use along with other exotic methods. The loss functions and training procedures are in most cases simply episodic training with cross-entropy loss, so these are not discussed.

TABLE 2.3. Overview of different modules used in Prototypical Networks

Model	Encoding Method	Aggregation Method	Scoring Method
(Gao et al., 2019)	CNN	Feature and Query-aware Attention	Euclidean
(Geng et al., 2019)	Bi-LSTM	Induction	Neural Network
(Geng et al., 2020b)	BERT	Induction and Query-aware Induction	Cosine Similarity
(Gidaris and Komodakis, 2018)	CNN	Average and Attention	Cosine Similarity
(Liu et al., 2020)	ResNet 18	Average	Mahalanobis distance
(Snell et al., 2017)	CNN	Average	Euclidean
(Yu et al., 2020)	BERT	Query-aware Attention	Euclidean

2.2.1 Scoring Methods

Euclidean distance was the first method introduced by Snell et al. (2017) and has been used by others simply because of that fact (Yu et al., 2020). The justification given by Snell et al. (2017) for its use is the fact that Euclidean distance is a Bremen divergence, and that when using Bremen divergences as a distance metric the optimal cluster centroid is the mean of each cluster's samples. However this justification no longer has relevance now that few techniques use averaging to calculate prototypes.

The case for the use of cosine similarity is best made by Gidaris and Komodakis (2018), one of the earliest papers to use the technique in meta-learning. They compare cosine similarity with a common approach at the time of using the prototypes as weights in a softmax classifier, also seen in Geng et al. (2019), where the distance function is a simple dot product $D(v_1, v_2) = v_1 \cdot v_2$. Standard supervised weights have a small initialization and are slowly optimized through gradient descent, learning the optimal magnitude. However because the prototypes depend on the support set, and the samples in the support set are randomly chosen, the magnitude of each prototype varies greatly. Hence we need a distance metric that removes the effects of magnitude, which can be achieved using cosine similarity as it is simply the dot product with L2 normalisation. While the authors don't discuss euclidean distance the same argument could be applied, euclidean distance is based on the absolute magnitude between points and could suffer from the same problem.

Finally there are some other exotic methods, Neural Network simply means the two vectors have a dense layer applied to them, and then dot product is used, which has the same limitations described above. Gao et al. (2019) uses feature level attention to weight the importance of each feature before applying euclidean distance. Liu et al. (2020) uses the Mahalanobis distance which they claim can improve the ability of prototypical networks to perform rejection in OSR as it handles the fact that some dimensions of the prototype have a higher variance than others by scaling the distance function in those dimensions.

Overall it appears the ideal distance function may be model dependent, given the variety of distances that have seen success. Cosine similarity is a good baseline method, though we may wish to experiment with a previously unused distance to tackle the unique problem we introduce, as Liu et al. (2020) found success with this approach.

2.2.2 Aggregation Methods

The original aggregation technique introduced by Snell et al. (2017) is the arithmetic mean, it's somewhat obvious that this might suffer from problems such as outliers but not too easy to think of alternatives. The use other of means like the geometric mean to reduce the affect of outliers would result in the prototype changing but the query remaining the same and hence comparison between prototypes and queries becomes difficult as they are misaligned. Hence there is a need for query-aware methods to tackle outliers in a way that allows the query to be compared with the resulting prototype, a point which almost all prototypical networks agree on. The exact method of performing query-aware aggregation is the subject of much debate, as this is the primary area where prototypical networks are modified given that scoring methods have little impact and the choice of encoder is often simply the current SoTA supervised model in the field. The most complex of these methods is Dynamic Memory Induction Networks (DMIN) (Geng et al., 2020b), which also achieved SoTA on the Mini RCV1 dataset, the architecture is shown below.

DMIN performs aggregation through using two induction modules. The induction module consists of an implementation of capsule networks, which are a new type of deep learning model using the "routing-by-agreement" mechanism, in which the inputs choose which output capsules to be transformed by (Sabour

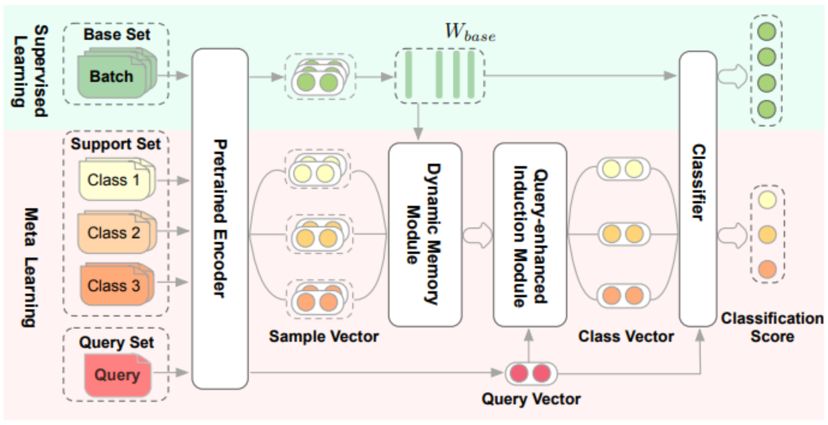


FIGURE 2.3. Architecture of Dynamic Memory Induction Networks. Figure source: Geng et al. (2020b) Figure 1

et al., 2017). This "agreement" is calculated by a dot product, and so in a way could be considered similar to attention, with the main difference being the "agreement" is calculated iteratively. DMIN itself uses these capsule networks to first use memory of the base classes to adjust the class prototypes and then use the query vector to adjust the class prototypes to be relevant to it. Other techniques such as Gidaris and Komodakis (2018); Gao et al. (2019) have also found success in calculating the prototypes through directly using attention, specifically an attention weighted sum of the support set i.e.

$$p_i = \sum_{s_i \in C_i} a(q, s_i) s_i \tag{2.3}$$

Where C_i refers to the set of samples of class i, a an attention function and q the query sample. The advantage of using attention is that it can find the samples in support set most relevant for classifying the query. For example if a product description included the fact it is small, then by examining the other descriptions of small objects from the support set classes a model focuses on samples that are more likely to have the same class instead of being distracted by other less important samples.

While the induction module may also achieve this there has been little reuse or reproduction of DMIN's results, whereas the simpler attention based approach of hybrid attention prototypical networks are widely used as solid baselines including in the FewRel benchmark (Sun et al., 2019a). Though it may be worth designing a system that also utilises the base class representations, as this approach found success both in DMIN and in (Gidaris and Komodakis, 2018). Ren et al. (2020) designed an aggregation technique specifically for incremental learning that uses attention alignment to handle the unique problems that arise in that setting. Overall there is a clear trend of adapting metric-learners to perform better and handle different tasks by designing new aggregation techniques, hence we develop a new aggregation method for incremental learning.

2.2.3 Encoding Methods

In contrast to the fierce debate over aggregation techniques there is little innovation in the encoding module, with most papers simply reusing the dominant SoTA supervised model i.e. BERT. Despite the lack of research it has been shown that improving encodings, through substituting LSTM for BERT, can significantly improve the performance of even older meta learners (Geng et al., 2020b). Apart from matching the blanks Yu et al. (2020) introduces the only other novel encoding relation method for relation classification we have found.

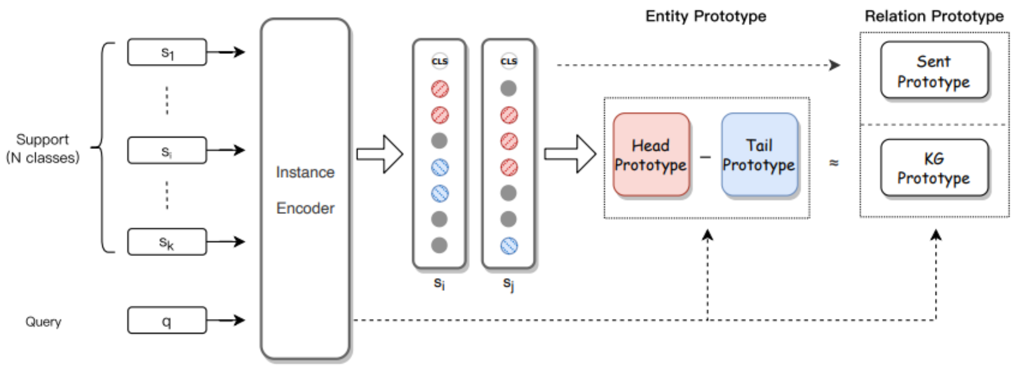


FIGURE 2.4. Architecture of the Multi Prototype Embedding model. Figure source: Yu et al. (2020) Figure 2

They use the well established assumption that for all samples in a relation class the head and tail entity should have a constant mapping between them, i.e. $H + R = T$. To create a prototype they first create prototypes for the head and tail entity, then form the relation prototype through the concatenation of the $T - H$ prototype and a standard prototype representing the sentence. Their ablation studies show that this method yields a significant performance increase. Improved encoding methods have shown to yield significant improvements, especially (Baldini Soares et al., 2019) and so we test our meta-learner with a several different encoding methods.

2.3 Incremental Learning and related Open learning paradigms

2.3.1 Overview

The standard classification problem, referred to as closed world learning in open learning literature, makes the unrealistic assumption that all classes in the underlying problem are present in the dataset (Bendale and Boult, 2015). This assumption doesn't hold for many real-world problems, for instance the FewRel contains 100 examples of relations in the Wikidata knowledge graph but there have since been thousands of new relations added (Gao et al., 2020). The closed world assumption means that classification models will misclassify data from unseen classes into existing categories, see figure 2.5.

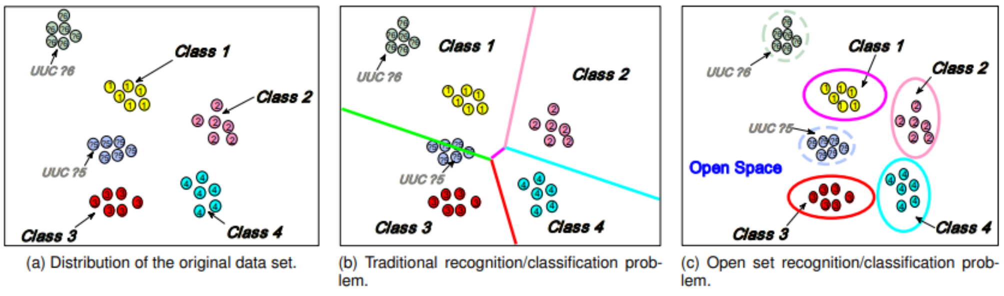

UUC ?6
Class 1
Class 2
UUC ?5
Class 3
Class 4
(a) Distribution of the original data set.
(b) Traditional recognition/classification problem.
Open Space
(c) Open set recognition/classification problem.

FIGURE 2.5. Traditional classification models misclassify, whereas Open World Models can reject samples from unseen classes based on distance from the known classes. Figure source: Geng et al. (2020a) Figure 2

This could be dangerous, for example in the medical field where new diseases are constantly created a misclassification could result in incorrect treatment. In addition it would be desirable for a model to be able to learn to identify these new classes, preferably from the few examples obtainable, hence why we examine meta learning for incremental learning. To address these problems Open World models perform at least one of three tasks: rejection of samples from unknown classes, identification of unknown classes and incremental learning of new classes.

2.3.2 Types of Open World Learning

The Open world paradigm encompasses a wide range of different types of models, of which there are three main types. Scheirer et al. (2013) outlined the first Open World Learning method, Open Set Recognition (OSR). They explained that given an open world problem there will be samples from unseen classes, and that these samples need to be rejected, developing the 1-vs-Set SVM to perform rejection.

While OSR is concerned with rejection of unknown classes, incremental learning focuses on methods for continually training a model. The earlier example of this are more akin to online learning, where a model is trained on new samples for existing classes that continually stream in. Cauwenberghs and Poggio (2001) presents an early example of an online learning model for machine learning, they outline the key challenge of this field: how to learn from additional samples in an effective manner without retraining the entire model. More relevant to us and open world learning is the area of class incremental learning, where a model learns to identify new classes (Masana et al., 2020). Here the key challenge of avoiding retraining is especially difficult, traditional supervised learning models have a fixed number of classes so require modification and applying meta-learning naively can result in novel classes overlapping with base classes (Ren et al., 2020).

Finally Open World Recognition (OWR) is a combination of the two that aims to accomplish all three goals of rejection, incremental learning and automatic identification of unknown classes (Bendale and Boult, 2015). In practice though, automatic identification of unknown classes is quite difficult and while stated as a goal in the influential literature (Bendale and Boult, 2015; Boult et al., 2019) there are few solutions addressing it, which leaves an interesting area for future work.

2.3.3 Class Incremental Learning in Meta-Learning

This thesis focuses on the class incremental learning paradigm, we set aside the rejection capability of Open World Recognition, while rejection is useful models that use it must trade off with accuracy on the learnt classes which presents a substantial challenge for future work. Masana et al. (2020) claim that application of meta-learners to incremental learning is a new trend that has potential and we observe this with new techniques such as in Gidaris and Komodakis (2018); Ye et al. (2020); Ren et al. (2020). Meta-learning is particularly well suited to class incremental learning due to its ability to add new classes simply through expanding the support set, along with few shot learning capability to learn new classes that may only have a few examples.

Gidaris and Komodakis (2018) introduced the first metric-learning based class incremental learning system. Their method finds prototypes for novel classes based on an attention weighted sum of the prototypes of the base classes, with the aim of remembering knowledge from the base classes to create novel class representations that are more aligned with the base classes. However this method isn't meta-learning, as it doesn't use episodic training, instead using traditional supervised learning for the base classes, which means the few shot learning ability for base classes and potentially novel classes may be damaged. We aim to develop a completely episodically trained meta-learning technique.

Ye et al. (2020) introduced a set-to-set aggregation technique for meta-learning and evaluated it on the incremental learning task, though the technique was not explicitly designed for incremental learning. This is the primary limitation, there is no training objective encouraging their parameterised aggregation method to learn how to best perform the incremental task. We aim to introduce such an objective using surrogate novel classes to simulate the incremental task while training. To facilitate the learning process we also make several large architectural changes to make the aggregation more flexible.

Finally Ren et al. (2020) is one of the first examples to apply meta learning to an incremental learning task in NLP. It trains two separate encoders for base and novel classes, embeds queries using both encoders and then uses an attention alignment technique to form a final query embedding based on attention between the query encodings and the novel and base class prototypes. The aim of using two encoders is to avoid the encoding overlap problem shown in figure 2.6, which they find causes a standard meta-learner to struggle to separate novel classes and misidentify novel samples as coming from base classes.

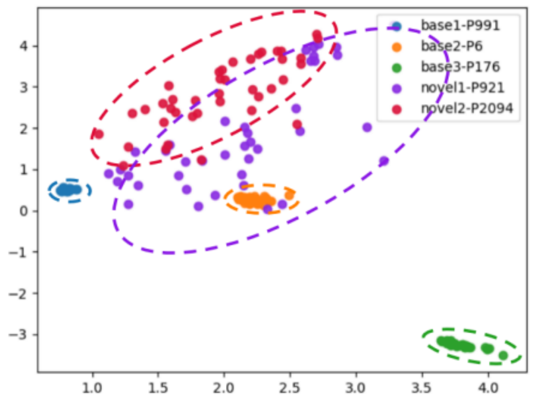


FIGURE 2.6. Novel Classes embedded using an encoder trained on Base classes (in the FewRel dataset). While the base classes are well separated the novel classes overlap with themselves and base classes. Figure source: Ren et al. (2020) Figure 1

The system bears a slight resemblance to Gidaris and Komodakis (2018) in that it uses attention to align novel class embeddings with base classes, however to align embeddings from two encoders requires a different usage of attention. This method has the primary limitation of requiring retraining when new classes are encountered, which we aim to remove by directly training for incremental learning using surrogate novel classes. The novel class overlapping problem it raises is a serious problem, and one we observe in our own experiments, we aim to address this problem through the use of surrogate novel classes to encourage the model to develop encodings that are robust to unseen classes.

Masana et al. (2020) outline four main challenges in incremental learning, which any learning system must address. These challenges are mainly a result of the difficulty of training purely sequentially. Masana et al. (2020) define class incremental learning as the process of sequentially learning a series of tasks that each contain a disjoint set of classes, with the end goal being a model that can perform classification over all learnt classes. This sequential training means the model may forget how to perform tasks learnt earlier, through either weight drift where weights slowly adjust to newer problems and lose effectiveness, activation drift which is similar but focuses on the shift in activation, or task recency bias where recently learnt tasks tend to have higher base logits. However our meta-learning approach means these three challenges don't apply as we only perform episodic training once for a large set of base classes, and add to the meta-learner's support set to learn new classes without training.

The major challenge our model will face is inter-task confusion: where incremental learners struggle to differentiate between classes coming from different tasks, because the learner did not train jointly on the classes from said tasks and so never explicitly learnt to separate them. This is a more general case of the novel and base class overlapping observed by Ren et al. (2020), the novel and base classes are learnt disjointedly and so overlapping occurs. We address this issue with surrogate novel classes.

2.4 Model Compression and Knowledge Distillation

2.4.1 Overview

Model Compression is a field concerned with reducing the size of a large model while maintaining its performance, with the goal of allowing very complex SoTA models to be imitated by a smaller model that can run on less powerful hardware (Gou et al., 2021). This is done through teaching a smaller model, the "student" to imitate the larger model, the teacher. Model Compression was outlined by Bucilă et al. (2006), who aimed to reduce the complexity of large ensembles of decision trees through training a neural network to imitate their predictions. They accomplished this by training the student neural network to minimise a loss between the prediction logits of the student and teacher for the same data sample. They showed that when the original training data is small a large amount of unlabelled synthetic data can be used with this loss to convey knowledge about the teacher model to the student.

This technique of learning from a larger model's logits was later popularized as knowledge distillation by Hinton et al. (2015), who also presented a method of distilling an ensemble to a single model. They highlighted the usefulness of using the teacher's logits as "soft" labels that are not one hot vectors but assign probability to incorrect classes. Using soft labels imparts more information to the student model, e.g. it can understand that the teacher model views the "cat" class as similar to the "dog" class when it assigns a significant probability for "cat" to an image of a dog. Later techniques expanded on knowledge distillation by adding losses to minimize difference between the hidden state activations of the student and teacher models (Romero et al., 2015). In particular it was discovered that performing knowledge distillation in stages was beneficial, Romero et al. (2015) outlines a method of distilling a model by first learning from the hidden state matching loss function and then from the logit matching loss function. This pattern of performing knowledge distillation in stages was adopted and expanded upon by later knowledge distillation methods for BERT that are of interest to us (Jiao et al., 2020).

There are two other major methods of model compression: pruning and quantization. Quantization involves reducing the precision of the data type used to store a model's weights and evaluate its activations. Vanhoucke et al. (2011) outlined an implementation of quantization that converted the weights of a trained neural network to 8-bit integers, by normalizing each weight into the range $[-128, 127]$. Converting to 8-bit integers reduces the model storage size and memory usage, through the use of specialised software a speedup in inference can also be achieved. While Vanhoucke et al. (2011) simply reduced the precision of a trained network, later quantization methods introduced more sophisticated techniques to reduce the performance loss introduced by quantization. Jacob et al. (2018) outlined a process of simulating quantization during training of a network through applying "fake quantization" to the weights and activations of a full precision network. This was done by normalising the variables' range to $[-128, 127]$ and rounding each to simulate transformation of variables into integers, with the goal of optimising the network to be robust to the quantization process that is performed after training. Quantization is a viable alternative form of model compression that can also reduce memory usage, however unlike knowledge distillation it requires specialised hardware to achieve a speedup in processing.

Pruning methods involve measuring the importance of weights in a neural network and removing unimportant weights to reduce the model size (Blalock et al., 2020). There are two categories of pruning methods, the first being unstructured pruning where each weight is considered and pruned individually. This is as opposed to structured pruning, where groups of parameters or entire layers are pruned to decrease the overall model size. Unstructured pruning results in a sparse model, with weights throughout the architecture removed, which may not increase speedup as typical matrix multiplication algorithms assume dense matrices. Structured pruning solves this through removing groups of parameters to decrease layer size which results in a speedup and memory savings. However Ganesh et al. (2021) found that compared to other BERT model compression methods such as quantization and knowledge distillation the model size reduction achieved by pruning techniques was limited. Potentially because pruning techniques may reach a point where almost all unimportant weights have been pruned and further reduction would significantly hurt performance.

We utilise model compression and specifically knowledge distillation because of the large amount of memory that meta-learners require, especially when using a large transformer to encode text for NLP tasks. Evaluating a 59 way 5-shot problem as is done by Ren et al. (2020) requires processing $59 * 5 + Q$ samples where Q is the number of query samples. Training on such a large task results in a batch size of at least 295 text samples that must all be encoded using BERT, which requires a large amount of memory. Hence we perform model compression to reduce the size of BERT to make it practical for meta learning. In our experiments we use a knowledge distillation technique as they can greatly reduce the memory usage of a model at a small cost to accuracy. Compared to quantization they demonstrate a much greater speedup that doesn't require specialised software, such a speedup is quite useful to reduce the time needed to perform experiments.

2.4.2 Knowledge Distillation for BERT

Due to the sheer size of SoTA language models in the pretrained transformer era there has been much research into knowledge distillation for BERT and transformers in general (Ganesh et al., 2021). Early method of distilling such as in (Sanh et al., 2019) were comparatively straightforward applications of the work done by Hinton et al. (2015) that simply matched the logits between teacher and student during pretraining. Later Sun et al. (2019b) applied the work done in Romero et al. (2015) to match the hidden layer activations using Mean Squared Error, which allows the student to more closely imitate the teacher by matching the hidden states. This matching can be done even when the hidden layer size is reduced through the use of a learned mapping function between the smaller layer and the larger layer, see figure 2.7.

The second transformer specific innovation was using the attention matrix to perform distillation (Aguilar et al., 2020). The justification for this is that the student model should understand the relations between words, represented by attention, in a similar way to the teacher. This distillation is done simply through

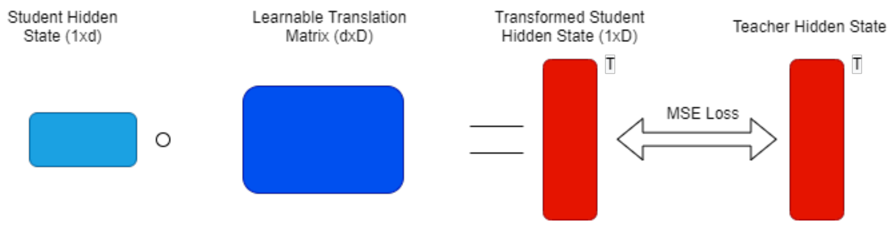

Student Hidden State (1xd)
Learnable Translation Matrix (dxD)
Transformed Student Hidden State (1xD)
Teacher Hidden State
T
MSE Loss
T

FIGURE 2.7. A learned transformation matrix allows knowledge distillation through training the student's hidden activations to match the teacher's. Both the student's parameters and the translation matrix are learnt through minimizing the MSE Loss

computing a loss such as KL divergence or Mean Squared Error between the student and teacher's attention matrices for each layer.

Using these innovations Jiao et al. (2020) created a 10x smaller transformer called TinyBERT by combining all three forms of knowledge distillation: logit, hidden state and attention matching, to allow for a large reduction in model size. The learnable translation matrix allows it to reduce the hidden size of the layers from 768 to 312, they also compress the layers through matching the i-th layer in the student model to the $3i$-th layer in the teacher, reducing the number of layers from 12 to 4. Finally they introduce a data augmentation method to increase the amount of data used for distillation, which performs a similar function to the unsupervised data used by Bucilă et al. (2006) to convey knowledge of how the teacher responds to a wider variety of data.

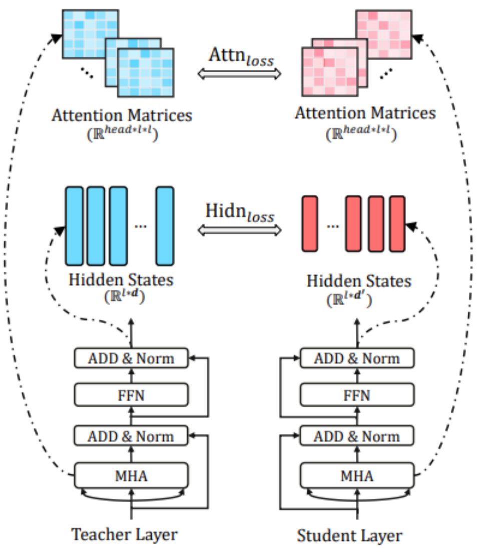

Attn$_{loss}$
Attention Matrices ($\mathbb{R}^{head*l*l}$)
Attention Matrices ($\mathbb{R}^{head*l*l}$)
Hidn$_{loss}$
Hidden States ($\mathbb{R}^{l*d}$)
Hidden States ($\mathbb{R}^{l*d'}$)
ADD & Norm
FFN
ADD & Norm
MHA
ADD & Norm
FFN
ADD & Norm
MHA
Teacher Layer
Student Layer

FIGURE 2.8. TinyBERT's additional knowledge distillation losses aim to match the hidden states and attention matrices of the teacher and student. Figure Source: Jiao et al. (2020) Figure 2

Ganesh et al. (2021) compares many different model compression techniques and we find TinyBERT to be the better option based on the results for memory reduction and speedup. Compared to BERT-base TinyBERT gives a 13.3% reduction in model parameters which translates to a large reduction in memory usage, and a $9.4x$ speedup in training. There are other compression methods surveyed by Ganesh et al. (2021) however we focus on knowledge distillation because it empirically gives the best combination of improvements in memory requirements and speedup.

CHAPTER 3

Methodology

We have identified the limitations in meta-learning evaluation and now aim to develop a meta-learner that performs class incremental learning. To address the limitation of existing models requiring retraining we devise a training procedure that emulates the class incremental learning problem using surrogate novel classes. Then we introduce a transformer-based meta-learner that can benefit from this training procedure through learning an aggregation method. Finally we outline how model compression is performed.

3.1 Surrogate Novel Classes

To manage the limitation existing class incremental learning models have of requiring further training to incorporate new classes we aim to introduce a meta-learner that only requires training on base classes. However this raises a serious problem: how will our model handle the inter-task confusion problem without some form of joint training on base classes and new classes? To address this we utilise surrogate novel classes: classes generated with self-supervised techniques used in training to emulate new classes. Using self-supervised techniques we can generate a new unseen class for each meta-learning episode, simulating the incremental learning task and encouraging a meta-learner to learn how to incorporate classes it has never seen previously.

The episodic training method for standard meta learning involves sampling N classes from a set of base classes $B = \{C_1, C_2, ..., C_b\}$ to obtain the episode's classes $E \subseteq B$. For each of the sampled classes we then randomly sample k examples from each of those classes to form a disjoint support and query set.

$$S = \{S_1, S_2, ..., S_N\} \text{ where } S_i = \{(x_1, y_1), (x_2, y_2), ...(x_k, y_k) | x_j \in C_i \in E, y_j = i\}$$

$$Q = \{Q_1, Q_2, ..., Q_N\} \text{ where } Q_i = \{(x_1, y_1), (x_2, y_2), ...(x_k, y_k) | x_j \in C_i \in E, x_j \notin S_i, y_j = i\}$$

Then the meta-learner makes predictions for each element in the query set and a loss is used to update the parameters of the model. New episodes are created and used to train the model until training is complete after a set number of iterations.

Algorithm 1 Episodic Training Algorithm for a single episode

```
  UniformSample($S, n$) uniformly samples a set of $n$ items from the set $S$ without replacement
**Input:** $N$, $K$, $lr$, $B$, A meta learner $F_\theta : S, Q \to Y$, Loss function $L : Y, Y \to \mathbb{R}$
**Output:** An updated $\theta$
  $E \leftarrow$ UniformSample($B, N$)
  **for** $i \in \{1, 2, ..., N\}$ **do**
      $S_i \leftarrow$ UniformSample($E_i, K$)
      $Q_i \leftarrow$ UniformSample($E_i \setminus S_i, K$)
  **end for**
  $S \leftarrow \{S_1, S_2, ..., S_N\}$
  $Q \leftarrow \{Q_1, Q_2, ..., Q_N\}$
  $\theta \leftarrow \theta - lr * \nabla_\theta L(F_\theta(S, Q), Y)$          ▷ Perform a step of gradient descent
```

Testing is done by sampling support and query sets from a set of novel classes $NC = \{C_b + 1, C_b + 2, ..., C_b + n\}$ that is disjoint from the training classes. The goal of this setting is similar to the goal of the test set in standard supervised learning: we wish to test the model's generalization ability, only here we measure generalization to unseen classes.

However in the testing setting for class incremental learning the support and query sets include also include new novel classes, i.e. $E \leftarrow$ UniformSample($B \cup NC, N$). This is an unfamiliar setting to the meta-learner, it was not trained to classify between base and novel classes and so will experience the inter-task confusion problem. To mitigate this effect we aim to better align the training setting with the class incremental learning test setting, through adding surrogate novel classes in training. We first generate a large set of surrogate novel classes $SC = SC_1, SC_2, ...SC_s$. Then during training we add a fixed number $NS << N$ of them to the support and query set, sampling E in a way similar to $E \leftarrow$ UniformSample($B \cup SC, N$) and hence similar to class incremental learning. The full algorithm is outlined below.

The idea of imitating novel classes is not new, many others have used such techniques to make their model robust to novel classes (Gidaris and Komodakis, 2018; Ren et al., 2020). However in previous methods these novel classes come from a fixed holdout set split off from the original training data. This reduces the amount of training data and causes a problem: either the novel classes are used multiple times in training and can be overfit to like the other training classes, or they are only used in a limited number of gradient updates to the model and so have a reduced impact. Our innovation is to sample

Algorithm 2 Episodic Training Algorithm with Surrogate Novel Classes

```
  UniformSample($(S, n)$) uniformly samples a set of $n$ items from the set $S$ without replacement
**Input:** $N$, $NS$, $K$, $lr$, $B$, $SC$ A meta learner $F_\theta : S, Q \to Y$, Loss function $L : Y, Y \to \mathbb{R}$
**Output:** An updated $\theta$
  $E \leftarrow$ UniformSample$(B, N)$
  $E \leftarrow E \cup$ UniformSample$(SC, NS)$
  **for** $i \in \{1, 2, ..., N\}$ **do**
      $S_i \leftarrow$ UniformSample$(E_i, K)$
      $Q_i \leftarrow$ UniformSample$(E_i \setminus S_i, K)$
  **end for**
  $S \leftarrow \{S_1, S_2, ..., S_N\}$
  $Q \leftarrow \{Q_1, Q_2, ..., Q_N\}$
  $\theta \leftarrow \theta - lr * \nabla_\theta L(F_\theta(S, Q), Y)$          $\triangleright$ Perform a step of gradient descent
```

novel classes from a far larger set of self-supervised classes, so that the model only observes each class once and hence is unlikely to overfit to them, better simulating how a model will process a completely unseen class.

Through this simulation the meta-learner can learn how to best perform incremental learning. We hypothesise that to adapt to the incremental setting the aggregation function must be learnable in order to make use of information in the support set to identify novel classes, and so we introduce such an aggregation in section 3.2.

Implementing surrogate novel classes requires a self-supervised method of generating one class per episode. Given we train for 30000 episodes this presents a challenge: we need a method that can generate such a large number of classes, each with at least 10 examples in order to sample a support and query set from them in the 5-shot task. To do this we use the pretraining method described by Baldini Soares et al. (2019), discussed in section 2.1.3. This method performs named entity recognition on sentences in a corpus, generating a large number of relation instances (h, s, t) where h is the head (first) entity, t the tail (second) entity and s the sentence the two entities were drawn from. We then assume that all instances which share the same head and tail entities belong to the same class. Finally in order to use these classes for training we mask the head and entity in the sentence, to prevent the classifier from simply comparing entities to identify a relation. We present examples generated by the technique below.

We can see some limitations such as overlap of multiple relations, for instance the first example could also represent the relation "Nexon published a game in Japan", but by indicating the entities Tokyo and Japan the relation becomes clearer. The other problem is that some relations are formed simply by the

Entities	Examples
Head: Tokyo, Tail: Japan	A similar publishing deal was made with the **Tokyo** based company Nexon to distribute and market the game in South Korea and **Japan**
	He lives in Meguro in **Tokyo**, **Japan**
Head: Turing, Tail: Bletchley Park	**Turing** worked for the Government Code and Cypher School GC&CS at **Bletchley Park**
	Turing's work at **Bletchley Park** was named an IEEE Milestone

TABLE 3.1. Examples of Classes Generated Through Matching the Blanks

presence of two entities in a sentence, even if that sentence doesn't mention the relation between them. The creation of a better self-supervised relation extracting mechanism is an interesting challenge for future work.

3.2 Meta Learning with Transformers

In our thesis we design a learnable aggregation model that can adapt to the class-incremental learning problem. As opposed to manually designing an aggregation using attention mechanisms such as in (Ren et al., 2020), we instead design a transformer with learnt attention weights that allow it to better control how the results of the attention mechanism are used. This model is named MetaFormer, short for Meta-learning Transformer and can theoretically learn to emulate other models such as Prototypical Networks and surpass them.

There have been other learnable aggregation models. Matching Networks (Vinyals et al., 2016) used an attention based LSTM to adapt the support set samples, our model utilises a transformer with the improved attention mechanisms that have developed since. Few-shot Embedding Adaptation with Transformer (FEAT) (Ye et al., 2020) also explores the use of a transformer to adapt the support set, however they first form prototypes of each class in the support set before adapting these prototypes. This limits the information the Transformer can use for aggregation to the mean of each class's K samples, we instead perform adaptation on all support set samples and the query which can allow the Transformer to better understand and aggregate a support set.

3.2.1 Set-to-Set Transformations

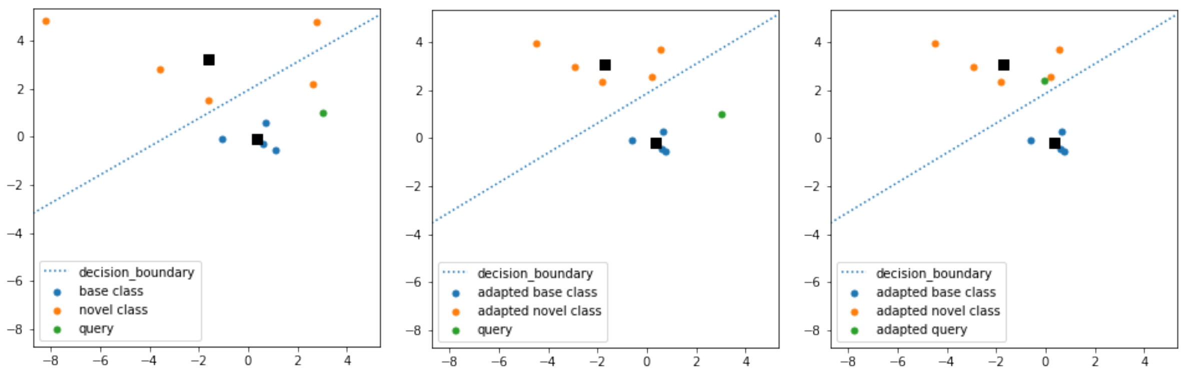
The framework of using Set-to-Set Transformation to perform meta-learning was outlined by Ye et al. (2020) and describes how the adaptation in models such as FEAT and Matching Networks functions. A

Set-to-Set transformation is performed by adapting a support set $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ using a learnable set-to-set transformation function $t_\theta : S \to \hat{S}$ that takes as input a set and modifies each element to form a new set. After adaptation the transformed support set $\hat{S}$ is used to generate class prototypes, usually through averaging, then a scoring technique is used between those prototypes and the query to calculate logits. A summary of the general algorithm is shown in algorithm 3.

Algorithm 3 Inference of a Set-to-Set Transformation based Meta-Learner F_θ

```
Input: $S, Q, N, K$, A Set-to-Set Transformation $t_\theta : S \to \hat{S}$, Sample embedding function $g_\phi : x \to \hat{x}$,
  Scoring function $sim : \hat{x}, \hat{x} \to \mathbb{R}$, Aggregation function $agg : S \to \mathbb{R}^{\text{hidden size}}$
Output: Logits $\hat{y}$
  $S \leftarrow \{(g_\phi(x), y) | \forall (x, y) \in S\}$
  $Q \leftarrow \{g_\phi(x) | \forall x \in Q\}$
  $\hat{S} \leftarrow t_\theta(S)$
  for $i \in \{1, 2, ..., N\}$ do
      $P_i \leftarrow agg(\{x | (x, y) \in \hat{S}, y = i\})$
  end for
  $\hat{y}_{ij} \leftarrow sim(Q_i, P_j)$
```

The aim of this transformation function is to aggregate using knowledge from the entire support set, the knowledge of how each class's samples are clustered could be used to further separate the class clusters and shift outliers closer to the center of their likely class, improving the results after scoring is applied. In incremental learning t_θ could be used to correct for the overfitting of the encoder to the base classes, by identifying a novel classes spread out distribution and aggregating it to reduce the spread, so that the scoring method doesn't misidentify spread out classes. We illustrate the ideal case for a set-to-set aggregation model in figure 3.1.



(A) Pre Adaptation (B) Post Adaptation without Query Adaptation (C) Post Adaptation with Query Adaptation

FIGURE 3.1. A Set-to-Set Adaptation applied to a synthetic 2-way 5-shot meta-learning problem. The decision boundary is calculated through the euclidean distance between a sample and the class means. Class means are shown in black

The change between figures 3.1a and 3.1c shows how adaptation with the full support set could be advantageous. It can recognise that the novel class is spread horizontally compared to the tightly centered base class, shifting the query that is next to a novel class sample toward its center and producing the correct prediction. This example also shows the advantage of adapting using the entire support set to adapt compared to the FEAT model, which only adapts the class means, so lacks the knowledge that the novel class is spread horizontally and would likely misclassify the query. Figure 3.1b shows why the Set-to-Set transformation function must also adapt the query. Without query adaptation even if we tighten the distribution of the novel class, the decision boundary is only slightly affected and the query remains on the wrong side of it.

To perform query adaptation we include the queries Q into the support set S and then perform adaptation as usual. However there are some challenges to consider, we could include the entire query set in the adaptation and set $\hat{S}, \hat{Q} = t_\theta(S \cup Q)$. However this would transform the model into something similar to a transductive algorithm, that learns from a specific set of examples to perform prediction for a specific query set. Such an algorithm could learn from the distribution of the queries, however it may have trouble if only one query is provided which often occurs in the real world. Instead we perform adaptation in an inductive manner, only providing one query to the adaptation function. This also allows the model to better aggregate for one specific query, as multiple queries may require different adaptations. Our new Set-to-Set Transformation algorithm is outlined in algorithm 4, it loops through each query, adding it to the support set which is then adapted and used to calculate the logits.

Algorithm 4 Inference of a Set-to-Set Transformation based Meta-Learner F_θ with query adaptation

```
Input: $S$, $Q$, $N$, $K$, A Set-to-Set Transformation $t_\theta : S \to \hat{S}$, Sample embedding function $g_\phi : x \to \hat{x}$,
    Scoring function $sim : \hat{x}, \hat{x} \to \mathbb{R}$, Aggregation function $agg : S \to \mathbb{R}^{\text{hidden size}}$
Output: Logits $\hat{y}$
  $S \leftarrow \{(g_\phi(x), y) | \forall (x, y) \in S\}$
  $Q \leftarrow \{g_\phi(x) | \forall x \in Q\}$
  for $i \in \{1, 2, ..., N * K\}$ do
      $X \leftarrow S \cup Q_i$
      $\hat{X} \leftarrow t_\theta(X)$
      $\hat{S} \leftarrow \{(x, y) | (x, y) \in \hat{X}\}$
      $\hat{q} \leftarrow x \in \hat{X}$                                ▷ retrieve the adapted query sample
      for $j \in \{1, 2, ..., N\}$ do
          $P_j \leftarrow agg(\{x | (x, y) \in \hat{S}, y = j\})$
          $\hat{y}_{ij} \leftarrow sim(\hat{q}, P_j)$
      end for
  end for
```

Note that our MetaFormer is both a learnable adaptation and aggregation model. It directly learns support set adaptation i.e. transforming the support set from X to $\hat{X}$. As this adaptation incorporates information from the entire support set into each adapted x however it also performs a form of aggregation, in combination with using the vector mean as agg this results in a learnable aggregation.

3.2.2 Transformers as Set-to-Set Transformations

Transformers were created to utilise the powerful properties of attention mechanisms to form a model composed solely of those mechanisms and feedforward layers (Vaswani et al., 2017). The self-attention mechanisms allow a Transformer to have a global understanding of sequence to sequence problems, hence they are useful to the support set adaptation problem. We propose the use of a modified Transformer to perform the set-to-set transformation t_θ. Such an architecture can use global attention to better understand the support set distribution, using that knowledge to perform transformations such as in figure 3.1.

Transformers were originally introduced as an encoder-decoder architecture for the purposes of neural machine translation. The purpose of this architecture is to encode an input sequence, the source language, then use the encodings to decode into the output sequence, the target language. The encoder takes a sequence of vectors that represent words and applies multiple layers of multi-head attention and neural networks to produce the encoded vector. The decoder is a slight modification of the encoder that takes input from the encoder and previous decoder output, producing probabilities for the most likely word. Devlin et al. (2019) later introduced BERT, an encoder only architecture that transforms an input sequence into a single output sequence of the same size. This encoder only approach is suitable for set-to-set transformation, the support set is taken as input and adapted to produce the output.

The key element of the transformer architecture is attention, which is a measurement of how similar two vectors are. The benefit of attention is that it gives global knowledge of the sequence, in a language modelling context we compute an attention score a_{ij} between each query word Q_i[1] and key word K_j in the sequence, which allows discovery of related words separated across the text. Such separated words can be highly relevant to understanding the text, imagine if you were reading a book and forgot all details of a character after each page! Attention is typically calculated as the scaled dot product $\frac{Q_i \cdot K_j}{\sqrt{d}}$ between vectors Q_i and K_j, where d is the dimension of the transformer's hidden layers. Then a softmax layer is applied to normalize the sum of attention for each row i to 1

[1]In this section Q refers to the query vectors for a sequence of words, not the query set

$$a_{ij} = \frac{e^{\frac{Q_i \cdot K_j}{\sqrt{d}}}}{\sum_k e^{\frac{Q_i \cdot K_k}{\sqrt{d}}}} \tag{3.1}$$

Finally a new vector $\hat{V}_i$ representing item i is calculated, using an attention weighted sum of the value vectors V_k that represent the other words in the sequence.

$$\hat{V}_i = \sum_j a_{ij} V_j \tag{3.2}$$

This encoding of $\hat{V}_i$ in terms of all sequence items gives transformers their global understanding. Similarly we aim to use attention to give meta-learners a global understanding of the support set, using this understanding to perform adaptation similar to figure 3.1. For instance the attention between the query and the close novel class sample (see figure 3.1a) would be high, so the query could be adapted in a similar way. We propose using a transformer that adapts through calculating attention between the support set items, see figure 3.2.

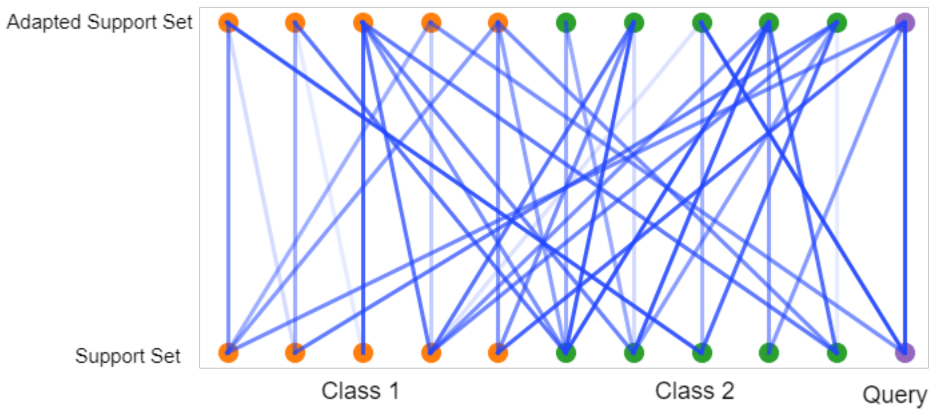


FIGURE 3.2. How Attention can be used to adapt a support set

A major limitation of attention is its computational complexity of $O(n^2)$ with respect to sequence length, computing attention requires calculating n dot products for each of the n sequence items. In recent pre-trained architectures this causes the large memory requirements that we address with knowledge distillation. It could give rise to problems when adapting support sets for large way problems, we experiment with support sets containing 295 items. However we find empirically that aggregation methods do not require as many parameters so this is not a major limitation.

Finally transformers add a positional embedding vector to the input items, to give the model understanding of the position of items in a sequence. Without this the attention mechanism is invariant with respect to position, no matter the position of items i, j both a_{ij} and the weighted sum have the same value. However for a set-to-set transformation this is a desired property. Sets simply contain items, they have no positional information and so a transformation of a support set should be immune to the order the support set is provided to the transformer. Such an argument was also made by Ye et al. (2020), who stated that $t_\theta(S)$ should provide the same result regardless of any permutation applied to S, i.e. the transformation is permutation invariant. Hence our meta-transformer does not utilise any positional embeddings.

3.2.3 Learnable Attention Weighting

While the ordering of classes in the support set is irrelevant to an adaptation, knowledge of which class a sample belongs to is essential. To execute algorithm 4 requires knowledge of classes to perform the aggregation step, hence we group the samples for each class as shown in figure 3.2. Beyond this simple case it would also be helpful for the transformation to know the relations between support set samples. For instance if one sample s_1 is very close to another sample s_2 of a different class then s_1 may be shifted away from s_2 to avoid class overlap. If both samples are from the same class however then no such action is needed.

We define a relation between samples s_1, s_2 as the common grouping that is shared by the samples. There are 6 types of relations under this definition:

(1) s_1 and s_2 are the same sample in the support set
(2) s_1 and s_2 share the same class
(3) s_1 and s_2 are from different classes, they are both in the same support set
(4) s_1 is from the support set, s_2 is the query, they are both in the same episode
(5) s_1 is the query, s_2 is from the support set, they are both in the same episode
(6) s_1 and s_2 are both the query

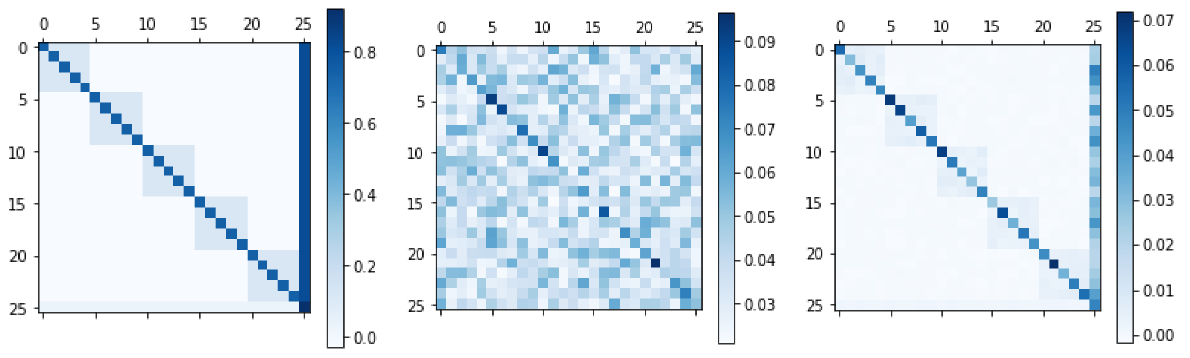
There are two main goals we wish to achieve through indicating these relationships to the transformation. The first goal is as above: to shift away from samples in different classes. Secondly the transformation should also give a higher weight to more important relationships, for instance relation (4) with the only query should have more influence on s_1 than the many samples from the same class in relation (3). We can achieve both goals through multiplying the attention values with a scalar weight $w^{(i)}$. Say i and j have relation (3), then we can calculate $\hat{a}_{ij} = a_{ij} * w^{(3)}$ where $w^{(3)} < 0$ to subtract s_j from s_i when calculating the attention weighted sum.

$$\hat{V}_i = \sum_j \hat{a}_{ij} V_j \tag{3.3}$$

This subtraction shifts s_i further from s_j. A model could also learn a $w^{(4)} > w^{(2)}$ to give more weight to the query example. In general we multiply the attention a_{ij} by the appropriate attention weight $w^{(r(i,j))}$ where $r(i,j)$ is a function that gives the relation number between items i, j.

$$\hat{a}_{ij} = a_{ij} * w^{r(i,j)} \tag{3.4}$$

In summary we define 6 scalars to weight each of the 6 relationships, these scalars can be learnt by the model to adjust to each dataset. In practice weighted attention is implemented through constructing a matrix W from the 6 scalars, then performing element-wise multiplication with the attention matrix, see figure 3.3.



(A) Attention Weight Matrix W (B) Attention Matrix a_{ij} (C) Weighted Attention Matrix $\hat{a}_{ij}$

FIGURE 3.3. Example showing the calculation of $\hat{a}_{ij}$ for a 5-way 5-shot problem. The attention weight matrix was learnt by a MetaFormer using RMSProp and weight regularisation. Note the query sample is placed in the last position of 25 and class samples are grouped.

The advantage of learning just 6 scalars is that we maintain a level of permutation invariance. The order of classes or samples within those classes is ignored again due to the summation, only the relations between samples have an impact on the output. This gives us the best of both worlds, knowledge of class information and ignoring useless positional information. An interesting property of the attention weights is they allow the model to theoretically approximate Prototypical Networks. Through learning $w^{(1)} = 1$, $w^{(6)} = 1$ and all other $w = 0$ no adaptation is performed, apart from scaling the value vector down i.e. $\hat{V_i} = a_{ii}V_i$, if this a_{ii} stays constant across most items i then the distance metric shouldn't be affected.

There are several other methods to modify the attention based on relationships using learnt parameters. Press et al. (2021) introduced an attention bias to replace the positional embedding in the language modelling context. They add a bias to the attention before softmax $\frac{Q_i \cdot K_j}{\sqrt{d}}$, with the bias becoming increasingly negative the earlier the word is in the sequence. Adding a learnt bias before the softmax in this context fails to achieve the first goal of shifting away from samples, as applying softmax results in positive a_{ij} so we cannot subtract vectors v_j from a different class. In the same vain applying a learnt bias after softmax is not ideal for fulfilling the first goal. While it is possible for $\hat{a}_{ij} = a_{ij} + b^{(3)}$ to be negative adding a bias ignores the value of a_{ij}, if a_{ij} is quite large then the bias can fulfill one of two cases. Either $b^{(3)}$ is a small value in which case $\hat{a}_{ij}$ will not be negative and the classes are not separated. Or $b^{(3)} > a_{ij}$ in which case we achieve the separation, but every other sample in relation 3 will also be subtracted. In fact the model will assign larger negative attention to samples with a low a_{ij} than a high one! This is the opposite of the desired behaviour, as samples near each other on the decision boundary should clearly have more of an effect than those near the class center that don't convey as much information. Hence we can see multiplication by a scalar is desirable for its ability to flip $\hat{a}_{ij}$ to be negative while maintaining the magnitude of a_{ij}.

Optimization

One disadvantage of using just 6 scalars to represent very important relations is that learning them is quite difficult. The first difficulty is the potential for the weights to grow quite large, resulting in large v_i. To avoid this we apply a $\tanh$ activation function to restrict the range of $w^{(i)}$ to $w^{(i)} \in (-1, 1)$, setting $w^{(i)} = \tanh(\breve{w}^{(i)})$ where $\breve{w}$ is the learnable parameter. To initialize the $\breve{w}$ vector we use Xavier initialisation (Glorot and Bengio, 2010) which was designed for the $\tanh$ activation function.

The next difficulty is that we find empirically the gradients for $\breve{w}$ are quite low, due to their multiplication with the softmaxed attention parameters which are small. The gradient of the loss function l with respect to $w^{(3)}$ is presented below to examine why.

$$\frac{\partial l}{\partial w^{(3)}} = \sum_i (\frac{\partial l}{\partial \hat{V}_i})^T \frac{\partial \hat{V}_i}{\partial w^{(3)}} = \sum_i (\frac{\partial l}{\partial \hat{V}_i})^T (\sum_j a_{ij} V_j) \tag{3.5}$$

Because we multiply by the low attention values this reduces the gradient. Additionally the mean of the elements of each vector V_j may be low, because they have a linear layer applied to them during multi head attention which may have activations centered near 0. Note this affect is not seen in standard neural networks, as here we have one weight $w^{(3)}$ multiplied by *all* elements of a vector.

These low activations mean that $\breve{w}$ only experiences slight updates when using standard gradient descent (SGD) with the same learning rate as the rest of the MetaFormer. We find $\breve{w}$ tends to remain near the initial values, with the important weight of $\breve{w}^{(3)}$ decreasing slightly. The Adam optimizer is also completely unusable, as the low gradients results in a low momentum which further lowers the gradients, the end result is a completely static $\breve{w}$. The alternative is creating a separate optimizer for $\breve{w}$. We can simply increase the learning rate of SGD to a large value such as 1, or use RMSProp which has an adaptable learning rate that increases for parameters with low gradients. Both of these successfully update the parameters, however we find that both solutions learn $\breve{w}$ as a near zero vector and performance degrades. Clearly the optimal $\breve{w}$ is not a zero vector, as this means attention is completely ignored, with $\hat{v}_i$ being set to v_i using the residual connection alone. We complete an ablation study which confirms that attention is beneficial. To counter this issue we introduce a regularisation to push each element of w toward 1, as we have prior knowledge that attention is beneficial and $w = 1$ results in the original attention mechanism. We use the mean squared error to calculate the regularisation loss as

$$\text{weight\_loss} = \frac{1}{6}\sum_{i=1}^{6}(w^{(i)} - 1)^2 \tag{3.6}$$

This regularisation gives us the attention weight matrix shown in figure 3.3 and results in a w where with high $w^{(1)}$, $w^{(4)}$ and $w^{(6)}$, low $w^{(2)}$ and other weights near 0. Across multiple runs the values of w consistently converges to this pattern, so long as the weighting of the weight_loss term is very low relative to the cross entropy loss, otherwise the parameters all converge to 1.

3.2.4 Architecture

We summarise the architecture of the MetaFormer in figure 3.4. Given a support set and query each sample is first encoded using BERT and then adapted using the set-to-set transformer. We then adopt

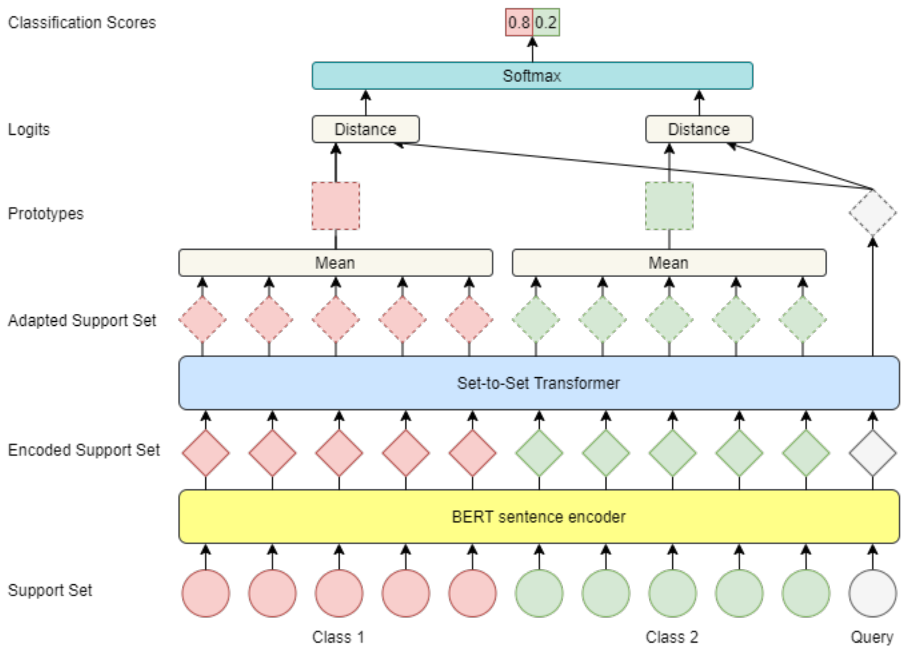


FIGURE 3.4. The overall MetaFormer architecture, shown performing inference on a 2-way 5-shot problem

the standard prototypical network setting of using the mean to aggregate each adapted sample into a prototype. The literature has several state of the art aggregation techniques, see section 2.2, however the set-to-set transformation itself also performs a form of aggregation through the weighted attention mechanism and so an alternate complex technique is not needed. For the scoring method we adopt negative euclidean distance $\hat{y}_i j = -||Q_i - P_j||_2$, taking the negative so that closer samples have higher logits. We find cosine similarity tends to give very similar results near 1 for each prototype and so it requires tuning a temperature τ to scale the logits $\tau \hat{y}_i j$. Finding the optimal value of τ is difficult and so we use euclidean distance which doesn't experience this problem.

The Set-to-Set transformer is the encoder only architecture discussed in section 3.2.2. We adopt several modifications from Ye et al. (2020) to the standard transformer architecture to simplify it as we find the adaptation need not be overly complex. These modifications are shown in figure 3.5.

The main changes when compared to the standard transformer encoder are the dense network being replaced with a simple linear layer and the removal of a residual connection between the attention and linear layers. The goal of this simpler encoder architecture is to focus on the weighted multi head attention. After said attention layer is applied the linear layer then weights how much $\hat{V}$ contributes to the final encoding relative to the residual V.

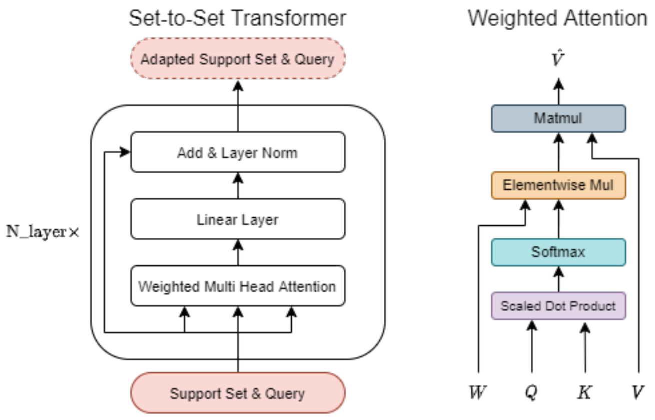


FIGURE 3.5. Architecture of the Set-to-Set Transformer with Weighted Attention

Multi-Head Attention involves separating the dimensions of the vectors Q, K, V into n separate vectors of size $\frac{d}{n}$, applying a linear layer and calculating the attention weighted sum for each separately, then concatenating each vector to form the final $\hat{V}$. The purpose of this is to allow each head to learn an attention mechanism that focuses on different information. For instance the relation embeddings V combine multiple different attributes of the relation into one vector. Perhaps one attention head may focus on the location aspect, giving high attention to relations that contain similar locations, whereas another head might give high attention to relations that are written in a similar style e.g. academic, informal. Weighted Multi-Head Attention is similar to Multi-Head Attention, except each head learns separate attention weights $\breve{w}$, which increases the uniqueness of each head. In addition to choosing which aspect of a vector to focus on these heads can also learn a unique adaptation.

When testing with multiple layers in the transformer the block shown on the left of figure 3.5 is simply repeated for each layer. Each layer also has separate attention weights so that they can learn a unique adaptation, perhaps earlier layers may have more equal attention weights to incorporate information from the support set and later layers where only slight adjustment is needed can utilise a higher $w^{(1)}$ and $w^{(6)}$.

3.2.5 Regularisation

Regularisation techniques for meta learners encourage desired clustering properties that we want each class in the support set S to have. The regularisation used by Ye et al. (2020) aims to minimize the intra class distance, i.e. the euclidean distance between the prototypes P and the support and query set samples of that class before adaptation. They form these prototypes through taking the mean of both that episode's support and query set samples, we modify this to match the objective, only using the support set to form prototypes.

Algorithm 5 Intra Class Regularisation Loss Calculation

```
Input: $S$, $Q$, $N$, $K$, A Set-to-Set Transformation $t_\theta : S \to \hat{S}$, Sample embedding function $g_\phi : x \to \hat{x}$,
    Aggregation function $agg : S \to \mathbb{R}^{\text{hidden size}}$
Output: intra_loss
    $S \leftarrow \{(g_\phi(x), y) | \forall(x, y) \in S\}$
    $Q \leftarrow \{(g_\phi(x), y) | \forall(x, y) \in Q\}$
    $\hat{S} \leftarrow t_\theta(S)$
    intra_loss $\leftarrow 0$
    for $i \in \{1, 2, ..., N\}$ do
        $P_i \leftarrow agg(\{x | (x, y) \in \hat{S}, y = i\})$
        $E \leftarrow \{x | (x, y) \in S \cup Q, y = i\}$        ▷ all samples with class i from this episode
        intra_loss $\leftarrow$ intra_loss $+ \sum_{x \in E} ||P_i - x||_2$
    end for
```

The strength and weakness of this regularisation is that the minimiser of $\arg\min_{P_i} \sum_{x \in E} ||P_i - x||_2$ is simply the mean of all samples in E, in fact this is the motivation behind prototypical networks (Snell et al., 2017). The strength is that it acts as a good prior knowledge as prototypical networks perform well, the weakness is it doesn't encourage learning a better aggregation technique. To accomplish this we also add the regularisation introduced by Yu et al. (2020) which minimizes the intra class distance of the adapted $\hat{S}$ and maximises the inter class distance between the prototypes P, as an ideal adaptation should separate the classes. This loss is calculated using euclidean distance for the intra class distance and cosine similarity for the inter class distance, note through minimizing cosine similarity we maximise inter class distance.

$$\text{cluster\_loss} = \frac{1}{NK} \sum_i \sum_{x \in \{x | (x,y) \in \hat{S}, y=i\}} ||P_i - x||_2 + \frac{1}{N} \sum_i \sum_j \frac{P_i \cdot P_j}{||P_i|| ||P_j||} \tag{3.7}$$

We then combine these two losses and the cross entropy loss to get the overall loss, using hyperparameters α and β to weight the regularization losses. The weight_loss term is added when training using separate optimizers for the weights.

$$\text{loss} = \text{cross\_entropy\_loss} + \alpha(\text{cluster\_loss} + \beta \text{intra\_loss}) \tag{3.8}$$

Finally we introduce a regularisation unique to learnable adaptation architectures. The MetaFormer aims solely to learn an adaptation of the support set, theoretically it should be able to improve the aggregation of support set vectors from any encoder. Hence when training the MetaFormer we do not update the encoders weights, instead initializing the BERT encoder with one learnt by another meta-learner. This allows the training process to focus on learning a better adaptation that is encoder agnostic, and prevents overfitting to training data using the encoder's parameters.

3.3 Knowledge Distillation

We use knowledge distillation to reduce the memory requirements of the sentence encoder relative to BERT. This allows us to perform larger way training e.g. 40-way training that we hypothesis should improve performance on the large 59-way task we evaluate on. Snell et al. (2017) found this approach of increase the training way to be effective, even for smaller 10-way tasks, so we wish to observe whether this effect applies to our architecture.

Our application of knowledge distillation is straightforward, we simply mirror the process outlined by Jiao et al. (2020) to train a version of TinyBERT for meta-learning. In the knowledge distillation for for BERT literature there is a trend of performing both general pretraining and then task specific fine-tuning, mirroring the training process for full sized BERT. We give an overview of this process in figure 3.6

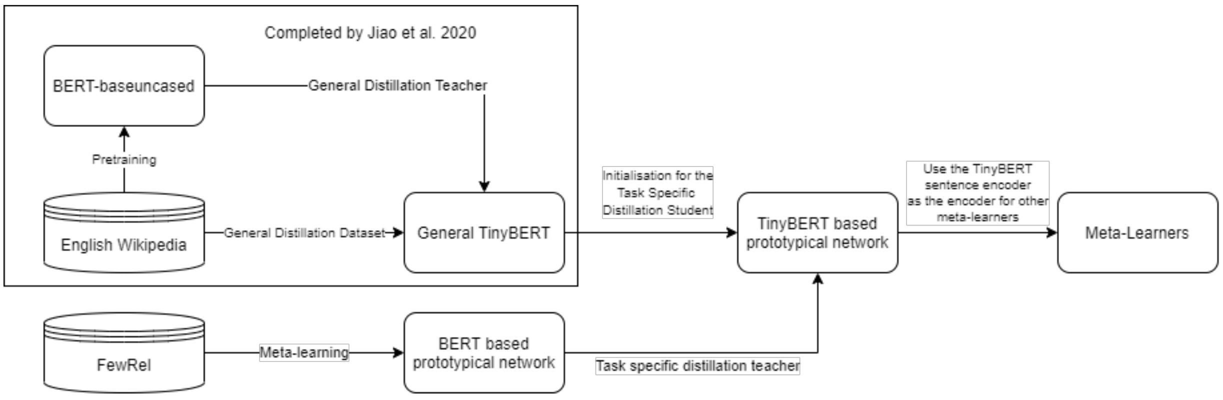

Completed by Jiao et al. 2020
BERT-baseuncased
General Distillation Teacher
Pretraining
English Wikipedia
General Distillation Dataset
General TinyBERT
Initialisation for the Task Specific Distillation Student
TinyBERT based prototypical network
Use the TinyBERT sentence encoder as the encoder for other meta-learners
Meta-Learners
FewRel
Meta-learning
BERT based prototypical network
Task specific distillation teacher

FIGURE 3.6. The Knowledge Distillation process used to train a TinyBERT for meta-learning on the FewRel dataset

General knowledge distillation refers to performing knowledge distillation using a large language modelling task, this results in the General TinyBERT shown in the Figure, we reuse the distilled General TinyBERT provided by Jiao et al. (2020). Task specific knowledge distillation involves performing knowledge distillation on the dataset we wish to use TinyBERT on, in this case FewRel. This step is vital as just like BERT the student model requires fine tuning on specific datasets to achieve the best performance. To perform knowledge distillation we first train a prototypical network with BERT as the encoder, fine-tuning BERT for the FewRel dataset. Then we use this BERT based prototypical network as the teacher, and a TinyBERT based prototypical network as the student to perform knowledge distillation on FewRel. The TinyBERT sentence encoder is first initialised with the General TinyBERT parameters and then used in a prototypical network that is trained for 30000 epochs using the knowledge distillation loss in figure 2.8. Finally we store the trained TinyBERT as a pretrained language model, that can be used as the sentence encoder for other Meta-Learners.

CHAPTER 4

Experiments

Measuring the performance of a meta-learner is made more difficult by the effects of randomly selecting a support set. Additionally we must also consider how to construct the incremental learning task. This chapter describes the procedures and dataset used to reliably estimate a meta-learner's performance in the incremental and standard settings.

4.1 Evaluation Setup

All models are trained in the standard episodic fashion, where numerous 10-way 5-shot episodes are sampled and the loss on their query set used to update the model. In later ablation studies we explore training on larger way problems such as 20-way 5-shot episodes. The key difference in our work is that we evaluate on the incremental setting, as well as the standard episodic setting. In the standard setting a model is trained on N-way K-shot episodes sampled from base classes, then tested on N-way K-shot episodes sampled from novel classes not seen in training.

In the incremental setting a model is trained on N-way K-shot episodes sampled from base classes, and then tested on episodes that include all the base classes and some novel classes. In this setting accuracy is separated into accuracy on base classes and accuracy on novel classes to observe how well models adapt to the novel classes independently of base class performance. The base class accuracy estimates $P(y = F_\theta(x)|(x, y) \in \bigcup B)$ and the novel class accuracy estimates $P(y = F_\theta(x)|(x, y) \in \bigcup NC)$. We also measure the standard accuracy over all classes, $P(y = F_\theta(x)|(x, y) \in \bigcup(B \cup NC))$. Though this measure is mostly determined by base-classes as they are more numerous, it represents the overall model performance.

The estimation of the probabilities such as $P(y = F_\theta(x)|(x, y) \in \bigcup NC)$ is more difficult for a meta-learner compared to a supervised model as F_θ requires a support set S in order to make a prediction for

x. This estimation is done through episodic evaluation, described in section 2.1.4. This involves randomly sampling numerous episodes each with different support, then calculating the number of correct predictions, see algorithm 6.

Algorithm 6 Episodic Evaluation in the Incremental Setting

```
  UniformSample($S, n$) uniformly samples a set of $n$ items from the set $S$ without replacement
  $\delta_{x=y}$ is a function equal to 1 if $x = y$, otherwise 0
Input: $N_b$, $N_{nc}$ $K$, $B$, $NC$, niter, A meta learner $F_\theta : S, Q \rightarrow Y$
Output: Accuracy $a$
  correct $\leftarrow 0$
  total $\leftarrow 0$
  for $i \in \{1, 2, ..., \text{niter}\}$ do
      $E \leftarrow$ UniformSample($B, N_b$) $\cup$ UniformSample($NC, N_{nc}$)
      for $j \in \{1, 2, ..., N\}$ do
          $S_j \leftarrow$ UniformSample($E_j, K$)
          $Q_j \leftarrow$ UniformSample($E_j \setminus S_j, K$)
      end for
      $S \leftarrow \{S_1, S_2, ..., S_N\}$
      $Q \leftarrow \{Q_1, Q_2, ..., Q_N\}$
      correct $\leftarrow$ correct $+ \sum_{\hat{y} \in F_\theta(S,Q)} \delta_{y=\hat{y}}$
      total $\leftarrow$ total $+ N * K$
  end for
  $a \leftarrow \frac{\text{correct}}{\text{total}}$
```

The aim of episodic evaluation is that through sampling thousands of episodes the effect of the randomly chosen support set S on the accuracy will be reduced. These effects are especially large in the 1-shot case, where the correct class whose average example is closest to the query may instead have an outlier sample that is far from the query chosen to form the support set, in which case the model would instead predict another class and be wrong due to random effects. In the 5-shot case these effects are less extreme, but in general if an outlier is included in the support set the class means may be distorted enough to change the prediction. Through sampling several episodes we can measure the expected performance of the meta-learner across the distribution of support sets, where we expect outliers in the support set to be rare.

When calculating separate accuracy for novel classes we simply add to algorithm 6 the constraint that Q_i only includes examples from E_i if $E_i \in NC$ in order to examine queries in the desired class set only, see algorithm 7. A similar process is done for the base classes.

Following the evaluation setting of Ren et al. (2020) we perform experiments with $N_b = 54$ and $N_{nc} = 5$. In addition we also present results for a 10 and 5 way version of the incremental setting. Here classes

Algorithm 7 Episodic Evaluation of accuracy on Novel Classes

```
Input: $N_b$, $N_{nc}$, $K$, $B$, $NC$, niter, A meta learner $F_\theta : S, Q \rightarrow Y$
Output: Accuracy $a$
  correct ← 0
  total ← 0
  for $i \in \{1, 2, ..., \text{niter}\}$ do
      $E \leftarrow$ UniformSample$(B, N_b) \cup$ UniformSample$(NC, N_{nc})$
      $Q \leftarrow \{\}$
      for $j \in \{1, 2, ..., N\}$ do
          $S_j \leftarrow$ UniformSample$(E_j, K)$
          if $E_j \in NC$ then
              $Q \leftarrow Q \cup$ UniformSample$(E_j \setminus S_j, K)$
          end if
      end for
      $S \leftarrow \{S_1, S_2, ..., S_N\}$
      correct ← correct + $\sum_{\hat{y} \in F_\theta(S,Q)} \delta_{y=\hat{y}}$
      total ← total + $|Q| * K$
  end for
  $a \leftarrow \frac{\text{correct}}{\text{total}}$
```

are purely randomly sampled i.e. $E \leftarrow$ UniformSample$(B \cup NC, N)$, so that we don't have to always include 1 novel class in each 5 way problem which would result in novel classes taking up 20% of the episode. This setting better matches a real world environment where novel classes may or may not be present. We summarise the three experimental settings in table 4.1.

Setting	Episode Sampling Method	N-way	K-Shot	Purpose
Standard	$E \leftarrow$ UniformSample(NC, N)	$10, 5$	$5, 1$	Measuring standard meta-learning performance
Incremental	$E \leftarrow$ UniformSample$(B, N_b) \cup$ UniformSample(NC, N_{nc})	59	5	Measuring Incremental learning performance using a consistent number of base classes
Incremental random	$E \leftarrow$ UniformSample$(B \cup NC, N)$	$10, 5$	$5, 1$	Measuring incremental learning performance on standard 10 and 5 way problems

TABLE 4.1. Outline of Experimental Settings

Of note is that the performance of meta-learners has been found by Dhillon et al. (2020) to have a large variance, as their predictions heavily depend on the randomly chosen support set. Reducing the variance of meta-learners is difficult as they must depend on the support set in order to perform inference on unseen tasks, and the support set itself has large variance due to its small size. While this problem is beyond the scope of this research we can increase the number of testing episodes to 10000 to reduce the variance of our estimate of the mean performance.

We also take into account the suggestions made by (Musgrave et al., 2020), who outline a number of flawed experimentation methodologies that have been used to show performance improvements in metric learning when there are none. While they discuss metric learning techniques outside of the context of meta-learning their suggestions are still of use. The most relevant suggestions are to keep the backbone model consistent for all techniques, test models multiple times, present confidence intervals of the accuracies and avoid using test set feedback for training. The later can be especially tricky, as they found some techniques used the test set to perform early stopping in training, a technique we also found in some papers' implementations. We address these concerns by testing all models using BERT and TinyBERT, testing across 10000 episodes and using a separate validation set to perform early stopping. Additionally we train each model 3 times and report the average accuracy across each run.

4.2 Datasets

FewRel (Han et al., 2018) is the main dataset used to evaluate meta-learners in NLP and so we evaluate our methods on it in both the incremental and episodic setting. The FewRel dataset contains 100 classes each with 700 examples. It is split into predefined train, validation and test splits with 64, 16 and 20 classes respectively. However the test data is private and can only be evaluated on in the standard episodic fashion. For the standard setting we train a model on the provided train and validation split. Then it is evaluated using the private test data, where accuracy is measured for all 4 combinations of $N \in \{5, 10\}$ and $K \in \{1, 5\}$. For the incremental setting (Ren et al., 2020) introduced a train, test and validation split that contains a set of 10 novel relations for validation, 16 for testing and 54 base relations with data samples split across training, validation and testing datasets (see figure 4.1). (Ren et al., 2020) performs testing by including samples from all 54 base relations and samples from 5 randomly chosen classes from the 16 test classes. This approach accurately models the full classification problem in incremental learning, where a model must classify across all base classes and a few novel classes not seen in training. Hence we adopt their dataset and approach to train and test all incremental models and report accuracies on the 59-way task.

To create the surrogate novel class dataset we utilise an existing implementation[1] whose code we modify to our purposes. We apply matching the blanks to the large Wikitext-103 dataset[2] (Merity et al., 2017) which contains 103 million tokens of text extracted from Wikipedia. The size of the dataset means we

[1]Available from https://github.com/plkmo/BERT-Relation-Extraction

[2]Available from https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/

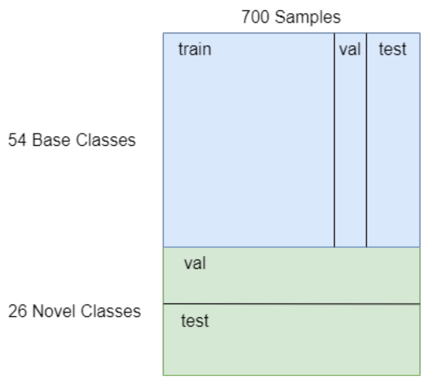


FIGURE 4.1. Under the incremental setting the dataset is split into base and novel classes. The base classes are then split by samples and the novel classes split by class to form the training, validation and testing sets. Both base and novel classes are included in the validation and testing sets, using unseen base class examples to measure generalization performance.

can extract 32000+ classes that each have at least 10 examples. In practice we need more then 32000 classes so that we can examine the effects of including multiple surrogate novel classes in training. So we sequentially cycle through each class in this surrogate novel set, repeating the cycle as needed. This way class repetition is split greatly over thousands of iterations, which should prevent overfitting.

4.3 Baselines

We compare our method against a variety of strong baselines. First we compare with the basic prototypical network (Snell et al., 2017) that inspired the architecture of many later metric learners. We choose two models that have performed well on the FewRel benchmark. One being Matching the Blanks (MTB) (Baldini Soares et al., 2019) which introduced the relation encoding method reused by several others along with a self-supervised pretraining objective, it achieves the best performance of all published methods. Hybrid Prototypical Attention Networks (HATT) (Gao et al., 2019) are examined for their novel aggregation method which utilises instance and feature level attention to aggregate relations.

We also test against two models that have been applied to incremental meta-learning. The incremental prototypical network (IncreProtoNet) introduced in (Ren et al., 2020) uses an attention mechanism to combine the embeddings of a standard backbone and another backbone specifically trained to embed novel classes, with the aim of reducing the overlap between base and novel embeddings. Note that their model is designed in such a way that it requires novel classes to make any predictions, so we do not compare to this on non-incremental settings, or the 5 or 10-way settings where novel classes may not be included in the support set. The Few Shot Embedding Adaption with Transformers (FEAT) model (Ye et al., 2020) uses a transformer to perform adaptation of the class means.

4.4 Backbones

The backbone of a meta-learner refers to the underlying model used to embed the data, in this case a relation triplet, into a vector. The embedded vectors of the support set are then used by the meta-learner to perform inference. We adopt the entity markers and entity start state method developed by (Baldini Soares et al., 2019) and used by several other models (Dong et al., 2020; Yu et al., 2020). This method adds tokens to indicate the position of entities in the sentence and then extracts a representation from BERT by concatenating the embedding of the marker tokens, see figure 2.1 for further discussion.

Following the standard in NLP meta-learning we use BERT as the backbone in the standard $N \leq 10$-way setting. However in the incremental settings we encounter memory constraints, see figure 4.2. This is because the entire support set is encoded by BERT to perform inference on a query set, meaning for inference on the 59-way 5-shot incremental task we have a batch size of $59 * 5 = 295$ samples that must be encoded, not including the query set itself which can have any size.

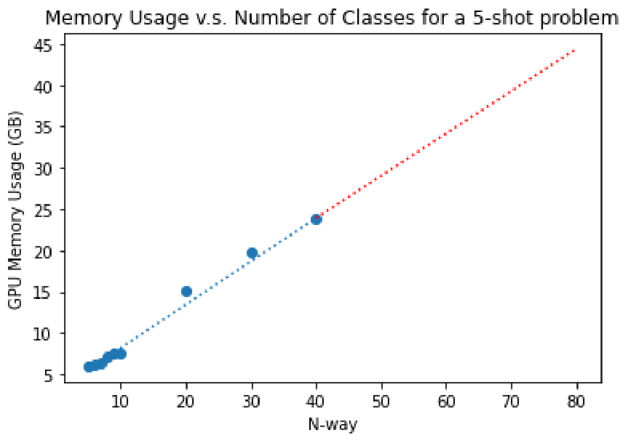


FIGURE 4.2. The large batch size required for inference on 59-way problems is beyond our server's capability. The red line indicates extrapolation

This problem is somewhat unique to meta-learning in NLP as the dominant large pretrained models have high memory requirements. (Ye et al., 2020) simply used a CNN with lower memory requirements in their experiments on computer vision datasets. (Ren et al., 2020) performed experiments on the FewRel dataset but also used CNN as their backbone, presumably because of memory issues. To avoid the limitations of CNNs in NLP, namely their inability to examine global data, we use knowledge distillation, replacing BERT with TinyBERT (Jiao et al., 2020) as outlined in section 3.3.

These memory limitations mean *training* on large way problems with BERT with limited memory is impossible, as the activations for each sample in an episode must be stored to calculate gradients and perform gradient descent. However *inference* on 59 way problems is possible through encoding the 295 support set samples in batches, as the intermediate activations need not be stored. However this method is quite time consuming, hence we evaluate only the critical experiments using BERT as well.

4.5 Hyperparameters

The experiments are conducted with consistent hyperparameters across each model where possible. Each model is trained over 30,000 training iterations where the model with the best validation performance is chosen, then tested over 10,000 test iterations. We use a learning rate of 0.0002 and dropout rate of 0.5 for all models except MetaFormer which has a frozen encoder and so benefits from a higher learning rate of 0.001. Adam optimization is used for the baslines, MetaFormer and FEAT both benefit from a standard gradient descent optimizer for learning the adaptation parameters. For the MetaFormer regularisation we set $\alpha = 0.01$ and $\beta = 10$. IncreProtoNet incorporates a CNN and complex attention mechanism into its architecture and so requires very different hyperparameters, we use the hyperparameters outlined by (Ren et al., 2020).

The MetaFormer freezes the encoder while training, so we use a TinyBERT from a previously trained Prototypical network to initialize the encoder. For the BERT experiments we first train prototypical networks on the same setting, saving their encoder and using it to train MetaFormer. Through hyperparameter tuning we find that using a single optimizer works better than the separate weight optimizers and that surrogate novel classes have little impact, which we show in ablations. Hence in all experiments we train using a single optimizer and no surrogate novel classes.

CHAPTER 5

Results

This chapter presents the results of our experiments in all three settings: Incremental, Incremental random and Standard, finding MetaFormer to be superior to FEAT and on par with Prototypical Networks. We then examine the affects of removing or modifying significant parts of the model in the ablation study, providing analysis on some surprising results. Finally we observe the different values for the learnt attention weights each variant of the MetaFormer learns.

5.1 Incremental Setting

TABLE 5.1. Accuracy of Models with a TinyBERT encoder in the Incremental Setting. We report overall accuracy in the 59-way 5-shot incremental setting along with the separate accuracies for samples from the base classes and those from the novel classes

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	59-way 5-shot	Base Classes	Novel Classes
Prototypical Networks	90.69	95.61	85.19	92.12	80.35	81.94	63.17
HATT	90.54	94.75	85.03	90.63	76.68	77.78	64.82
MTB	90.66	94.62	85.23	90.53	76.6	77.82	63.45
TwoPhase Prototypical	-	-	-	-	80.27	81.73	64.53
FEAT	90.56	95.5	85.33	91.95	80.15	**81.99**	60.28
MetaFormer	**91.14**	**96.02**	**85.71**	**92.71**	**80.63**	81.95	**66.38**

First we examine the performance of the MetaFormer compared to the baseline models in table 5.1. Interesting to note is the performance drop between the 10-way 5-shot and 59-way 5-shot setting is only about 12 accuracy points, despite the number of classes increasing near 6×. This is perhaps caused in part by the nature of the dataset, there are few similar classes and so it is rare that new classes introduced into an episode are near the query sample. It also indicates that the embedding function of even basic meta-learners such as prototypical networks is complex enough to separate several more classes than it was trained to.

Overall we see that FEAT, Prototypical networks and MetaFormer are the top performers, with MetaFormer achieving a far higher accuracy in the novel classes. This is due to two factors, first we can see the overall meta-learning performance is higher on the base classes because of the learnt adaptation, second this is achieved even while having strong regularisation which increases the generalisation ability. Note that despite Matching the Blanks (MTB) achieving the top published performance it falls behind here, this occurs because no pretraining is used and its encoding method is shared across all models, so MTB's main contributions have no impact. Also note that the code for IncreProtoNet requires each episode contain at least one novel class, which is not guaranteed for the 5 and 10 way settings, hence we do not evaluate it on these settings.

TABLE 5.2. Accuracy of Models with a BERT encoder in the incremental setting

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	59-way 5-shot	Base Classes	Novel Classes
Prototypical Networks	**93.23**	**97.34**	88.89	**94.93**	**85.55**	86.61	**74.05**
FEAT	92.41	96.38	88.17	93.56	84.27	86.07	64.75
MetaFormer	**93.2**	**97.33**	**89.0**	94.88	**85.58**	**86.76**	72.88

We then evaluate the performance of the second best model Prototypical Networks, a similar model FEAT and MetaFormer using a BERT encoder on the incremental setting. MetaFormer is initialised with the BERT encoder trained using Prototypical Networks, so to ensure a fair comparison we also initialise FEAT with this encoder. Here the performance of MetaFormer is similar to Prototypical Networks, but superior to the alternate transformer based meta-learner, especially in novel class performance. When using BERT as an encoder the embedding size increases from 624 for TinyBERT to 1476, perhaps this increase in model complexity renders the additional complexity offered by MetaFormer's adaptation mute.

5.2 Standard Setting

TABLE 5.3. Accuracy of Models with a BERT encoder in the Standard Setting

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot
Prototypical Networks	89.73	96.21	83.36	93.32
FEAT	**90.55**	96.08	**84.38**	92.97
MetaFormer	89.79	**96.32**	83.46	**93.42**

Finally the three models are tested using a BERT encoder in the Standard Setting. With the novel classes removed FEAT's main area of difficulty is removed and its performance increases, especially in the 1-shot case. The relative increase in the 1-shot performance compared to 5-shot performance can be attributed to FEAT's averaging of the support set before adaptation. We've shown this to be disadvantageous theoretically and experimentally (see ablations) however in the 1-shot case this has no impact, as there aren't multiple examples to average. Without this major limitation FEAT's direct manipulation of the class prototypes may be the cause of the improvement. In contrast MetaFormer can only indirectly manipulate class prototypes through shifting each one of the support set samples in similar directions. Again we see similar performance between Prototypical Networks and MetaFormer, likely due to the model complexity issue we discussed.

5.3 Ablations

This section examines which parts of the model are effective through ablation, removing each module one by one and measuring the performance impact.

TABLE 5.4. Accuracy of Ablated models using the TinyBERT encoder in the Incremental Setting. The bold highlights indicate the best performing method of each section

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	59-way 5-shot	Base Classes	Novel Classes
Original MetaFormer	91.106	96.034	85.634	92.7	80.598	81.916	66.354
Attention							
- attention weights removed	91.02	96.02	85.58	92.67	80.6	81.91	**66.43**
- attention removed completely	**91.3**	**96.08**	**85.84**	**92.77**	**80.63**	**81.98**	66.05
- cosine similarity attention	91.26	96.02	85.6	92.66	80.6	81.93	66.33
- euclidean distance attention	91.14	96.07	85.74	92.71	80.61	81.95	66.15
Regularisation							
- remove FEAT regularisation	91.16	96.05	85.49	92.65	80.6	81.92	66.31
- remove MPE regularisation	91.07	95.97	85.49	92.69	80.58	81.84	67.01
- remove All regularisation	**91.34**	**96.09**	**85.78**	**92.74**	80.62	**81.96**	66.16
- remove Surrogate Novel Classes	91.14	96.02	85.71	92.71	**80.63**	81.95	**66.38**
- remove Frozen Encoder	91.07	96.01	85.67	92.66	80.47	81.87	65.26
Architecture							
- Query adaptation removed	90.96	95.82	85.22	92.45	80.31	81.51	**67.3**
- Pre-Averaging	**91.19**	**95.97**	**85.63**	**92.64**	**80.57**	**81.89**	66.28
Optimization							
- SGD Attn Weight Optimizer	91.18	96	85.55	**92.7**	**80.61**	**81.91**	**66.5**
- RMSProp Attn Weight optimizer	**91.21**	**96.02**	**85.66**	92.69	80.56	81.87	66.45

Note that in table 5.4 the original metaformer is shown with surrogate novel classes so that the effect of ablating surrogate novel classes can be seen. However the other ablations do not use surrogate novel classes as they have a negative effect and increase the training time.

There are three primary sections to ablate: the attention mechanism, regularisation and architecture. With the attention we test different methods of attention such as cosine similarity or euclidean, removing the attention weights and even removing attention entirely. We notice a common trend throughout the ablation study which is the low impact of these modifications, only slight changes are observed such as the decrease in novel class performance when attention is removed. This is likely due to using the underpowered TinyBERT encoder, so we repeat key adaptations using BERT as well. We tested cosine similarity and negative euclidean distance as attention functions, which may be better for identifying similarity between samples. For instance scaled dot product attention considers the vectors $[1, 1]$ and $[4, 1]$ to have higher attention then between $[1, 1]$ and $[1.1, 1]$, where clearly the latter two a closer and more likely to be part of the same cluster. Cosine similarity and euclidean distance give high attention to the latter pair, but there is no improvement, perhaps this intuition doesn't carry over to higher dimensions where the scale of most activations is small and the difference between cosine similarity and scaled dot product is minimal.

In regularisation removing all regularisation hurts the novel class performance, and removing the frozen encoder (i.e. training BERT as well) hurts performance across the board. The value of query adaptation is clear, removing it hurts performance everywhere except the novel classes, perhaps removing this decreases variance at a cost to increased bias and so the generalisation to completely unseen classes slightly improves. Pre-Averaging refers to calculating the class mean before adaptation, as FEAT does, as opposed to post averaging where the means are calculated after adaptation.

The optimization section shows the impact of using separate optimizers for the attention weights is minimal. This is surprising given that these optimizers consistently converge to optimal values for w, as opposed to the original optimizer which essentially leaves the weights at their random initialization values. This can be seen through comparing the standard deviation of the weights shown in table 5.5. This consistent convergence should mean these values of w better minimize the loss function compared to the random values of w learnt by the MetaFormer, but the performance does not increase. Perhaps this is caused by a misaligned regularisation term, where minimizing the regularisation loss doesn't improve performance. We did observe in table 5.4 that removing all regularisation losses had only a small affect on novel class performance, and slightly increased performance in other areas.

The contribution of key areas of our model, Post Averaging and Attention Weights, are not clear here so we repeat these experiments using BERT to verify this result.

TABLE 5.5. Attention Weights Learnt by separate optimizers with the weight_loss regularisation. The average and standard deviation over 3 runs is reported.

Optimizer	$w^{(1)}$	$w^{(2)}$	$w^{(3)}$	$w^{(4)}$	$w^{(5)}$	$w^{(6)}$
Original MetaFormer	0.46 ± 1.44	0.76 ± 0.56	0.17 ± 0.54	-0.27 ± 1.36	-0.23 ± 0.27	0.67 ± 0.7
SGD	0.58 ± 0.03	0.06 ± 0.03	-0.01 ± 0.01	0.76 ± 0.06	-0.01 ± 0.01	0.71 ± 0.02
RMSProp	0.73 ± 0.04	0.09 ± 0.04	-0.01 ± 0.02	0.88 ± 0.05	0.02 ± 0.01	0.87 ± 0.05

TABLE 5.6. Accuracy of key Ablated models using the BERT encoder in the Incremental Setting

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	59-way 5-shot	Base Classes	Novel Classes
MetaFormer	**93.2**	**97.33**	**89.26**	**94.88**	**85.58**	**86.76**	72.88
- Attention Weights removed	92.87	**97.3**	88.64	**94.86**	85.31	86.39	**73.64**
- Pre-Averaging	92.95	97.14	88.55	94.63	84.9	86.12	71.74

The BERT experiments show a clearer trend in the two important ablations. We can clearly see the advantage of post averaging, with the pre-averaging model being substantially worse in the 59-way task. Interestingly the MetaFormer without attention weights performs similarly, trading base class performance for novel class. We did find learning the attention weights to be a major difficulty, perhaps this is because their impact is small.

TABLE 5.7. Effects of tuning the architecture of a MetaFormer using TinyBERT on the Incremental Setting

Model	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	59-way 5-shot	Base Classes	Novel Classes
Number of Heads							
- 1 head	91.14	96.02	**85.71**	**92.71**	**80.63**	**81.95**	**66.38**
- 2 heads	**91.23**	**96.06**	85.65	92.70	80.58	81.9	66.35
- 4 heads	91.14	94.73	85.54	**92.71**	80.62	81.94	66.34
- 8 heads	91.04	96.02	85.42	92.70	80.54	81.86	66.3
Number of Layers							
- 1 layer	91.14	96.02	85.71	92.71	**80.63**	**81.95**	**66.38**
- 2 layers	**91.41**	96.04	85.68	**92.73**	80.53	81.94	65.26
- 4 layers	91.25	**96.10**	**85.74**	92.69	80.53	81.94	65.31
Different way training							
- 5 way	91.13	**96.10**	85.68	**92.71**	80.54	81.86	66.32
- 10 way	91.14	96.02	**85.71**	**92.71**	**80.63**	**81.95**	66.38
- 20 way	**91.28**	96.08	85.51	92.67	80.62	81.93	**66.64**

Additionally we perform tuning of MetaFormer's important hyperparameters, examining the effects of using multiple heads, layers and different way training. Overall we find that increasing the complexity of the MetaFormer is not necessary. Increasing the number of heads has little impact on performance and additional layers hurt performance on the novel classes substantially, dropping an entire accuracy point. The main improvement of the MetaFormer is its attention weighted sum of the entire support set, so perhaps this result indicates even a simple consideration of the entire support set is sufficient to improve adaptation.

Increasing the number of classes used in each episode was one of the motivations behind using knowledge distillation, however we can see even with 5-way training the model can still generalise to much larger problems. This could be because both 5-way and 10-way settings are very similar, coming from the same dataset. The most difficult part of meta-learning could be classifying between the few classes that are very close to the query, other relations being easy to identify as different. In this case decreasing to the 5-way problem would still leave several challenging episodes with similar classes to distinguish between, so having a very similar problem to the 10-way task would result in models with similar performance. An additional cause could be the BERT encoder being already learnt by a prototypical network on the 10-way task, in this experiment the encoder was a controlled variable but another experiment could train separate encoders to view the effects. However we do see the both models improve over the prototypical network in table 5.1, so there is learning occurring that is not impeded by the reduced way.

CHAPTER 6

Discussion

This chapter discusses the main trends in the results and surprising results, including the phenomena surrounding random attention weights and the interesting advantages FEAT has over MetaFormer. We analyse the strengths and weaknesses of the study and outline interesting avenues for future work.

The first major trend we can observe in all experiments is that base v.s. novel class performance is a trade off. We can see in table 5.4 removing the query adaptation hurts performance across the board but improves novel class performance, perhaps by having a reduced variance but higher bias. This is again seen when removing attention weights, which reduces the variance and increases the novel class performance with both BERT and TinyBERT. In general across most ablations we see that one model achieves the best base class performance and a different one achieves higher novel class performance. This tradeoff is similar to the bias-variance tradeoff observed in standard machine learning, where a model with higher variance performs well on the training set but worse in testing whereas one with slightly higher bias but lower variance could do better in testing. Here we have models with high variance performing well on unseen examples of known base classes, but worse on unseen novel classes.

The second major trend that affected the quality of experiments is knowledge distillation. It hurt the study slightly, with even influential ablations such as switching to pre-averaging having little impact on the TinyBERT MetaFormer's performance. We needed to repeat some experiments with BERT to clearly examine the benefits of the important modules. Knowledge distillation could also be responsible for the higher performance of MetaFormer when using TinyBERT because freezing an encoder prevents it from "forgetting" the weights it learnt during the distillation process that allow it to emulate BERT. Knowledge distillation was beneficial in greatly reducing the computational resources required. However the accuracy decrease is somewhat larger than in other papers, as we don't continue to perform distillation when training meta-learners. Knowledge distillation also allows for larger way training, however this appears to not have much of an impact, our model has little problem adapting to problems with a larger N-way than seen in training. Overall given sufficient time it would be better to repeat all other experiments using the BERT encoder, though we were limited by available resources.

Regarding our methodology we notice that training with Surrogate Novel Classes has very little effect on the model. The cause is clear: the only existing procedures for creating these surrogate novel classes in a self-supervised method involve masking. The two self-supervised meta-learning task creation methods we are aware of both utilise masking, replacing a word with the "[MASK]" token, a common word shared between two pieces of text (Baldini Soares et al., 2019; Bansal et al., 2020b). This is a natural construction, as two sentences that share common words are likely to describe similar things and so could loosely be considered belonging to the same class. Masking is required to prevent a model from simply identifying the shared common word, encouraging it to look at semantic similarities in texts to predict which support set examples share the word with the query. However the original FewRel dataset has no "[MASK]" tokens in it, so a model can simply identify a surrogate novel class by the fact it contains a mask token, hence learning no useful knowledge. The only solution is to perform masking similar to how Matching the Blanks does to make both sets of classes indistinguishable. However this involves masking all entities in the FewRel dataset, these entities make up a large amount of each relation and so much important information is lost, under this setting MetaFormer's performance further decreases. In future work we should examine creating a self-supervised task that doesn't require masking, perhaps through using a form of data augmentation on existing relations that creates a new class but can still preserve the class similarity label. For instance we could take an existing relation, then use a language model to negate it, creating a new negated relation class.

Intriguingly we find that a MetaFormer with random weights tends to do better than the high learning rate SGD or RMSProp optimized MetaFormer which consistently learn the same weights. The regularisation loss may be encouraging the wrong values of these weights, in general more work on designing a regularisation specific to our model would be beneficial. Alternatively the weights may simply have low impact, though that is surprising given that the MetaFormers with RMSProp ignore the support set sample from the same class using a weight $w^{(2)} = 0$ which should have a large impact on the aggregation. Given more time it would be better to repeat our experiments with learnt attention weights using BERT. Additionally we could increase their impact by learning a weight and a bias, in the form $\hat{a} = w(a + b)$, which would combine both the ability to flip to negative that a weight provides and the ability to easily increase or decrease the performance that a bias provides. Finally we should explore why the attention weights don't flip to negative values, throughout our experiments we observe that attention weights tend to maintain the sign they're initialised to. Some decrease to 0 and then stay there. We suspect that once a weight $w^{(i)}$ decreases to 0 the gradients for that weight also become 0 as it has no impact on predictions anymore. When $w^{(3)} = 0$ then the vector $\hat{V_i}$ and its gradients are independent of the vectors $w^{(3)}$ multiplies by, hence the covariance of the element k of the two vectors would be zero.

$$E|(\frac{\partial l}{\partial \hat{V}_{ik}})V_{jk}| - E|(\frac{\partial l}{\partial \hat{V_i}})|E|V_{jk}| = 0 \tag{6.1}$$

Assuming the mean of each element is zero then $E|(\frac{\partial l}{\partial \hat{V}_{ik}})V_{jk}| = 0$, and hence the dot product term in the gradient gradient of $w^{(3)}$ below would be zero, resulting in zero gradient.

$$\frac{\partial l}{\partial w^{(3)}} = \sum_i (\frac{\partial l}{\partial \hat{V_i}})^T \frac{\partial \hat{V_i}}{\partial w^{(3)}} = \sum_i (\frac{\partial l}{\partial \hat{V_i}})^T (\sum_j a_{ij} V_j) \tag{6.2}$$

This inability to learn when weights reach zero could be addressed through randomly adding a small negative value to any weight that reaches zero to escape from the zero gradient regime. In general more research is needed into this attention weight idea, even for the purpose of understanding why they have little impact.

Overall we can see the transformer-based meta-learning architecture has potential. Considering the full support set gives large performance improvements over FEAT and minor improvements over Prototypical networks in the incremental settings, even when the other contribution of learnt attention weights has little impact. The major limitation is that the encoder must be frozen, which hurts the model's ability to learn and fit the data in the standard setting. Initialisation to a prototypical network's BERT may also restrict the ability of the model to learn a better aggregation. Through further exploration of learning with an unfrozen encoder we can develop transformer-based meta-learners that significantly improve over prototypical networks.

CHAPTER 7

Conclusion

Few-shot learning is a critical technique in the current deep learning environment where ever larger models that require even larger datasets continue to be produced. Developing a model and dataset from scratch to solve a real world problem is costly, in most cases it is far preferable to use a few-shot learning technique and lose a few points of accuracy instead of gathering and labelling a massive dataset. Despite the real world applications of few-shot learning the subfield of meta-learning mostly focuses on models trained for and evaluated on simulated problems that don't accurately reflect real-world tasks. In this thesis we argued that class-incremental learning better matches said real-world tasks, and showed that meta-learners for this task have desirable qualities of few-shot learning ability and out of distribution generalisation to new classes. We proposed a method of simulating the class-incremental learning problem to train meta-learners that are robust to new classes and don't require retraining. Then we devised a transformer-based meta-learner with a learnable aggregation function that uses a weighted attention mechanism to adapt to the class-incremental setting. Our experiments show this MetaFormer to be superior to an existing transformer-based method and on-par or better than the best baseline meta-learner. Additionally Meta-learners were shown to have good generalisation ability to larger way problems, we found the standard method of training on a 10-way problem is sufficient to achieve good performance in class incremental learning. Areas for future research include creating a method of self-supervised meta-learning that doesn't use mask tokens to address the limitations of surrogate novel classes. Further investigation of the attention weight learning process is needed to address the challenging optimization problem they present. Finally we should examine the extension of these techniques to the more challenging Open World Recognition setting which better models real-world learning problems where a model may encounter queries from entirely unknown classes.

Bibliography

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Edward Guo. 2020. Knowledge distillation from internal representations. In *AAAI*.

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3988–3996.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020a. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020b. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534. Association for Computational Linguistics.

Abhijit Bendale and Terrance Boult. 2015. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1997. On the optimization of a synaptic learning rule.

Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. 2020. What is the state of neural network pruning? In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze, editors, *Proceedings of Machine Learning and Systems*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

T. E. Boult, S. Cruz, A.R. Dhamija, M. Gunther, J. Henrydoss, and W.J. Scheirer. 2019. Learning and the unknown: Surveying steps toward open world recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9801–9807.

Lake Brenden, Salakhutdinov Ruslan, Gross Jason, and Tenenbaum Joshua. 2011. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society, 33*.

Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’06, page 535–541.

Gert Cauwenberghs and Tomaso Poggio. 2001. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, pages 409–415.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1597–1607.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. 2020. A baseline for few-shot image classification. In *International Conference on Learning Representations*.

Bowen Dong, Yuan Yao, Ruobing Xie, Tianyu Gao, Xu Han, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Meta-information guided meta-learning for few-shot relation classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1594–1605.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6407–6414.

Tianyu Gao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Neural snowball for few-shot relation learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7772–7779.

Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. 2020a. Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020b. Dynamic memory induction networks for few-shot text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1087–1094.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3904–3913.

Spyros Gidaris and Nikos Komodakis. 2018. Dynamic few-shot visual learning without forgetting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4367–4375.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 33(129):1789–1819.

Aakriti Gupta, Kapil Thadani, and Neil O'Hare. 2020. Effective few-shot classification with transfer learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1061–1066.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809.

Bharath Hariharan and Ross Girshick. 2017. Low-shot visual recognition by shrinking and hallucinating features. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3037–3046.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, page 1735–1780.

T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiang Jiang, Mohammad Havaei, Gabriel Chartrand, Hassan Chouaib, Thomas Vincent, Andrew Jesson, Nicolas Chapados, and Stan Matwin. 2018. Attentive task-agnostic meta-learning for few-shot text classification. *International Conference on Learning Representations*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, and Nuno Vasconcelos. 2020. Few-shot open-set recognition using meta-learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8795–8804.

Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. 2020. Class-incremental learning: survey and performance evaluation. *arXiv preprint arXiv:2010.15277*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. 2020. A metric learning reality check. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 681–699.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

L. Rabiner and B. Juang. 1986. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Haopeng Ren, Yi Cai, Xiaofeng Chen, Guohua Wang, and Qing Li. 2020. A two-phase prototypical network model for incremental few-shot relation classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1618–1629.

Adriana Romero, Samira Ebrahimi Kahou, Polytechnique Montréal, Y. Bengio, Université De Montréal, Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *in International Conference on Learning Representations (ICLR*.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3859–3869.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. 2013. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Jurgen Schmidhuber. 1987. *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook.* Diploma thesis, Technische Universitat Munchen, Germany.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019a. Hierarchical attention prototypical networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 476–485. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019b. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332.

Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. 2011. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, NIPS'17, page 6000–6010.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, volume 29.

Duo Wang, Yu Cheng, Mo Yu, Xiaoxiao Guo, and Tao Zhang. 2019. A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning. *Neurocomputing*, 349:202–211.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3).

Hu Xu, Bing Liu, Lei Shu, and P. Yu. 2019. Open-world learning and application to product classification. In *The World Wide Web Conference*, WWW '19, page 3413–3419.

Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. 2020. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Haiyang Yu, Ningyu Zhang, Shumin Deng, Hongbin Ye, Wei Zhang, and Huajun Chen. 2020. Bridging text and knowledge with multi-prototype embedding for few-shot relational triple extraction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6399–6410.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215.