The Hong Kong Polytechnic University

Department of Computing

COMP4913 Capstone Project

Report (Final)

Application of Intra-ensemble Network on Machine Reading Comprehension (MRC)

Student ID No.:	18081072D
Programme-Stream Code:	61431-FCS
Supervisor:	Prof LI Wenjie
Co-Examiner:	Dr. LI Bo
2nd Assessor:	Dr. LIU Yan Fiona
Submission Date:	3 April 2022

Abstract

Ensemble is a widely used technique to improve the performance of neural networks, including the MRC models. However, the typical ensemble methods need to train multiple models separately, which is time-consuming and takes high cost. In this project, I propose to apply intra-ensemble in the MRC task to improve the MRC models with low additional cost. Intra-ensemble is an end-to-end ensemble, which means that the model trains several sub-networks simultaneously within one neural network. By referencing the intra-ensemble used in other research areas, I propose two possible methodologies to implement the model in the MRC task. One method is inspired by a one-shot model and generates sub-networks by adjusting the hidden layer structure. The other one usues MIMO (Multi-Input Multi-Output) configuration and generates sub-networks by modifying the input layer and output layer.

Table of Content

1 Introduction

The project aims to find a suitable intra-ensemble method to ensemble the Bert-related models for implementing the MRC task in a new way. In the first section, I mentioned the project's background, which introduces what MRC is and what is ensemble. In the second and third sections, I said the motivation and objective, explaining why this project is proposed. In the fourth part, I mentioned the current work related to this project, which provides some prerequisite knowledge. In the fifth part, I introduce two methodologies to implement the project's key point, which illustrates why this project is feasible. In the last part, I mentioned the model design and implementation and some experiment results, which show how this project can be implemented and what should be done in future work.

2 Background

2.1 Definition and classification of MRC tasks

Machine Reading Comprehension (MRC) is one of the challenging Natural Language Processing (NLP) research topics with extensive attention [1]. The MRC task is defined to **ask the machine to answer questions based on the given context** [2], which can be regarded as a sub-topic of the Question Answering (QA) task typically.

Based on different standards, there are several category schemes for MRC tasks. The most common standard is based on different answer forms, categorizing MRC into four typical tasks: cloze tests, multiple-choice, span extraction, and free answering [3]. The answer form of each task is elaborated in Table 1.

MRC tasks	cloze test	multiple choice	span extraction	free answering
answer form	a word or entity	one of the candidate's answers in the list	continuous subsequence from the context	may not be subsequence in the context

Table 1. The answer form of each typical MRC task [3].

In the recent five years, with the construction of more diverse datasets [8], researchers started to focus on the MRC tasks closer to the real-world application,

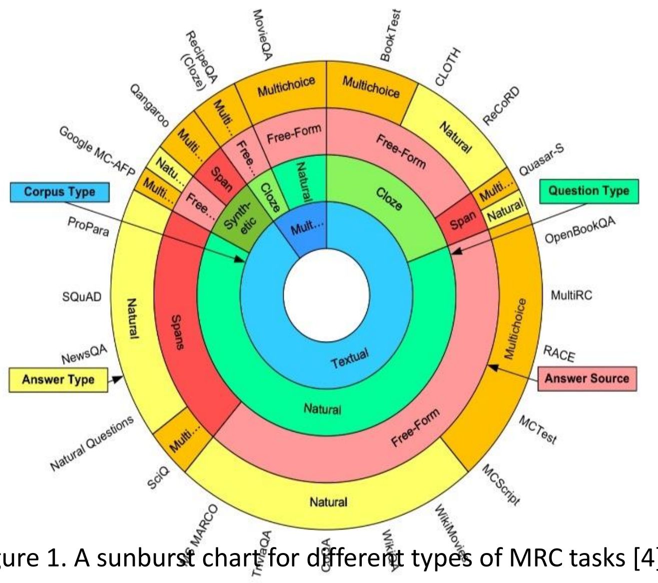


Figure 1. A sunburst chart for different types of MRC tasks [4].

such as multi-modal MRC, conversational MRC, etc. Meanwhile, a more complicated category scheme emerged. For example, Figure 1 shows a method to categorize MRC into thirteen tasks based on the corpus type, the question type, the answer source, and the answer type [4]. Each task displays one or several benchmark datasets that promote the development of the corresponding task. It is noted that, even though the categories of MRC tasks are more diverse and novel MRC tasks emerge, typical MRC tasks, like span-extraction MRC, are still one of the prime focuses of the researchers.

In this project, the focus is on improving the ensemble method rather than novel MRC tasks. Thus, I focus on the span-extraction MRC, one of the most typical MRC tasks, and **the term “MRC” refers to “span-extraction MRC” in the following part**.

2.2 Development of MRC models

With the review of plenty of MRC models, I roughly divide the development of MRC models into three stages. Specifically, the use of neural networks and the emergence of BERT [20] are respectively two cut-off points of these three stages.

2.2.1 Stage 1: before the use of neural network

In the early stage of MRC development, **traditional rule-based methods** and **classical machine-learning-based methods** are two main ways to deal with MRC tasks [5]. Traditional rule-based methods rely on the hand-crafted rules made by linguistic experts [6]. The classical machine-learning-based method relies on a set of human-defined features and then maps input features to the output by training a classical machine learning model [7].

However, these methods have several weaknesses that impede the further development of MRC tasks [5]. First, substantial human resources are required to handle the complicated hand-crafted rules and human-defined features, which consume a lot of resources and limit the use of larger datasets. Second, the rules and features are incapable of generalization, for humans produce them under specific domains. New rules and features need to be produced to handle the situation under other domains, which seems inefficient. Third, contextual information fails to be captured. The human-defined rules and features can only focus on words and sentences, and it is hard to consider each possible situation in the context. To address these problems, researchers began to think about how to use machines to further replace human work to achieve higher efficiency and better model performance.

2.2.2 Stage 2: from the use of the neural network to the emergence of BERT

With the construction of a larger dataset, e.g., CNN/Daily Mail, neural networks can apply MRC tasks [8]. With the use of the neural network, the machine can automatically learn features from the raw input data and extract contextual information. Thus, deep learning-based methods turned into the mainstream way to deal with MRC tasks from 2015.

In this stage, MRC models can be generally divided into three modules: embedding, reasoning, and answer prediction [2],[5]. **Embedding** is the first step, which converts the query and context in natural language into the form that the machine can understand (namely word representation) and then extracts features in the query sentence and context, respectively (namely feature extraction). **The reasoning** is the second step, which makes context and query interact with each other, and looks for the part of the context relevant to the query. **Answer prediction** is the last step, which predicts the start point and endpoint of the answer span, respectively, and then extracts a continuous subsequence from the context.

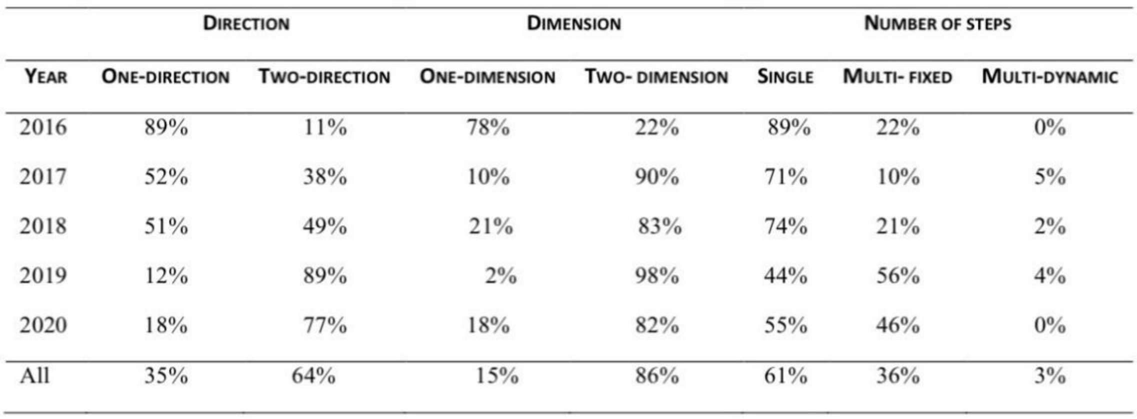
(1) Embedding

Pre-training is an important technique used in the embedding module [9], for the model with pre-training can exploit more information before the training and improve both the efficiency and the performance. As the pre-training level becomes deeper, the development of the embedding module can be divided into three stages.

As mentioned above, there are two sub-steps in the embedding modules: word representation and feature extraction [2]. **In the first stage, models do not use pre-training.** Models use a technique like Word2Vec [10] to represent the word and use single-LSTM or bi-LSTM to extract the features (e.g., Attention Reader [11], Impatient Reader [11], Match-LSTM and Pointer Network [12]). The disadvantage of these models is that word representation cannot utilize global statistical information. Thus, **in the second stage, the word representation step is executed in the pre-training process** to obtain more semantic and syntactic information. For example, GloVe [13] is one of the models for word representation with pre-training. Meanwhile, the techniques used in feature extraction are no longer confined to LSTM. Researchers try to use bi-GRU [14], a structure with fewer parameters than LSTM, to achieve faster iterations (e.g., AOA reader [15], R-Net [16]); use the combination of CNN and self-attention to capture localized information and long-distance information simultaneously (e.g., QANet [17]). However, the work representation with only static, independent linguistic contexts may lead to poor performance when the word comes to polysemy, and feature extraction is also needed in the pre-training process. Thus, **both the word representation step and the feature extraction step are executed in the pre-training process** in the third stage. The representative techniques are Elmo (use bi-LSTM to extract features) [18], GPT [19] and BERT (use variants of Transformer to extract features) [20].

(2) Reasoning

The goal of the reasoning module is to determine the parts of the context relating to the query by calculating the relevance between the context and the query. To achieve this goal, an attention mechanism is used in this module. As with the exhibition of Form 2, the development of the reasoning module can be concluded in three aspects: **direction, dimension, and several steps** [5].



	Direction		Dimension		Number of steps		
Year	One-direction	Two-direction	One-dimension	Two- dimension	Single	Multi- fixed	Multi-dynamic
2016	89%	11%	78%	22%	89%	22%	0%
2017	52%	38%	10%	90%	71%	10%	5%
2018	51%	49%	21%	83%	74%	21%	2%
2019	12%	89%	2%	98%	44%	56%	4%
2020	18%	77%	18%	82%	55%	46%	0%
All	35%	64%	15%	86%	61%	36%	3%

Table 2. Statistics of the development of different reasoning aspects [5].

- Direction

One-direction reasoning means that only context-to-query attention is considered (e.g., Attentive Reader [11], Impatient Reader [11]). However, such reasoning overlooks the information from query-aware context representation. Thus, **two-direction** reasoning is used more widely later, which means that both context-to-query attention and query-to-context attention are considered (e.g., Match-LSTM and Pointer Network [12], BiDAF [21], and so on).

- Dimension

One-dimension reasoning means that the query is considered as a whole and represented in one vector (e.g., Attentive Reader [11]). However, one-dimension reasoning loses the information about the query, especially for the long-length query. Thus, **two-dimension** reasoning becomes a mainstream method later, which embeds the query into a two-dimensional vector, taking each word as an individual representation (e.g., Inpatient Reader [11], Match-LSTM and Pointer Network [12], BiDAF [21] and so on).

- Number of steps

There are three categories of several steps: single steps, multi-fixed steps, and multi-dynamic steps. **Single-step** reasoning means that there is only one layer for reasoning (e.g., Attentive Reader [11], BiDAF [21], AOA Reader [15]), which is easier and time-saving. **Multi-dynamic steps** reasoning means that there are multi-layers for reasoning, and the number of layers is dynamic, which can be modified by the neural network itself according to the actual situation (e.g., Inpatient Reader [11], R-NET [16]). **Multi-fixed steps** reasoning means that there are multi-layers for reasoning, and the number of layers is fixed (e.g., DCN [22], QANet [17], BERT [20]), which can have better reasoning performance than single-step reasoning and is more manageable than multi-dynamic steps reasoning. According to the statistics, the prime methods are single step and multi-fixed steps, and the use of multi-fixed steps is wider and wider.

(3) Answer prediction

There is not too much variation in this module. The mainstream method directly adds a non-linear layer to combine the context and query representation. The non-linear predicts the probability to be the start and end for each word in the context, respectively, and then selects the final answer span (e.g., Attentive Reader [11], Impatient Reader [11], AOA reader [15], BERT [20]). In addition, some models make some alterations in this module. For example, DCN [22] uses dynamic pointing, a combination of highway maxout network and LSTM, to make the start index changeable even after the determination of the end index.

2.2.3 Stage 3: after the emergence of BERT

BERT [20] achieved state-of-the-art performance on SQUAD [33], the dataset of span-extraction MRC, and after the emergence of BERT, the two-stage learning architecture (pre-train + fine-tuning) is widely used in various models [23]. However, how to improve the performance of BERT further becomes a key problem for researchers [24]. Their initial idea is to expand the size of BERT. However, this idea has several problems: memory limitations, large communication overhead, and lower performance with too many hidden nodes exceeding a certain number.

Therefore, researchers began to focus on modifying the architecture of BERT, aiming to achieve improvement in different aspects. For example, Roberta [25] uses longer training time and more data to achieve better performance; DistilBERT [26] achieves a little degradation of performance from BERT but useless training time and data; Albert [24] is a lite BERT, even though it makes little improvement on the training time, it can use fewer parameters to gets significantly better results.

Besides modifying the architecture of BERT, the introduction of the verifier mechanism is also an important improvement in MRC. To validate the answerability of the query by verifying the legitimacy of the predicted answer, researchers propose a read and verify system [27], which provides a new research direction for MRC. After that, researchers propose a retro-reader, a new verification mechanism, which uses Albert as a baseline model and achieves SOTA performance at that time [28].

2.3 Definition and classification of inter-ensemble learning

Ensemble learning is the method to **combine several individual models to achieve better generalization performance** [29] than a single model. Some ensemble methods can kindly reduce the bias of the models, while others can reduce their variance. Normally, a complicated model tends to have low bias but a high variance and vice versa. Thus, we can apply different ensemble methods according to the nature of different models. Essentially, the ensemble is used to achieve a balance status of bias-variance trade-off.

2.3.1 Definition of inter-ensemble learning

For the normal ensemble methods, each model is relatively independent in the training stage and interacts with each other after individual training. This is the most used ensemble method, which we call "inter-ensemble." When applying inter-ensemble learning on NLP, there are three typical strategies: **bagging, stacking, and boosting**.

2.3.2 Bagging

Averaging technique means that multiple sub-models learn in parallel and then combine following some deterministic averaging process. We can apply diverse algorithms to different sub-models using the same dataset, or we can apply the same algorithm for each sub-model on different instances. Bagging is mainly used to reduce the variance of base models.

2.3.3 Stacking

Stacking [30] is like bags, but they have one main difference. Compared with the equal contribution of each sub-model in averaging, we weight the contribution of the predicting result of each sub-model and train a new model to calculate the weights for the predicting result of each sub-model to get the result. Stacking is mainly used to reduce the bias of base models.

2.3.4 Boosting

Boosting and stacking are techniques where sub-models can execute simultaneously while boosting is a technique where sub-models need to execute sequentially [29]. After each round of training, we test the errors with the use of an existing model and then improve the weights of samples that result in error predictions, which can promote the following training to pay more attention to the errors. After several rounds, we combine the predictions in each round and get the results. Boosting can effectively reduce the model's bias, but it is not very efficient because the sub-models cannot run parallel.

3 Motivation

MRC is a task to test the degree to which a machine understands the text [1], which fully matches the main goal of NLP, which is to make computers achieve human-like comprehension of the natural language. Therefore, MRC is a field with great research value so far.

However, the performance of a single model is hard to achieve a very effective breakthrough after reaching a certain level, especially for MRC, which is a relatively mature research field. Thus, researchers started to think about whether the integration performance of multiple models can surpass the single model. Then, ensemble learning emerged and began to apply to the MRC tasks.

However, typical inter-ensemble methods have several drawbacks that impede their application. One of the main problems is that an inter-ensemble needs to train multiple models, which is time-consuming and costs more [31].

We can find that when we do the inter-ensemble, some of the training models have partially similar architecture and part of the parameters are similar, which leads to some redundant work. If we can train several models together, part of parameters from different models can share with each other, which is timesaving and resource-saving and can reach similar performance theoretically.

By referencing several papers relating to intra-ensemble in other research areas, I reckon that intra-ensemble is realizable in MRC tasks. In the paper "Intra-Ensemble in Neural Networks" [32], the author proposes an intra-ensemble method. Differentiating from the typical inter-ensemble method, the intra-ensemble model trains several sub-networks simultaneously within one neural network. Even though the author does not provide the official codes and I cannot see the implementation in detail, in my point of view, the intra-ensemble method has the potential to kindly address part of the problem mentioned above, for the parameters can be shared between diverse sub-networks in the training time. What's more, the authors claim that intra-ensemble achieves competitive results compared with other methods on classification tasks, but they do not show the application on MRC tasks yet. Recalling the answer prediction method for span-extraction MRC, we can consider it a classification task's derivation. Thus, I take span-extraction MRC as the target to test the effect of the intra-ensemble method.

4 Objective

The objective of this project is in two layers. The **basic objective** is to design and apply the intra-ensemble method to span extraction MRC task concerning the paper "Intra-Ensemble in Neural Networks" [32]. The **further objective** is to find the drawbacks of existing intra-ensemble networks and modify the architecture of intra-neural networks to achieve better performance.

To achieve the **basic objective**, I need first to determine the dataset and base models used to integrate. For implementing base models, I can refer to the code on open-source platforms, like HuggingFace. The key point is how to generate sub-networks and combine them with various strategies.

To achieve the **further objective**, I need to compare the drawbacks of existing intra-ensemble networks applied to MRC tasks. First, I need to compare the performance of intra-ensemble networks with inter-ensemble networks with the same base-models selection. Then, I also need to analyze why it makes a difference in performance when applying an intra-ensemble network on different base-model selections. With these observations and analysis, I hope to find some feasible methods to improve the architecture of intra-ensemble networks.

5 Related Work

5.1 Database: SQuAD 2.0

This project uses SQuAD 2.0 to train and test the models. Stanford Question Answer Dataset (SQuAD) [33] is a benchmark for span extraction MRC task, consisting of a set of Wikipedia articles, questions proposed according to these articles, and answers for each question extracted from a span from original articles. Based on SQuAD 1.0, unanswerable questions are added in SQuAD 2.0, closer to real-world application.

5.2 Bert for MRC task

Bert and Bert-related models have a highly similar structure to implement MRC tasks. Here I take BERT as an example.

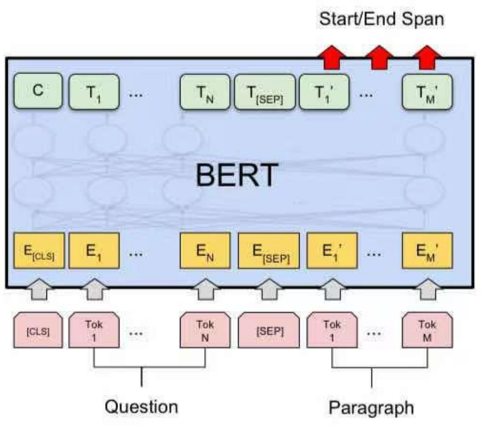


Figure 2. Structure of Bert for MRC task [32].

As shown in Figure 2, we take the concatenation of question and paragraph as the input of the model. Bert manipulates the embedding and reasoning module. For the answer prediction, we add one more layer above the original Bert structure. This linear layer is used to learn the probability that one position in the paragraph can be the start/end of the answer span. Then, we take the position with the highest start probability at the start of the span and take the position with the highest end probability at the end of the span.

6 Project Methodologies

The main idea of intra-ensemble is to train several sub-networks within one neural network, namely, an end-to-end ensemble. To get better ensemble performance within one network, the sub-networks should share part of parameters with each other and be partially distinctive from each other. Therefore, the difficulty of intra-ensemble implementation lies in the method to generate sub-networks, and our methodology is mainly related to this key point.

With reference to intra-ensemble applications to other machine learning problems, I want to find a methodology that is also applicable to MRC tasks.

6.1 Intra-ensemble inspired one-shot model

6.1.1 One-shot model [34]

The one-shot model proposes a way to generate several sub-networks when training one neural network. This method can generate distinctive sub-networks because, with the use of the one-shot model, the neural network has multiple optional operations at each position during the training process. Therefore, different sub-networks are generated by taking different operations at each position.

6.1.2 One-shot model for intra-ensemble

In the paper "Intra-Ensemble in Neural Networks" [32], the author was inspired by the one-shot model and proposed a new ensemble method for a classification task that is different from the normal inter-ensemble method. In this section, I summarize this paper, pointing out the main idea and main approaches mentioned in the paper, which inspired this project.

6.1.2.1 Overview of the referenced paper

The paper proposes an intra-ensemble network, an **end-to-end** ensemble model, which trains several sub-networks simultaneously within one neural network. The structure of the network is shown in Figure 3. There are two main **motivations** to

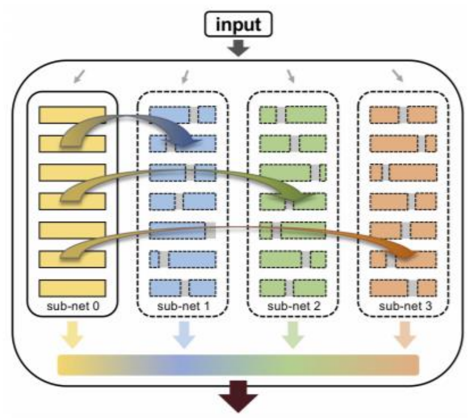


Figure 3. Structure of intra-ensemble network [32].

propose the intra-ensemble network. The first is that stand-alone neural networks always suffer marginal effects when stacking more layers. And the second one is that training several independent deep neural networks for ensembles costs multiple resources. Both motivations are consistent with our goals.

The **main idea** of the intra-ensemble network is to use **stochastic channel recombination** and **weight sharing**. For stochastic channel recombination, high-performance and diverse sub-networks should be generated from the original base model. For the weight sharing, parameters can share among different sub-networks to reduce the cost of training.

Inspired by the one-shot model, the method to generate sub-networks in the mentioned paper is that, at different network positions, the model can choose the cut of the channel or shuffle the channel based on the original neural network architecture. In other words, we can generate sub-networks by doing channel recombination.

To be more specific, the author elaborates on two-dimension channel recombination, **width recombination, and depth recombination**.

6.1.2.2 Sub-network generation: width recombination

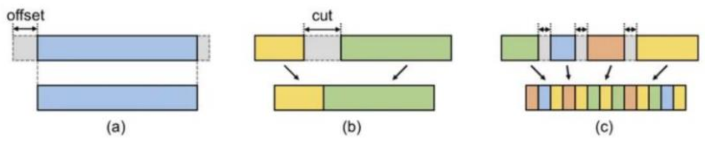


Figure 4. Methods for width recombination. (a) Random offset. (b) Random cut. (c) Shuffle channel. [32]

First, the author elaborates on three methods for **width** recombination, which are shown in Figure 4. **Random offset** (Figure 4 (a)) means that if we have a layer with c channels in total, and we want to cut out p percent channels with offset t, then we remain the channels with index [t, t+1, …, t + pc). **Random cut** (Figure 4 (b)) means that if we have a layer with c channels in total, and we want to cut out p percent channels with cut index t, then we cut the channels with index [t, t+1, …, t + pc) and keep the remaining. **Shuffle channel** (Figure 4 (c)) means that we cut the sub-net layer into several sub-lists of channels and shuffle these sub-lists. Then, different sub-network can use the channels in a different order.

For the width recombination method, the shuffle channel is our first choice. In this task, we need to predict the probability that one position in the paragraph can be the start/end of the answer span. All positions in the paragraph have the potential to have a determinative influence on the result. Thus, we do not want to cut any fragment of the channel in width. Based on this consideration, random offset and random cut are unsuitable for this project.

6.1.2.3 Sub-network generation: depth recombination

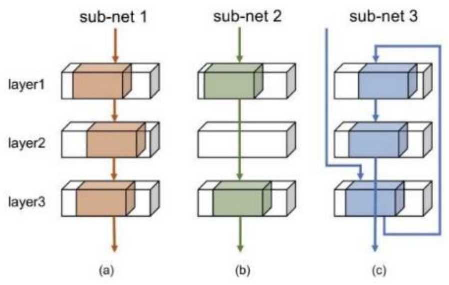


Figure 5. Methods for depth recombination. (a) Original. (b) Random skip. (c) Shuffle layer. [32]

What's more, the recombination can perform in**-depth** dimensions (Figure 5). The first method is called **random skip** (Figure 5 (b)), in which each subnet can randomly skip one or more layers to generate sub-networks with different depths. Another method is called **shuffle layer** (Figure 5 (c)), in which we shuffle the layers in different sub-networks to make them in various depth order.

Random skip and shuffle layers are also reasonable and can be considered for the depth recombination method.

6.1.2.4 Sub-networks combination

After generating several diverse sub-networks, we use averaging or stacking to combine the outputs of sub-networks and get the result of the whole model. The process of sub-network combination in intra-ensemble is like the process of inter-ensemble. There are three alternative strategies: bagging, boosting, and stacking. We can implement the model with these three strategies respectively, compare the performance with each other, and compare its performance with the pure inter-ensemble model with corresponding strategies.

6.1.2.5 Base model selection

Based on stochastic channel recombination and weight sharing considerations, the base model should have good performance and a relatively simple structure. For Albert, even though this model achieves the best performance among all tested base models, cross-layer parameter sharing is already applied in this model. Thus, the ensemble structure will be too complicated if weight sharing among sub-networks is also considered. Therefore, Bert is the most suitable base model for this project.

6.2 Intra-ensemble with MIMO Configuration

6.2.1 MIMO configuration

Marton [35] proposed a Multi-input Multi-output (MIMO) configuration [35] in the article "Independent Training Subnetworks for Robust Prediction." MIMO means that the network takes M=n> one input and gives M outputs instead of one input and one output.

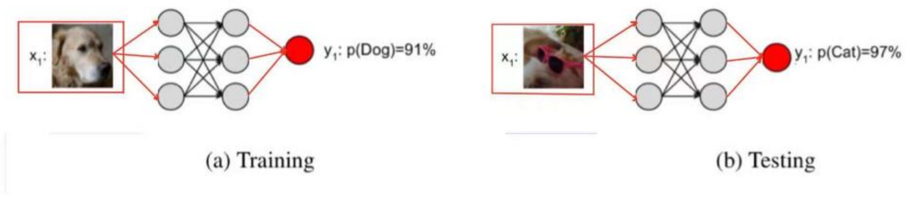


Figure 6. Original model structure with one input and one output.

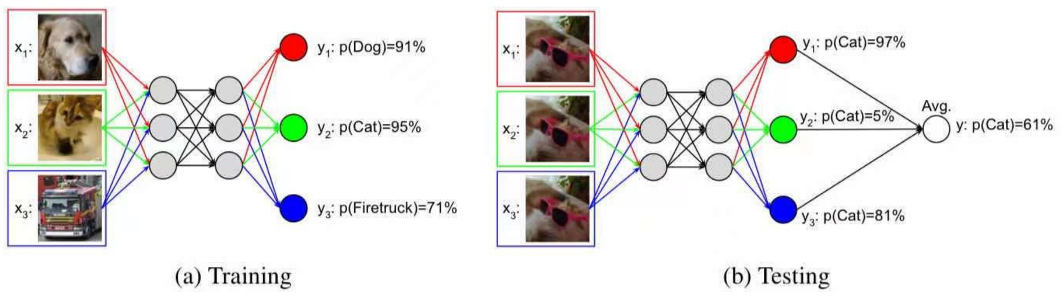


Figure 7. Modified model structure with M=3 input and M output. [35]

MIMO configuration is normally used in classification problems, and here, we take an image recognition task as an example. Figure 6 shows the original model with one input and one output, while Figure 7 illustrates the modified model with MIMO configurations, which means the networks have M=3 inputs and M=3 outputs.

We can find that hidden layers of the network remain the same after applying the MIMO configuration. To add MIMO configuration into a neural network, we only need to modify the input and output layers. When training the model, we take M data points as the inputs for the input layer instead of one single point. For the output layer, we take M predictions corresponding to each input. When testing the model, we use the same data for all input points, namely, input the testing data for M times in each iteration. We make M predictions for the same testing data for the output layer and calculate the final result by taking the average or using other combination methods.

6.2.2 MIMO configuration for intra-ensemble

The motivation of MIMO configuration is that the neural network is not fully used when making the one-input one-output prediction. The lottery ticket hypothesis [34] illustrates that shows that the performance of a neural network can not be heavily affected even if cutting 70% to 80% of connections. Therefore, in theory, one neural network has the capacity to fit more independent subnetworks (around 3 to 4) at the same time.

Since M input datapoints are independent and the model calculates M predictions for each input data separately, we can consider them part of the neural network used to calculate each prediction as one separate subnetwork. Therefore, we can consider M predictions as the results of the M subnetwork generated automatically from the original network. It is noticed that Marton [35] shows that M subnetworks use disjoint parts of the network, and they are relatively independent. It is known that the diversity of subnetworks is a prerequisite of the high-performance model ensemble. That is why we can implement intra-ensemble in a single forward pass by MIMO configuration without modifying the hidden layers of the network.

7 Model Design and Implementation

7.1 Implementation of base models

Before ensemble the models, we first run the base models and take them as benchmarks, which can be used to measure the ensemble's performance. What's more, the analytics of base models can help us select the set of base models for the ensemble better.

7.1.1 Source of the code

The code of this project is modified based on the transformer provided by the Huggingface, which provides thousands of pretrained models (e.g., Bert, Albert, and so on) to perform tasks (e.g., span extraction MRC) on the different datasets (e.g., SQuAD).

In this project, we downloaded the official code of the transformer from the GitHub of Huggingface and made some modifications to complete our tasks. In the first step, based on the pre-trained models provided by Huggingface, we fine-tune and retrain the base model using SQuAD. The detailed structure is mentioned in the last section.

7.1.2 Results of base models

model_ type	**model_name _or_path**	**epoch**	**training time**	**HasAns _exact**	**HasAns _f1**	**NoAns_ exact**	**NoAns_ f1**	**exact**	**f1**
Bert	Bert-base-uncased	1	1:32:23	68.7416	74.8450	72.7839	72.7839	70.7656	73.8130
		2	3:10:56	71.7949	78.0118	73.5408	73.5408	72.6691	75.7731
		4	6:10:00	72.4359	79.7706	72.5652	72.5652	72.5006	76.1628
Albert	Albert-base-v2	1	1:29:47	73.7011	80.0952	83.2296	83.2296	78.4722	81.6646
Roberta	Roberta-base	1	3:23:31	72.0310	78.7510	77.4769	77.4769	74.7579	78.1130

distilBert	distilBert-base-uncased	1	47:00	61.2854	67.9924	65.0799	65.0799	63.1854	66.5341
xlnet	xlnet-base-cased	1	2:54:56	0.21929	1.24681	67.4348	67.4348	33.8752	34.3882

Table 3. Running result of base models

7.1.3 Result analytics

According to the observation of the results, Albert is a base model with moderate training time and the best f1 score within the provided base models. If we want to achieve better performance using an ensemble, Albert can be chosen as a base model. At the same time, Bert (with two epochs) is a base model with much simpler architecture and relatively good performance. If we want to experiment new method, starting with Bert as a base model is also a good choice.

More specific, according to the observation of models with the same type but a different number of training epochs, we find that even though a large number of epochs can always bring higher accuracy, the cost in time is huge, and the improvement of the performance is limited if the number of epochs is too large. Thus, besides the accuracy, the epochs-accuracy trade-off should also be considered.

7.1.4 Further implementation

To further analytics, I will consider the nature of different base models to help us make better decisions when selecting the set of base models used in different ensemble models. The following three aspects are under consideration:

1. bias and variance in each base model
2. performance for a different types of questions in each base model
3. different performance for has-answer questions and no-answer questions in each base models

7.2 Design and Implementation of inter-ensemble methods

7.2.1 Implementation

For each ensemble model, we can add an ensemble class in the downloaded transformer package and make a little adjustment in the original fine-tune process. In this way, we can make different ensemble combinations of base models using different methods and conveniently record the results.

7.2.2 Strategy 1: bagging

I first considered bagging, for it is the most typical and basic one. However, the integration process of this method is too greedy and not very effective enough. Thus, more sophisticated ensemble methods should be considered.

7.2.3 Strategy 2: boosting

Then, I consider boosting. Even though boosting ensembles is time-consuming for sub-models that cannot run parallel, it can kindly reduce the model's bias. However, I will not implement it in this project, for the base models in this task are relatively sophisticated and always have low bias and high variance. In this project, I pay more attention to improving efficiency and variance.

7.2.4 Strategy 3: stacking

After that, I consider stacking. It is implementable and widely used for MRC tasks. Stacking is generally applied to heterogenous models and can kindly reduce the variance of the base models. Thus, the staking model is used to represent inter-ensemble and will compare its performance with intra-ensemble models.

7.2.5 Strategy 4: read + verify system [36]

I also tried the read plus verify system, which can be considered a derivation of inter-ensemble. Retrospective reader [37] is a classical architecture using a read plus verify system. Instead of making more than one prediction and combining them, retrospective reader architecture trains two models one is for prediction, and another is for verification. Firstly, it implements sketchy Reader, which briefly extracts the interactions of passage and question and makes a sketchy judgment. Secondly, it implements an intensive Reader, which is used to verify the answer and generate the final predictions.

model_name	epoch	training time	HasAns _exact	HasAns _f1	NoAns_ exact	NoAns_ f1	exact	f1
sketchy Reader	2	3:33:08	acc = 0.80645					
intensive Reader	1	1:48:15	71.7949	73.9372	82.2372	82.2372	78.0932	81.3998
rear verification	NA	NA	72.7227	79.2707	84.3230	84.3230	78.5311	81.8005

Table 4. Running result of Retrospective Reader

Table 4 shows my running result of retrospective reader architecture after debugging the official code. We can find that the read plus verify the system is a good way to make the prediction more robust, but its performance is limited.

7.3 Design and implementation of intra-ensemble methods

Even though stacking ensemble already makes some improvements on MRC models, some drawbacks cannot be addressed. One of the main drawbacks is that each sub-network runs independently, even though they have similar structures and parameters. Thus, I think about whether some methods can take all sub-networks as a whole and build effective and efficient interaction among them. Theatrically, this method can be very practical and resultful. Thus, I decided to implement the intra-ensemble model, an end-to-end ensemble model, and test its performance. For Bert has the most simple and classical structure, I take Bert as the base model for implementing intra-ensemble methods.

7.3.1 Comparison of different methodologies

In the 5th part, I illustrate two methodologies to implement intra-ensemble. For the intra-ensemble inspired by the one-shot model, we should modify hidden layers and recombine the channels in both width and depth to construct the subnetworks. It is tough to implement since hidden layers of Bert are highly encapsulated and hard to fine-tune. This method is hard to transfer to other base models, for each Bert-related model has different hidden layer structures. We need to take a lot of energy to modify the model structure for each base model, and it is time-consuming and inefficient. In other words, the first method has low reusability.

For the intra-ensemble with MIMO configuration, the only thing we need to adjust for the model is its input layer and output layer, which is more feasible and can be easily reused when we take more sophisticated models as the base model. Therefore, I implement the intra-ensemble model with MIMO based on Bert in this project.

7.3.2 Implementation of intra-ensemble with MIMO configuration

The result of the MRC model with the use of Bert is considered the benchmark of the intra-ensemble model. In this project, I use the class "BertForQuestionAnswering" from the HuggingFace and take that result as the benchmark. What's more, the ensemble model is implemented based on modifying the class "BertForQuestionAnswering."

To implement the intra-ensemble with MIMO configuration, we first need to determine the value of M, namely the number of inputs and outputs. According to the lottery ticket hypothesis [34] we mentioned before, 70% to 80% of neural network connections are redundant in training, and 3 to 4 subnetworks can be trained simultaneously in one network. Therefore, I set M=3 in the model. After determining the value M, I need to make modifications based on the original MRC model in two aspects: the input and output layers.

7.3.2.1 Input layer

1. Feature reconstruction

Normally, I should modify the input layer to fit the situation of multiple inputs. However, I did not find a suitable way to build connections between the multiple inputs and the modified input layer. Therefore, I use a tricky way to reconstruct the input feature to simulate the multiple-input. The feasibility of this method needs further verification.

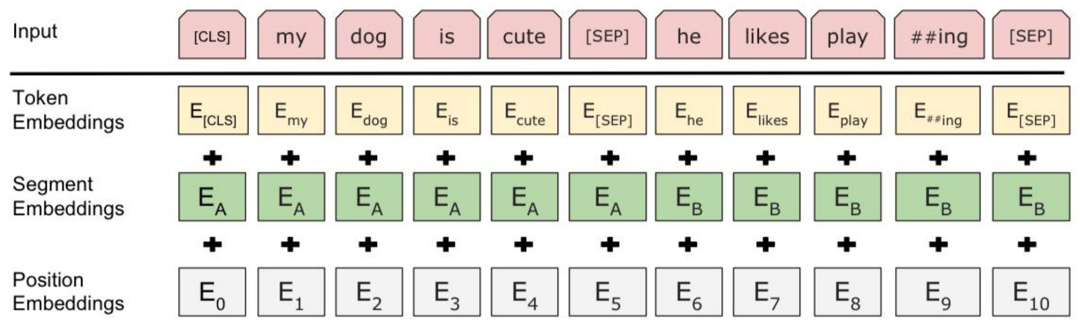


Figure 8. Input format after Bert embedding. [20]

Figure 8 illustrates the original feature formats after embedding the input. In the ensemble model, I combine three inputs into one new input. Then input should be [CLS] + <paragraph 1> + [SEP] + <question 1> + [SEP] + <paragraph 2> + [SEP]+ <question 2> + [SEP]+ <paragraph 3> + [SEP]+ <question 3> + [SEP]. With the new input, the embedding methods should be adjusted accordingly to fit Bert's input layer format. With the use of this tricky method, the convenience increases while the independence of multiple inputs declines, for Bert uses self-attention to extract the features of the query and context, and it cannot encode the sequentially of the input sequence. Therefore, the performance of embedding may be affected.

7.3.2.2 Output layer

1. Number of nodes for the output layer

the output layer is added above the Bert model to compute the start and end positions after the embedding and reasoning. For the model to have three predictions, the model should have three start and end position pairs, namely six output nodes.

2. Loss computation

I use cross-entropy to calculate the loss of each position respectively and take the average of these six values as the final loss.

2. Output content for testing

We will have three predictions based on the same testing data when we do the testing. In the model, I simply calculate the average score of all predictions for each position that can be the start of the answer and choose the position with the highest score as the final start position. The calculation of the end position is in the same way.

8 Conclusion

The literature review about the Machine Reading Comprehension includes its definition, primary models, key points, main innovations for each node of development, and the future development trend. With the deep thinking of the problem development, I propose that the application of intra-ensemble can be the way to make future improvements based on the existing Stage-of-the-Art models. By referencing the intra-ensemble methods used in other research areas, I build the methodologies applicable for MRC tasks and choose one of the methodologies to implement the model. Even though the model is runnable, the performance is under expectation. I will further optimize the methodology and debug the code to reach expectations.

References

[1] Changchang Zeng and Shaobo Li and Qin Li and Jie Hu and Jianjun Hu, "A Survey on Machine Reading Comprehension: Tasks, Evaluation Metrics, and Benchmark Datasets", arXiv: 2006.11880, [cs.CL], 2020.

[2] Li, Kaixuan and Xian, Xiujuan and Wang, Jiafu and Yu, Niannian, "First-principle study on honeycomb fluorated-InTe monolayer with large Rashba spin splitting and direct bandgap", Applied Surface Science, 471, Elsevier BV, pp. 8–22, March 2019.

[3] Dani Chen, “Neural Reading Comprehension and Beyond”, PhD thesis, Stanford University, 2018.

[4] C. Zeng, S. Li, Q. Li, J. Hu, and J. Hu, “A Survey on Machine Reading Comprehension—Tasks, Evaluation Metrics and Benchmark Datasets,” Applied Sciences, vol. 10, no. 21, pp. 7640, October 2020.

[5] Razieh Baradaran and Razieh Ghiasi and Hossein Amirkhani, "A Survey on Machine Reading Comprehension Systems", arXiv: 2001.01582, [cs.CL], 2020.

[6] Riloff, Ellen, and Michael Thelen, "A rule-based question answering system for reading comprehension tests." In Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems, pp. 13-19, Association for Computational Linguistics, 2000.

[7] Ng, Teo, Kwan 2000: Ng, Hwee Ton , Leong Hwee Teo, and Jennifer Lai Pheng Kwan. 2000, "A Machine Learning Approach to Answering Questions for Reading Comprehension Tests." In Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, pp. 124-32, 2000.

[8] Xin Zhang and An Yang and Sujian Li and Yizhong Wang, "Machine Reading Comprehension: a Literature Review", arXiv: 1907.01686, [cs.CL], 2019.

[9] Zhengxiao Du and Yujie Qian and Xiao Liu and Ming Ding and Jiezhong Qiu and Zhilin Yang and Jie Tang, "All NLP Tasks Are Generation Tasks: A General Pretraining Framework", arXiv: 2103.10360, [cs.CL], 2021.

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

[11] Karl Moritz Hermann, Toma's Ko ˇ cisk ˇ y, Edward Grefenstette, Lasse Espeholt, Will Kay, ` Mustafa Suleyman, and Phil Blunsom, “Teaching machines to read and comprehend”, In Proceedings of the 28th International Conference on Neural Information Processing Systems, Vol. 1, pp. 1693–1701. MIT Press, 2015.

[12] Shuohang Wang and Jing Jiang, “Machine comprehension using match-lstm and answer pointer”, arXiv preprint arXiv:1608.07905, 2016.

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning, “Glove: Global vectors for word representation”, In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, 2014.

[14] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734, Association for Computational Linguistics, 2014.

[15] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu, “Attention-over-attention neural networks for reading comprehension”, In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vol. 1: Long Papers, pp. 593–602, 2017.

[16] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou, “Gated self-matching networks for reading comprehension and question answering”, In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vol. 1: Long Papers, pp. 189–198, 2017.

[17] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le, “Qanet: Combining local convolution with global self-attention for reading comprehension”, arXiv preprint arXiv:1804.09541, 2018.

[18] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer, “Deep contextualized word representations”, arXiv preprint arXiv:1802.05365, 2018.

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, “Improving language understanding by generative pre-training”, Technical report, OpenAI, 2018.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding”, In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1: Long and Short Papers, pp. 4171–4186, 2019.

[21] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi, “Bidirectional attention flow for machine comprehension”, arXiv preprint arXiv:1611.01603, 2016.

[22] Caiming Xiong, Victor Zhong, and Richard Socher, “Dynamic coattention networks for question answering”, arXiv preprint arXiv:1611.01604, 2016.

[23] Kaito Song and Hao Sun and Xu Tan and Tao Qin and Jianfeng Lu and Hongzhi Liu and Tie-Yan Liu, "LightPAFF: A Two-Stage Distillation Framework for Pre-training and Fine-tuning", arXiv: 2004.12817, [cs.CL], 2020.

[24] Zhenzhong Lan and Mingda Chen and Sebastian Goodman and Kevin Gimpel and Piyush Sharma and Radu Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", arXiv: 1909.11942, [cs.CL], 2020.

[25] Yinhan Liu and Myle Ott and Naman Goyal and Jingfei Du and Mandar Joshi and Danqi Chen and Omer Levy and Mike Lewis and Luke Zettlemoyer and Veselin Stoyanov, "Roberta: A Robustly Optimized BERT Pretraining Approach", arXiv: 11907.11692, [cs.CL], 2019.

[26] Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", arXiv: 1910.01108, [cs.CL], 2020.

[27] Minghao Hu and Furu Wei and Yuxing Peng and Zhen Huang and Nan Yang and Dongsheng Li, "Read + Verify: Machine Reading Comprehension with Unanswerable Questions", arXiv: 1808.05759, [cs.CL], 2018.

[28] Zhuosheng Zhang and Junjie Yang and Hai Zhao, "Retrospective Reader for Machine Reading Comprehension", arXiv: 2001.09694, [cs.CL], 2020.

[29] M. A. Ganaie and Minghui Hu and M. Tanveer and P. N. Suganthan, "Ensemble deep learning: A review", arXiv: 2104.02395, [cs.CL], 2021.

[30] David H Wolpert, “Stacked generalization”, Neural networks, vol. 5 no. 2, pp. 241–259, 1992.

[31] Anna Aniol and Marcin Pietron, "Ensemble approach for natural language question-answering problem", arXiv: 1908.09720, [cs.CL], 2019.

[32] Yuan Gao and Zixiang Cai and Lei Yu, "Intra-Ensemble in Neural Networks", arXiv: 1904.04466, [cs.CL], 2020.

[33] Pranav Rajpurkar and Jian Zhang and Konstantin Lopyrev and Percy Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", arXiv: 1606.05250, [cs.CL], 2016.

[34] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le, “Understanding and simplifying one-shot architecture search,” in ICML, pp. 549–558, 2018.

[35] Havasi, Marton and Jenatton, Rodolphe and Fort, Stanislav and Liu, Jeremiah Zhe and Snoek, Jasper and Lakshmi Narayanan, Balaji and Dai, Andrew M. and Tran, Dustin, “Training independent subnetworks for robust prediction”, in ICLR, 2020.

[36] Hu, Minghao and Wei, Furu and Peng, Yuxing and Huang, Zhen and Yang, Nan and Li, Dongsheng, "Read + Verify: Machine Reading Comprehension with Unanswerable Questions", arXiv: 1808.05759, [cs.CL], 2018.

[37] Zhang, Zhuosheng and Yang, Junjie and Zhao, Hai, "Retrospective Reader for Machine Reading Comprehension", arXiv: 2001.09694, [cs.CL], 2020.