Emma Boulharts

Master Nanotech
2022/2023

IBM Research
San Jose, California

Pruning ALBERT transformer for Analog-AI

from March to August 2023

Supervisor Stefano Ambrogio
Research Staff Member, IBM Almaden
stefano.ambrogio@ibm.com

Tutor Liliana Prejbeanu
Professor, Grenoble INP - UGA
liliana.buda@cea.fr

Confidential : yes

École nationale supérieure de physique, électronique, matériaux
3 Parvis Louis Néel - 38000 Grenoble, France

I would like to express my sincere gratitude to my supervisor, Stefano Ambrogio, for his support and mentorship during these six months. I am proud of the work we have accomplished together. I also extend my thanks to Sidney Tsai for her leadership and trust in my abilities. Her management style and vision are a great source of inspiration. I would also like to thank Pritish Narayanan and Geoffrey Burr for their technical assistance and advices. I am immensely grateful to the entire Analog-AI team for their warm welcoming.

Contents

List of Figures

List of Tables

Glossary

Analog-AI : Analog In-Memory Computing

LLM : Large Language Model

NLP : Natural Language Processing

DNN : Deep Neural Network

MAC : Multiply-Accumulate operation

PCM : Phase Change Memory

ALBERT : transformer encoder-based model

BERT : transformer encoder-based model with shared weights

GLUE : General Language Understanding Evaluation benchmark to evaluate natural language processing tasks

Pruning : Removal of less important or redundant components (such as neurons or connections) from a neural network

Abstracts

English

Analog In-Memory Computing enables latency and energy consumption reduction on Deep Neural Network inference and training. The Analog-AI group developed a chip, ARES, capable of computing the Multiply-Accumulate (MAC) operation using Phase Change Memory devices. To demonstrate the performance of the chip, the ALBERT model, a more compact version of the widely known BERT transformer, is currently under experimental study. In this report, a general in-depth analysis of the contributions to the MAC is provided, revealing that some activation/weight pairs assume larger importance, while others can be safely pruned with very limited impact on accuracy. A new row-wise pruning strategy is proposed, followed by fine-tuning, which leads to reduced model size with equivalent accuracy. The proposed algorithm is then applied on the GLUE task using the ALBERT architecture, demonstrating simulated software-equivalent performance even with consistent weight pruning, potentially enabling several improvements such as reduction of required hardware tiles, superior power performance and simpler model on-chip deployment.

French

L’ Analog In-Memory Computing permet de réduire la latence et la consommation d’énergie lors de l’inférence et de l’entraînement des Réseaux de Neurones Profonds (Deep Neural Networks). Le groupe Analog-AI a développé une puce, ARES, capable d’effectuer l’opération Multiply-Accumulate (MAC) à l’aide de mémoire à changement de phase (Phase Change Memory). Pour mettre en avant les performances de la puce, le modèle ALBERT, une version plus compacte du célèbre transformateur BERT, fait actuellement l’objet d’une étude expérimentale.

Dans ce rapport, une analyse approfondie des contributions au MAC est fournie, révélant que certaines paires activation/poids ont une importance élevée, tandis que d’autres peuvent être supprimées avec un impact très limité sur la précision. Une nouvelle stratégie d’élagage par lignes (row-wise pruning) est proposée, suivie d’un fine-tuning, ce qui permet de réduire la taille du modèle tout en maintenant une précision équivalente.

L’algorithme proposé est ensuite appliqué au dataset GLUE en utilisant l’architecture ALBERT, démontrant des performances simulées équivalentes à celles du logiciel, avec un pruning permettant plusieurs améliorations telles que la réduction du nombre de tiles nécessaires pour l’implementation, de meilleures performances énergétiques et un déploiement plus simple du modèle sur la puce.

Italian

L’ Analog In-Memory Computing consente di ridurre la latenza e il consumo energetico nell’elaborazione e nell’addestramento di reti neurali profonde (Deep Neural Netowrks). Il gruppo Analog-AI ha sviluppato un chip, ARES, in grado di eseguire l’operazione Multiply-Accumulate (MAC) utilizzando dispositivi a memoria a cambiamento di fase. Per dimostrare le prestazioni del chip, il modello ALBERT, una versione più compatta del ben noto trasformatore BERT, è attualmente oggetto di uno studio sperimentale.

In questo rapporto, viene fornita un’analisi generale e approfondita dei contributi al MAC, rivelando che alcune coppie di attivazioni/pesi assumono una maggiore importanza, mentre altre possono essere sicuramente eliminate con un impatto molto limitato sulla precisione. Viene proposta una nuova strategia di potatura basata sulle righe (row-wise pruning), seguita dalla messa a punto, che porta a una riduzione delle dimensioni del modello con una precisione equivalente.

L’algoritmo proposto viene quindi applicato al compito GLUE utilizzando l’architettura ALBERT, dimostrando prestazioni simulate equivalenti al software anche con una costante potatura dei pesi, consentendo potenzialmente diverse migliorie come la riduzione delle piastrelle hardware richieste, migliori prestazioni energetiche e una più semplice distribuzione del modello a livello di chip.

Introduction

Artificial intelligence models with billions of parameters can excel in accuracy across various tasks. However, they exacerbate the energy inefficiency of standard processors, like GPUs and CPUs. Analog in-memory computing (Analog-AI) offers a solution by operating matrix-vector multiplications efficiently in parallel across memory tiles, thus enhancing energy efficiency. Nevertheless, Analog-AI has yet to prove its software-equivalent accuracy for models requiring multiple tiles and efficient communication of neural network data between these tiles.

The Analog-AI team from IBM research in Almaden designed an Analog-AI chip, composed of 35M Phase-Change Memory devices spread across 34 tiles. To demonstrate its performance, a large model is implemented. The goal of this project is to reduce the size of the implemented model retaining a software equivalent accuracy.

The following report will first give the general context of this internship. An overview of Large Language Models (LLMs) will be given in order to explain the motivation of the project. Analog In-Memory Computing being a promising hardware innovation to reduce the latency and energy consumption of Deep Neural Network of LLMs will be presented. Then, the Analog-AI chip developed by the group will be studied. The following chapter will focus on transformer models, as the model implemented on chip, ALBERT is based on this architecture. The GLUE benchmark used to fine-tune ALBERT will be described. Finally, the challenging implementation on-chip of the ALBERT model will be explained. The third chapter will focus on the pruning technique used during the project to try to simplify the implemented model on-chip. Starting from an overview of pruning in deep learning, a new method based on the MAC study will be developed and applied to the ALBERT model specifically on one block of the transformer architecture. The last chapter will present the accuracy results of the pruned ALBERT model and its possible implementation on-chip.

Chapter 1

General Context

This first chapter is intended to explain the context of this internship. An overview of Large Language Models (LLMs) will be given to explain the motivation of the project. Analog In-Memory Computing is a promising approach to reduce the latency and energy consumption of Deep Neural Network like LLMs. The group developed a chip, that will be studied, capable to compute the MAC operation using non-volatile memories.

Contents

1 Large Language Models

Large Languages Models (LLMs) are deep learning algorithms that can perform a variety of Natural Language Processing (NLP) tasks. They are trained using massive datasets, often involving tens or hundreds of billions of parameters. This enables them to perform a wide range of natural language processing tasks, including text generation, translation, summarization, and question answering.[Chang *et al.*, 2023].

The recent gain of interest on LLMs is due to the breakthrough in Generative AI. In fact, ChatGPT based on BERT [Devlin *et al.*, 2018], powered by a set of language models developed by OpenAI, gathered over 100 million users in two months after its release in 2022. As a consequence, numerous competing models have emerged. The release of Llama2 by Meta in 2023 as an open source model enhanced even more the potential of LLMs [Touvron *et al.*, 2023].

However, the management of the large amount of data required for LLMs and the need to train them is an obstacle to the spread of these models. Large models can take a long time to train, and hardware that can handle the training process can be hard to come by and expensive to develop. To address this energy consumption issue, researchers are actively exploring techniques to make AI models more energy-efficient. This includes model reduction, quantization, and the development of smaller, specialized models for specific tasks. Model reduction is the solution that has been explored during the project. Even more than only modifying the models, energy consumption can also be widely reduced using Analog In-Memory Computing.

2 Analog In-Memory Computing

Analog In-Memory Computing (AIMC) is a promising approach to reduce the latency and energy consumption of Deep Neural Network (DNN) inference and training. In fact, large networks are trained and implemented using processors such as GPUs and CPUs, which lead to excessive energy consumption when large amounts of data must move between memory and processor (CPU) : this is the so-called Von-Neumann Bottleneck [Rasch *et al.*, 2021]. Thus, accelerators based on AIMC using non-volatile memories (NVMs) are currently the focus of active research. They are designed based on resistive memory device technologies such as Phase Change Memory (PCM), Resistive Random Access Memory (ReRAM), and Magnetic Random Access Memory (MRAM). They exhibit significant potential in enhancing the performance and reducing the energy consumption of deep learning systems. These accelerators harness the physical properties of these memory devices to perform computations at the same place where data is stored (Fig.1.1). AIMC could improve both run-time efficiency and power consumption compared to conventional digital computing technologies.[Wouters *et al.*, 2015]

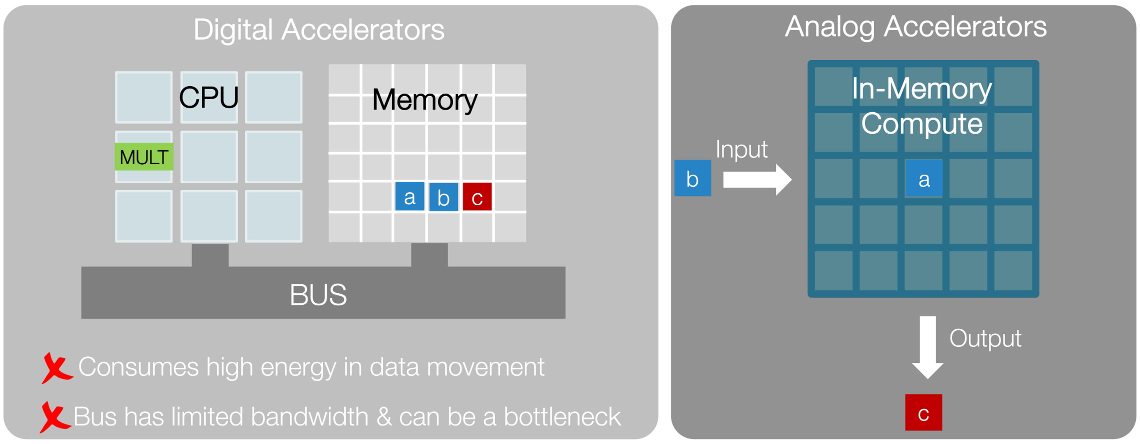


Figure 1.1: Digital accelerator (left) where the operation and memory are divided in two different entities. Analog accelerator (right) allows to perform the operation directly in the memory device.

3 MAC accelarator chip

The Analog-AI team of IBM Research Almaden developed an AIMC chip capable of computing the Multiply-Accumulate (MAC) operation using non-volatile memories, specially Phase Change Memory devices. This chip can offer better energy efficiency and throughput than digital accelerators. This section will focus on the architecture of the chip and its operation.

3.1 Architecture

The Analog-AI team developed a 14-nm all-analog inference chip (Fig.1.2(a)) with 34 analog tiles, two Processing Elements (PE) and six Input/Output Landing Pads (ILPs/OLPs) (Fig.1.2(b)). For data transmission, a 2D routing mesh is employed, allowing data to flow from input to tile, tile to tile, and tile to output. The data is conveyed in duration-format, where pulse widths signify excitation values (Fig.1.2(c)). Pulse-Width Modulators (PMWs) and duration-to-byte converters located at the chip's periphery facilitate the conversion between duration and digital bits at the ILPs and OLPs. Controllers on each tile configure the transmission and reception directions of the local mesh, enabling flexible routing topologies. Each tile stores 512 × 512 unique DNN weights, corresponding to 512x2048 PCM, and each weight consists of four PCM conductances labeled as (Fig.1.2(d)). There are 8.9M weights that can be programmed in one chip which corresponds to 35M PCM devices per chip. [Narayanan *et al.*, 2021]

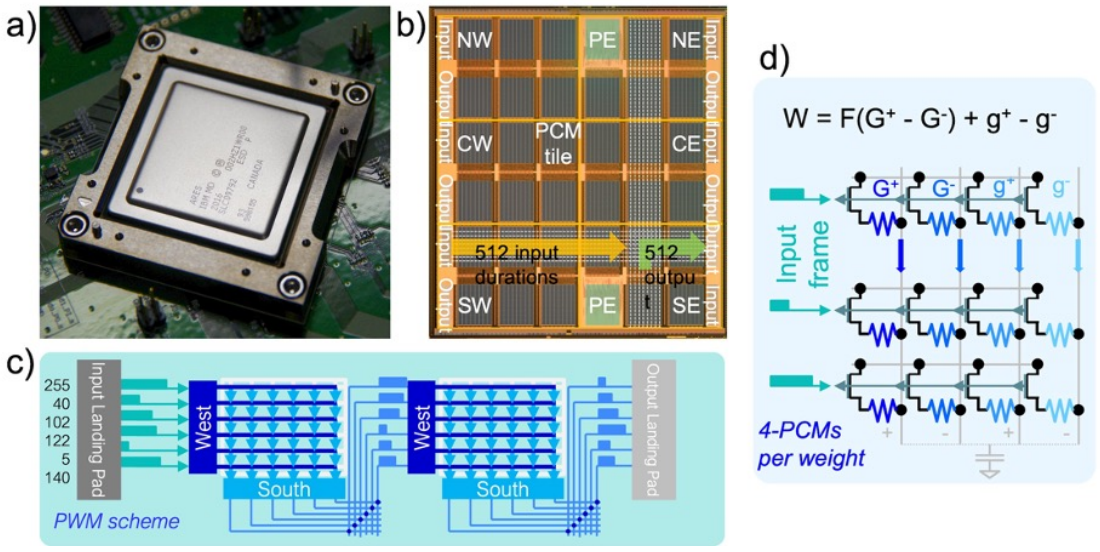


Figure 1.2: Chip architecture [Ambrogio *et al.*, 2023]

3.2 Phase Change Memory devices

The chip relies on the operation of Phase Change Memory (PCM) devices. A PCM cell is a two-terminal element, with a phase change material in between 2 electrodes. The bottom electrode acts as a heater (Fig.1.3(a)). The resistive switching material is a chalcogenide : the $Ge_2Sb_2Te_5$ (GST) that is electronically conducting in the crystalline phase and insulating in the amorphous phase. The memory information is stored in the atomic configuration of the material : the ordered state in the crystalline phase and the disordered state in the amorphous state. Current is used to switch between the two phases. It induces a temperature rise in the material, by Joule heating and thermoelectric effects. The cell can SET by crystallization of the amorphous phase to reach the conducting state (R~kOhms) and RESET by the melting of the crystalline phase into amorphous reaching the insulating state (R~MOhms) (Fig.1.3(b)). [Wouters *et al.*, 2015]

Interest in PCM devices is motivated by several advantages. They are non-volatile memories, allow read and write data at high speeds, have a low power consumption and high density. PCM cells can store and process analog data, not only binary information. They can retain a range of resistance values, corresponding to a continuum of analog voltage levels. PCM devices can store multiple states in a single cell. This is achieved by adjusting the amorphous-to-crystalline phase transition with different levels of current or voltage. Each resistance level can represent a different analog value, allowing PCM to perform multi-level analog computations. PCM devices can also perform parallel processing. Their ability to store multiple analog states in a single cell means they can simultaneously process multiple analog inputs, which is advantageous for tasks like matrix-vector multiplication in neural networks. For instance, PCM devices are arguably considered the most mature among resistive memory types. However, they suffer from temporal conductance drift, and temperature sensitivity (T » 100°C) that could make them hard to use in data centers.

3.3 Multiply–Accumulate operation

In the context of machine learning and neural networks, the term MAC refers to Multiply-Accumulate operations. These operations involve multiplying two numbers and then adding the result to an accumulator. The MAC operation is a fundamental building block in many mathematical computations and plays a crucial role in neural network training and inference. During inference, the MAC operation computes the weighted sum of inputs, which is often followed by an activation function to produce an output. During training, MAC operations are used to calculate gradients for weight updates through back-propagation.

PCM devices are used to perform the Multiply–Accumulate (MAC) operation on-chip. To perform the operation $W * x = y$ *(W is the weight matrix, x is the input vector)*, the elements of the matrix W are mapped linearly to the conductance values of the PCM devices on-chip organized in a crossbar configuration. The values of the input vector x are mapped linearly to the durations of read voltages and are applied to the crossbar along the rows. The charge measured along the columns of the array will be proportional to the result of the computation, y. On-chip, multiplication is performed using Ohm's law, and accumulation using Kirchhoff's law (Fig.1.3(c)).

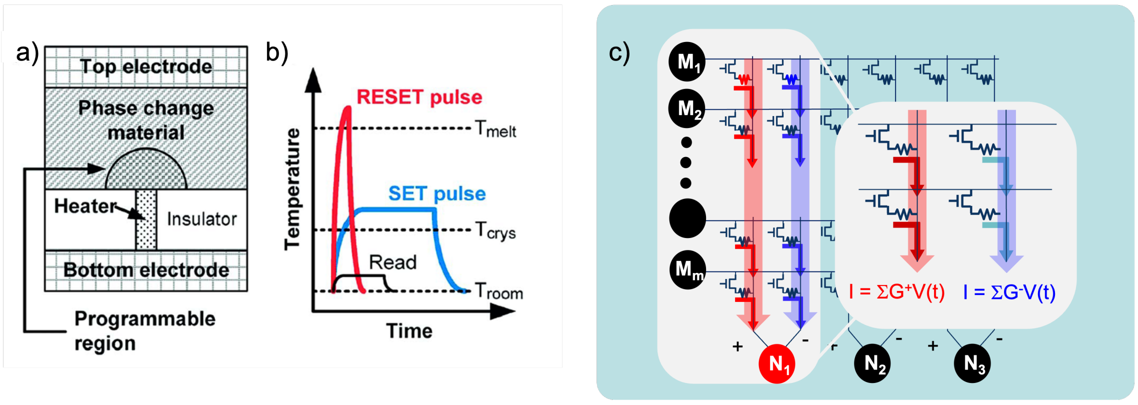


Figure 1.3: (a) and (b) Phase Change Memory device ([Wouters *et al.*, 2015]) used to perform (c) Multiply–Accumulate on-chip

In this chapter, we described the motivations behind Analog In-Memory Computing. To accelerate deep learning training and inference using less power energy, the Analog-AI team developed a chip which operation is based on PCM devices. The weights used for the MAC operation are implemented on-chip which allows efficient Matrix Vector Multiplication. To demonstrate the performance of the chip developed by the group, the ALBERT model, a more compact version of the widely known BERT transformer, is currently under experimental study. The next chapter will then focus on transformer-based models, BERT and ALBERT, and on the weight implementation of ALBERT on-chip.

Chapter 2

Transformer-based models

This chapter will give an overview of transformer models, in particular BERT and ALBERT models. The GLUE benchmark will be described. The on-chip implementation of the ALBERT model will be explained.

Contents

1 Transformers

Transformers were introduced in 2017 in the "Attention is All You Need" paper by [Vaswani *et al.*, 2017]. In that paper, authors proposed a completely new way of approaching deep learning tasks such as machine translation, text generation, and sentiment analysis. The three key elements that make transformers powerful are self-attention, positional embeddings and multi-head attention. The self-attention mechanism enables the model to detect the connection between different elements even if they are far from each other and assess the importance of those connections, therefore, improving the understanding of the context. Due to positional embeddings and multihead attention, transformers allow simultaneous sequence processing, which means that model training can be sped up through parallelization. This is a huge benefit of using transformers over architectures like Recurrent Neural Network (RNN) and has enabled the creation of Large Language Models.

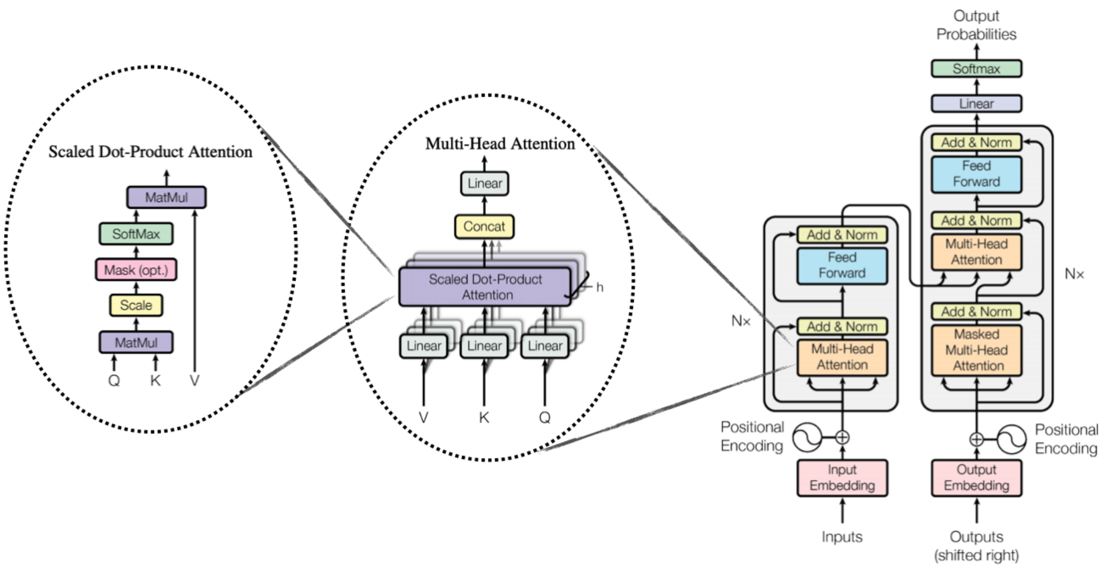


Figure 2.1: Encoder-Decoder Transformer architecture [Vaswani *et al.*, 2017]

The following section will describe the transformer architecture based on Fig.2.1.

The Encoder is on the left and the Decoder is on the right. Both the Encoder and Decoder modules can be stacked multiple times. These modules primarily consist of Multi-Head Attention and Feed Forward layers. To ease processing, the inputs and outputs (target sentences) are initially embedded into an n-dimensional space because direct string usage is not feasible. The positional encoding is used to give to every part in our sentence a relative position. These positions are added to the word embeddings. By incorporating positional embeddings into the input representation of an NLP model, the model is able to capture the order and position of words in a sequence.

In transformers, encoding and decoding are done by using attention. Attention helps the model to understand the context of a word by considering words that go before and after it. The context of other words in the sentence helps computer to understand different meanings of the word and translate accordingly. With the example sentence : "I could hear the dog bark", somewhere next to the word bark in the vocabulary one would find words "dog", "loudly", "car". Attention mechanism helps the model to analyze different parts of another sequence, which is the sequence that is being generated by the decoder.

Self-attention mechanism works with the sequence that is being encoded. Self-attention mechanism allows the model to weigh the importance of different parts of the input sequence against each other. By doing this, the model is able to effectively capture long-range dependencies in the input sequence and learn to recognize patterns that span multiple elements. The Scaled Dot-Product Attention is a specific mathematical formulation used within the self-attention mechanism. The following equation describes the Scaled Dot Product Attention,

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.1}$$

Q is a matrix that contains the query (vector representation of one word in the sequence), K are all the keys (vector representations of all the words in the sequence) and V are the values, which are again the vector representations of all the words in the sequence. d_k refers to the dimension of the key vectors. We can define,

$$a = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2.2}$$

$$Attention(Q, K, V) = a \cdot V \tag{2.3}$$

The weights a are defined by how each word of the sequence (represented by Q) is influenced by all the other words in the sequence (represented by K). Additionally, the SoftMax function is applied to the weights a to obtain a distribution between 0 and 1. Those weights are then applied to all the words in the sequence that are introduced in V.

The previous attention-mechanism can be parallelized into multiple mechanisms that can be used side by side. It refers to the so-called Multi-Head Attention. The attention mechanism is repeated multiple times with linear projections of Q, K and V. This allows the system to learn from different representations of Q, K and V, which is beneficial to the model. These linear representations are done by multiplying Q, K and V by weight matrices W that are learned during the training. The vectors responsible for tokens are broken up into multiple parts called heads which go through the same attention computing process as before. The results of the process are concatenated together to form an output of the same type.

The Feed-Forward layer is a position-wise transformation that consists of two linear transformations with a ReLU activation in between.

After the introduction of the transformer architecture, which proved highly effective in natural language processing, several models were developed based on it. BERT is one of these transformer-based models.

2 BERT model

The BERT (Bidirectional Encoder Representations from Transformers) model has been developed by researchers at Google AI Language in 2018, [Devlin *et al.*, 2018]. It demonstrated state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1) and Multi-Genre Natural Language Inference (MNLI).

BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in the previous section. In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. BERT model exists in two sizes : $\text{BERT}_{\text{BASE}}$ with L=12, H=768, A=12 and Total Parameters=110M and $\text{BERT}_{\text{BASE}}$ with L=24, H=1024, A=16 and Total Parameters=340M. A GELU activation is used rather than the standard RELU from the Transformer paper.

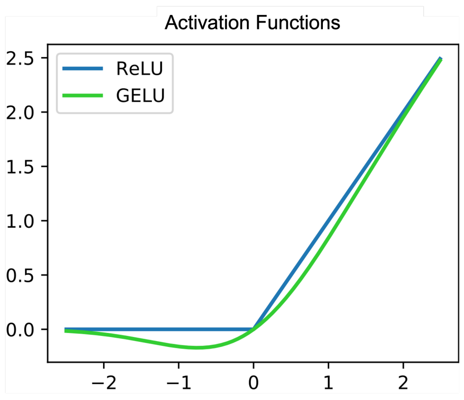


Figure 2.2: Activation Functions ReLU and GELU

BERT's key technical innovation is applying the bidirectional training of Transformer to language modelling. In order to train BERT, a portion of the input tokens is randomly masked and then those masked tokens are predicted. This procedure is called "masked language model" (MLM). The second pre-training task is the Next Sentence Prediction (NSP). It trains the model to understand sentences relationship. The model is then fine-tuned on different tasks and evaluated. Fine-tuning will be developed in the next sections.

BERT has demonstrated remarkable capabilities in understanding contextual information from text but comes with a high computational cost due to its large number of parameters. ALBERT was developed to alleviate this issue.

3 ALBERT model

ALBERT (A Lite BERT) was developed as an improvement of the BERT model to achieve greater parameter efficiency and computational scalability while maintaining or enhancing NLP performance. ALBERT reduces the number of parameters compared to BERT while maintaining or even improving performance. This reduction is achieved thanks to two techniques : cross-layer parameter sharing and factorized embedding parameterization [Lan *et al.*, 2019].

Cross-layer parameter sharing prevents the parameter from growing with the depth of the network. The parameters are shared between all the layers. Factorized embedding parametrization makes it easier to grow the hidden size without significantly increasing the parameter size of the vocabulary embeddings. In fact, the large vocabulary embedding matrix is decomposed into two small matrices, then, the size of the hidden layers is separated from the size of vocabulary embedding. At the end, ALBERT has 18x fewer parameters than BERT-large and can be trained 1.7x faster.

Model	Parameters	Layers	Hidden	Embedding Parameter-sharing
BERT base	108M	12	768	False
BERT large	334M	24	1024	False
ALBERT base	12M	12	768	True
ALBERT large	18M	24	1024	True
ALBERT xlarge	60M	24	2048	True
ALBERT xxlarge	235M	12	4096	True

Table 2.1: Comparison of Model Parameters and Characteristics [Lan *et al.*, 2019]

The ALBERT architecture is similar to BERT, using transformer encoder with GELU activation in the Feed-Forward layer. ALBERT model is pre-trained on MLM, same as BERT, but NSP is replaced by a sentence-order prediction (SOP) based primarily on coherence.

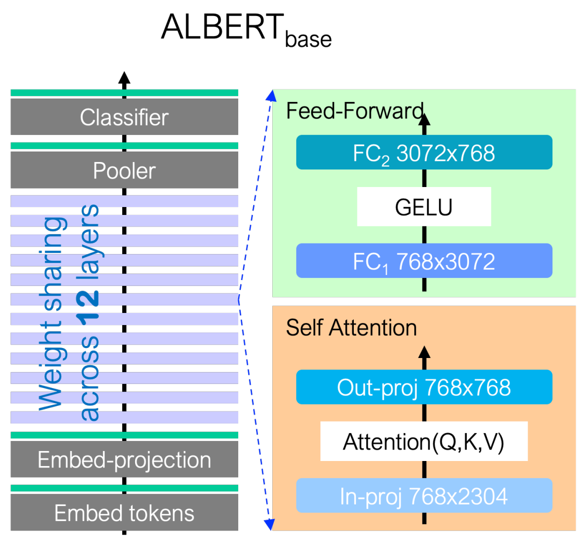


Figure 2.3: ALBERT architecture

Now that we have discussed the architectures of the BERT and ALBERT models, our next section will focus on fine-tuning these models.

4 GLUE benchmark

After pre-training, the model is used as a starting point for a new task. This new task is different from the original pre-training task. This step is called fine-tuning. Fine-tuning datasets vary depending on the specific downstream task to solve. The GLUE dataset (General Language Understanding Evaluation) includes a variety of NLP tasks, making it a versatile choice for fine-tuning [Wang *et al.*, 2018].

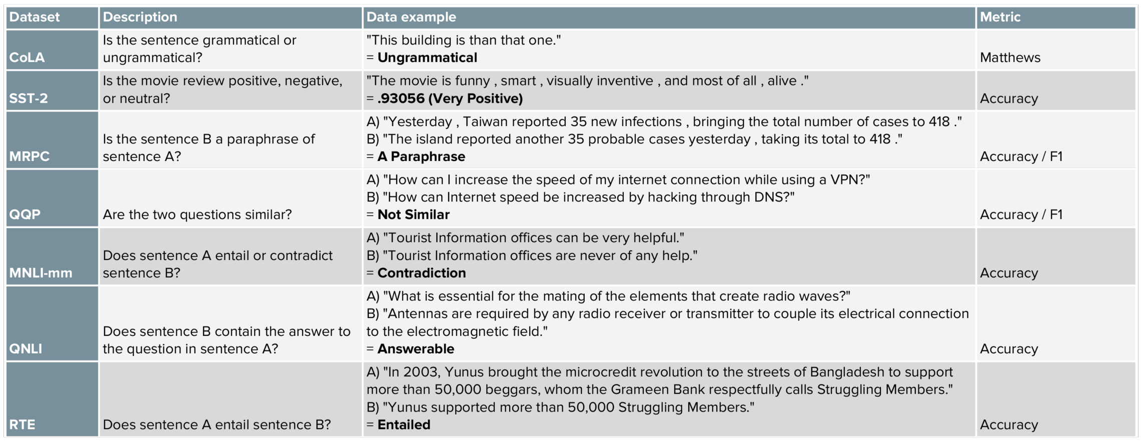
4.1 GLUE dataset description

The GLUE dataset is a benchmark dataset designed to evaluate the performance of natural language understanding models (NLU) and systems across a range of language tasks. GLUE consists of several diverse tasks that cover various aspects of language understanding, including question answering, sentiment analysis, and textual entailment. GLUE is composed by a suite of nine sentence or sentence-pair NLU tasks, built on established annotated datasets and selected to cover a diverse range of text genres, dataset sizes, and degrees of difficulty. Only seven have been be use in this project. There are three different type of tasks : the single-sentence, the paraphrase and the inference.

The single-sentence tasks are the Corpus of Linguistic Acceptability (CoLA) and the Stanford Sentiment Treebank (SST-2). CoLA consists of English acceptability judgments drawn from books and journal articles on linguistic theory. It focuses on determining if a sentence is linguistically acceptable or not. SST-2 consists of sentences from movie reviews and human annotations of their sentiment. The aim of the task is to predict the sentiment of a given sentence.

The Microsoft Research Paraphrase Corpus (MRPC) and the Quora Question Pairs (QQP) are paraphrase tasks. QQP dataset is a collection of question pairs from the community question-answering website Quora. The task aims to determine whether a pair of questions are semantically equivalent. MRPC is a corpus of sentence pairs automatically extracted from online news sources. Similar to QQP, models are required to determine if two sentences are paraphrases of each other.

Finally, the inference tasks are the Multi-Genre Natural Language Inference Corpus (MNLI), the Stanford Question Answering Dataset (QNLI) and the Recognizing Textual Entailment. MNLI is a crowd-sourced collection of sentence pairs with textual entailment annotations. For this task, models are required to determine if a hypothesis is entailed by, contradicts, or is neutral with respect to a given premise. QNLI is a question-answering dataset consisting of question-paragraph pairs. Models are asked to determine if an answer to a given question can be inferred from a passage of text. Finally, RTE comes from a series of annual textual entailment challenges. This task assesses if a given hypothesis can be inferred from a provided text.



Dataset	Description	Data example	Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	Accuracy / F1
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	Accuracy

Figure 2.4: GLUE tasks

4.2 BERT and ALBERT fine-tuning on GLUE tasks [Lan *et al.*, 2019]

[Wang *et al.*, 2018] reported state-of-the-art accuracy results on the GLUE benchmark. The ALBERT configuration is ALBERT-xxlarge using combined MLM and SOP losses, and no dropout. The BERT configuration is BERT-large using combined MLM and NPS losses, and dropout.

Model	MNLI	QNLI	QQP	RTE	MRPC	CoLA
BERT-large	86.6	92.3	91.3	70.4	88.0	60.6
ALBERT-xxlarge	90.8	95.3	92.2	89.2	90.9	71.4

Table 2.2: State-of-the-art accuracy results on the GLUE benchmark

The previous sections described the BERT model and its lite version, ALBERT. The accuracy results obtained for ALBERT are consistent. Then, to evaluate the performances of the chip developed by the team, it is more interesting to implement ALBERT model as it contains less parameters so the final implementation will require the use of smaller number of chips.

5 ALBERT implementation on-chip

To demonstrate the performance of the chip, the ALBERT model is used. The 7.7M weights of ALBERT$_{\text{BASE}}$ are implemented on-chip, which can contain up to 8.9M weights (35M PCM devices).

As it is transformer based, each layer among the 12 of the ALBERT$_{\text{BASE}}$ model is composed of a Self Attention and a Feed-Forward layer. The Self Attention is divided in three blocks : the In-projection, the Attention, and the Out-Projection. For the Feed-Forward, we can find the Feed Forward Neural Network 1 (FC1), the GELU activation and the Feed Forward Neural Network 2 (FC2). The different operations performed in each block are displayed in Fig.2.5.

The In-Projection block performs linear operations multiplying the input with the Query (Q), Key (K), and Value (V) weight matrices and adding a bias. This operation can be done on-chip as the weight matrices are stationary. The inputs will be sent on-chip through the Input Landings Pads.

The operations performed in QK, Softmax, and QKV blocks are not implemented on-chip. They are the Attention operations. For demonstration purpose, they are done in software off-chip. The QK and QKV blocks cannot be implemented on-chip as they are performing operations between non stationary terms. Implementing Q, K, or V on-chip will require to re-program the tiles for each new input. The Softmax block computes a non-linear operation that is to complicated to be performed with PCM devices,

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.4}$$

The Out-Projection block performs a linear operation multiplying the output of QKV with the weight matrix of the Out-Projection block, and then adding a bias. This operation can be done on-chip as the weight matrix is stationary.

The output of the Self-Attention is then normalized. The normalization operation is linear. In this block, one can notice that the input is added to the Self-Attention layer output adding even more complexity. Thus, this operation is done off-chip.

For the next blocks, FC1 and FC2, have the same structure as In-Projection and Out-Projection. They are implemented on-chip. The GELU activation is non-linear, and the second normalization layer is similar to the first one with a different input.

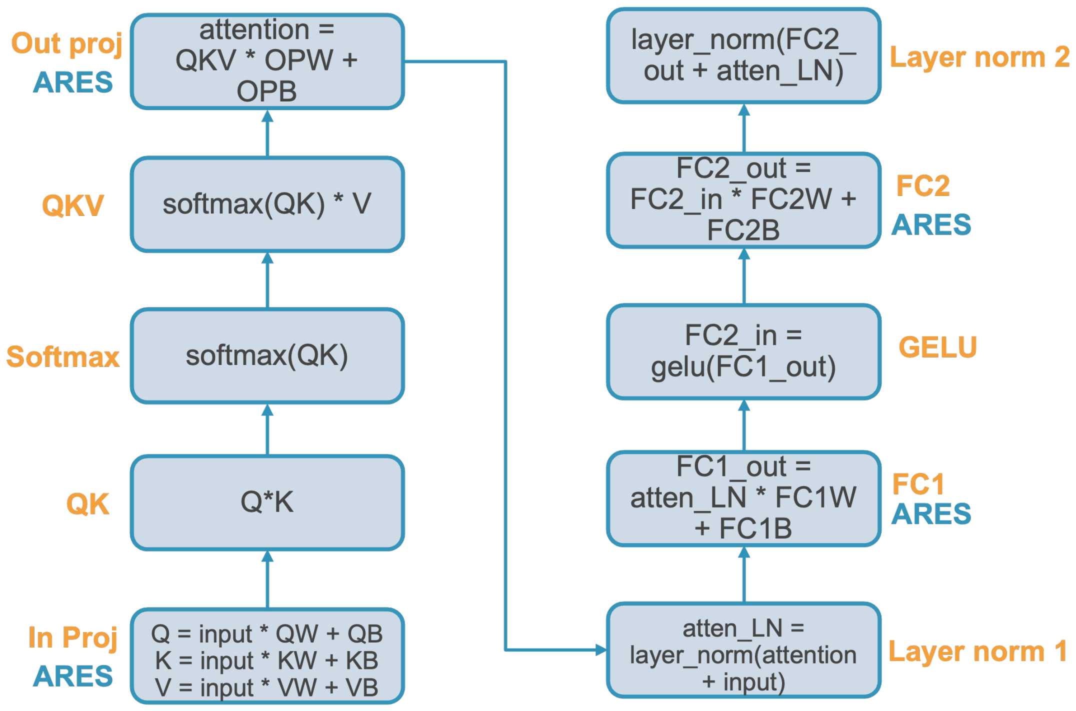


Figure 2.5: ALBERT transformer operation

To summarize, only basic and stationary operations can be performed on-chip. At the end, QW, KW, VW, OPW, FC1W and FC2W are implemented on one chip. This represents 7.7M weights programmed on the chip developed by the team, which can contain up to 8.9M weights (35M PCM devices). Thus, the model mapping across tiles is complex due to non-trivial input/output activation paths.

Fig.2.6 represents the mapping of the ALBERT model on-chip. Focusing on the first chip representation, the mapping appears compact, there are only two half tiles free. It is due to the fact that almost the maximum capacity of the weight implementation is reached out, but also due to the need to rout the input and output signal having only 6 input/output pads, so we have to wisely use the tiles to distribute the inputs. The three chips remaining from Fig.2.6 represent the three separate routings corresponding to the inputs. Three other routings are needed for the outputs. The different routings will be operated sequentially. Additionally, as one tile can only be filled with 512x512 weights, the layers are split in multiple tiles.

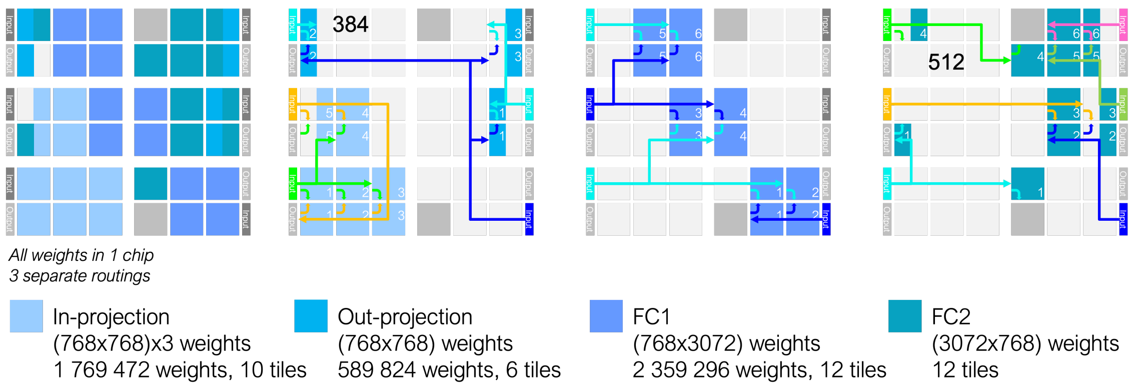

384
512
All weights in 1 chip
3 separate routings
In-projection (768x768)x3 weights 1 769 472 weights, 10 tiles
Out-projection (768x768) weights 589 824 weights, 6 tiles
FC1 (768x3072) weights 2 359 296 weights, 12 tiles
FC2 (3072x768) weights 12 tiles

Figure 2.6: ALBERT model mapping on-chip

The In-Projection block is composed of QW, KW and VW. The three matrices have a dimension of (768, 768). The row implementation on chip is as follows : each matrix is divided by 2, so 384 rows are implemented in each tile. The 2304 (768*3) columns are divided between 4 tiles with 512 columns and 1 tile with 256 columns. The total implementation is distributed over 10 tiles, corresponding to 1 769 472 weights implemented.The Out-Projection block is composed by OPW, with a size of (768, 768). Six half tiles are used with 384 rows implemented per tile and 256 columns. FC1W is implemented for the FC1 block. Its weight matrix is composed of (768, 3072) weights. The implementation is distributed over 12 tiles, with 384 rows and 512 columns per tile. Finally, the dimension of FC2W is different from the other weight matrix as it contains 3072 rows instead of 768. The distribution is as follows : the 6 tiles are filled with 512 rows and 512 columns (maximum capacity) and 6 with 512 rows and 256 columns.

This chapter described transformer-based models, with a focus on BERT and ALBERT. ALBERT model has been chosen to be implemented on-chip for its high capability with respect to its parameters. The weights of the model are implemented in one chip. Routing the ALBERT model on-chip is a hard task which will become even harder with larger models. For this reason, the project focused on reducing the model size using weight pruning. The pruning technique developed and its operation on the ALBERT model will be discussed in the next chapter.

Chapter 3

Pruning

This third chapter will focus on the pruning technique used during the project. Starting from an overview of pruning in deep learning, we will develop a new method based on the MAC study. This method is applied to the ALBERT model specifically on one block of the transformer architecture

Contents

1 Definition

Pruning in deep learning is used to reduce the size of a neural network by selectively removing certain neurons, connections, or parameters while attempting to maintain or even improve the model's performance. Pruning is primarily applied to reduce the computational and memory requirements of a model, making it more efficient or easier to implement.[Han *et al.*, 2015]

There are different types of pruning, each focusing on different part of the neural network : weight pruning, neuron pruning and structured pruning. In weight pruning, less important weights of the neural network are set to zero or removed. It can be done based on different criteria, such as magnitude of the weights or their importance to the model's performance. Neuron pruning technique consists in removing nodes from a neural network's layer. In structured pruning entire blocks are removed from the neural network. For example, an entire layer can be removed.

Successful pruning processes have been observed in the literature in recent years. The paper "Progressive DNN Compression" (2019) [Ye *et al.*, 2019] introduces an approach to deep neural network (DNN) compression by progressively applying pruning and quantization techniques. The authors propose a two-step compression process that begins with layer-wise pruning, followed by filter-level pruning and weight quantization. [Frankle *et al.*, 2019] introduce the "lottery ticket hypothesis," showcasing the potential of sparse, trainable sub-networks that can outperform their dense counterparts.

In the context of this project, pruning is applied on transformer-based models that also generated significant attention in recent research. [Wang *et al.*, 2019] propose a structured pruning approach targeting attention heads within large language models, showcasing its potential for memory reduction while maintaining task performance. Additionally, [Michel *et al.*, 2019] investigate the necessity of a large number of attention heads in transformer models. The authors demonstrate that a smaller number of heads can achieve comparable performance, leading to potential pruning opportunities and increased efficiency.

Based on the literature search, the need of a specific pruning method to adapt to Analog-AI was needed. In fact, the project requires structured weight pruning taking into account the input activation. The next sections will present a developed technique based on the contribution of each weight in the MAC.

2 Development of a weight pruning technique

2.1 MAC study

Let's consider two matrices: W, which is the weight matrix (n,n), and X, which is the input matrix (n,m),

$$W = \begin{bmatrix} w_{0,0} & \cdots & w_{0,\mathrm{n}} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ w_{\mathrm{n},0} & \cdots & w_{\mathrm{n,n}} \end{bmatrix} \quad X = \begin{bmatrix} x_{0,0} & \cdots & x_{0,\mathrm{m}} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ x_{\mathrm{n},0} & \cdots & x_{\mathrm{n,n}} \end{bmatrix}$$

The MAC operation is as follows,

$$W \cdot X = \begin{bmatrix} y_{0,0} & \cdots & y_{0,\mathrm{m}} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ y_{\mathrm{n},0} & \cdots & y_{\mathrm{n,n}} \end{bmatrix} = Y$$

For better understanding we focus on the first MAC element for explanations. It is possible to categorize the product elements, *x*w*, among positive product elements and negative product elements,

$$y_{00} = \sum_{j=0}^{n} w_{0,j} \cdot x_{j,0} = w_{0,0} \cdot x_{0,0} + w_{0,1} \cdot x_{1,0} + \ldots + w_{0,n} \cdot x_{n,0}, \quad (w_{0,j} \cdot x_{j,0}) \in \mathbb{R} \quad (3.1)$$

$$y_{00} = \sum_{l} w_{0,l} \cdot x_{l,0} + \sum_{k} w_{0,k} \cdot x_{k,0}, \quad (w_{0,l} \cdot x_{l,0}) \in \mathbb{R}^{+}, \quad (w_{0,k} \cdot x_{k,0}) \in \mathbb{R}^{-} \quad (3.2)$$

At the end the MAC$_{0,0}$ is divided in two parts as follows,

$$\boldsymbol{y_{00} = MAC^{+}{}_{0,0} + MAC^{-}{}_{0,0} = MAC_{0,0}}$$

The product elements are then sorted in descending order for the positive side, and ascending order for the negative one.

$$MAC^{+}{}_{0,0} = \sum w_{0,l} * x_{l,0} = y^{+}{}_{max} + \ldots. + y^{+}{}_{min} \qquad MAC^{-}{}_{0,0} = \sum w_{0,k} * x_{k,0} = y^{-}{}_{min} + \ldots. + y^{-}{}_{max}$$

descending sorted **ascending sorted**

We obtain the following graph (left) that highlights the contribution of each element from the most to the least important one to the MAC$_{0,0}$.

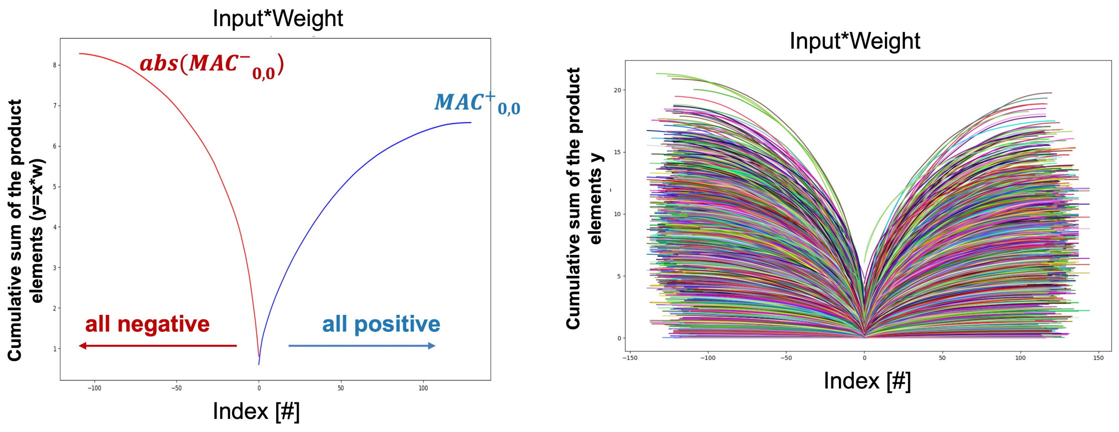


Figure 3.1: Contribution of the product elements to the first input frame MAC (left) and to the all the input frame MACs (right)

Focusing on the first part of Fig.3.1, the first values show a sharp slope. This sharp slope indicates that a limited amount of elements makes a significant contribution to the overall MAC computation. In contrast, for the last values, the curves flatten. This phenomenon signifies that a large number of elements exhibit low contribution to the MAC operation. Consequently, a few *(input * weight)* pairs have a strong influence, highlighting their crucial role in the computation. Other pairs may hold less impact and could potentially be disregarded. The next step is to find a method to be able to track the index of the low contribution elements.

2.2 Product elements contribution

The objective of this sub-section is to develop a metric that could help understand how each product element, *(w*x)*, fractionally contributes to the final MAC,

$$\Delta = \frac{w \cdot x}{MAC}, \tag{3.3}$$

With this metric, we compute the product element divided by the MAC corresponding to the input frame. For example, for $w_{0,0}$,

$$\Delta_{0,0} = abs(\frac{w_{0,0} \cdot x_{0,0}}{MAC_0}) + abs(\frac{w_{0,0} \cdot x_{1,0}}{MAC_1}) + ... + abs(\frac{w_{0,0} \cdot x_{0,m}}{MAC_m}), \tag{3.4}$$

After the computation of the metric Δ for all the corresponding weights, we obtain a matrix of the same size of the weight matrix (n,n). Each element corresponds to its contribution to the MAC taking into account the activation and partial MAC for each input frame.

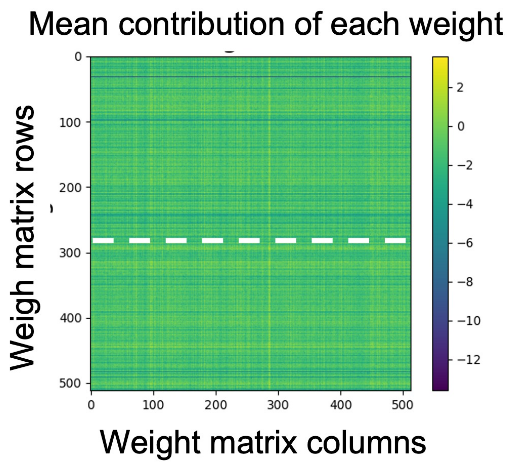


Figure 3.2: Matrix of the weight contributions

Each point of the Fig.3.2 represents the contribution of the corresponding weight. Dark blue corresponds to low MAC contribution. In our case, we focus on the entire row contribution. In fact, for hardware application removing entire rows allows not to program the corresponding tiles.

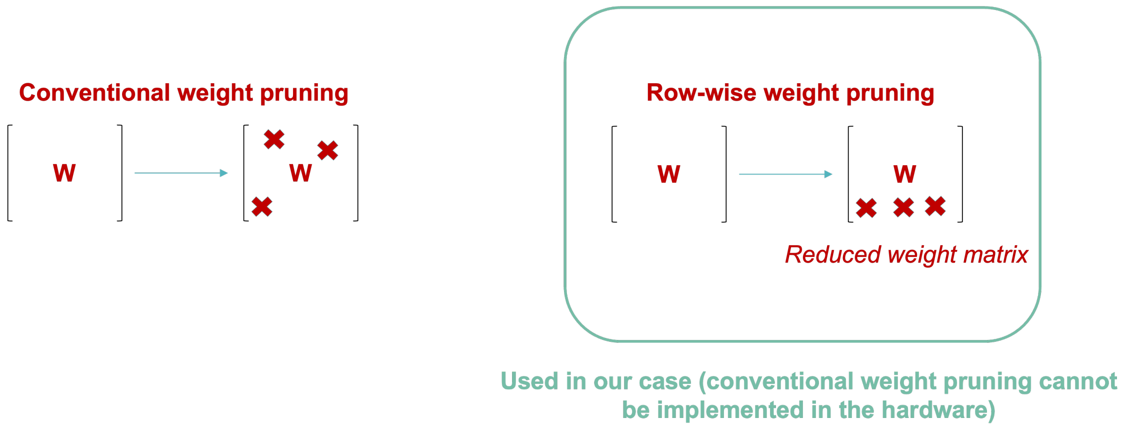

Conventional weight pruning
W
W
Row-wise weight pruning
W
W
Reduced weight matrix
Used in our case (conventional weight pruning cannot be implemented in the hardware)

Figure 3.3: Conventional vs. row-wise pruning

Finally, the Δ metric has been used to compute the matrix of the contribution of the weights. This matrix is then used to detect the least important rows that will be removed from the model. The pruning technique developed could be used on any model as it only relies on the MAC study. In this project, we apply it on ALBERT model.

3 Pruning on ALBERT model

3.1 FC2 pruning decision

After building the previous pruning method, we need to apply it to the ALBERT model. A study has been conducted to choose with block of the ALBERT model should be pruned. Indeed, pruning all the blocks and layers leads to an important drop in accuracy. We decided to focus on pruning for all layers only for one block. As a reminder, Fig.3.4 shows the blocks of the ALBERT model. In-Projection, Out-Projection, FC1 and FC2 blocks had been studied as they are implemented on-chip.

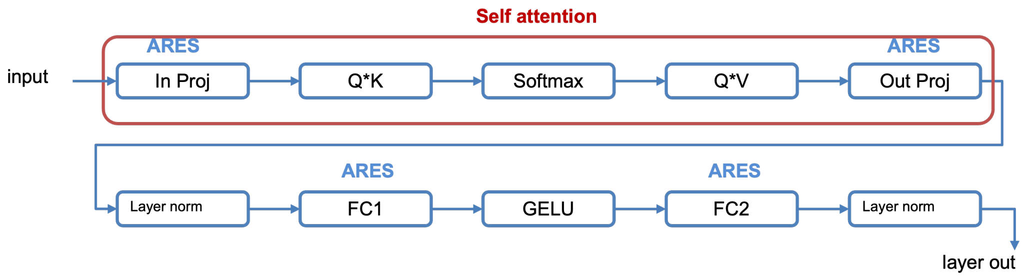


Figure 3.4: ALBERT architecture

The pruning capability refers to the number of weight rows we can free from each block and the ability to retain accuracy. The following Fig.3.5 is a reminder of the section ALBERT implementation on-chip (5). The distribution of the weights across the tiles for all the blocks is displayed, indicating the number of rows and columns implemented for each tile.

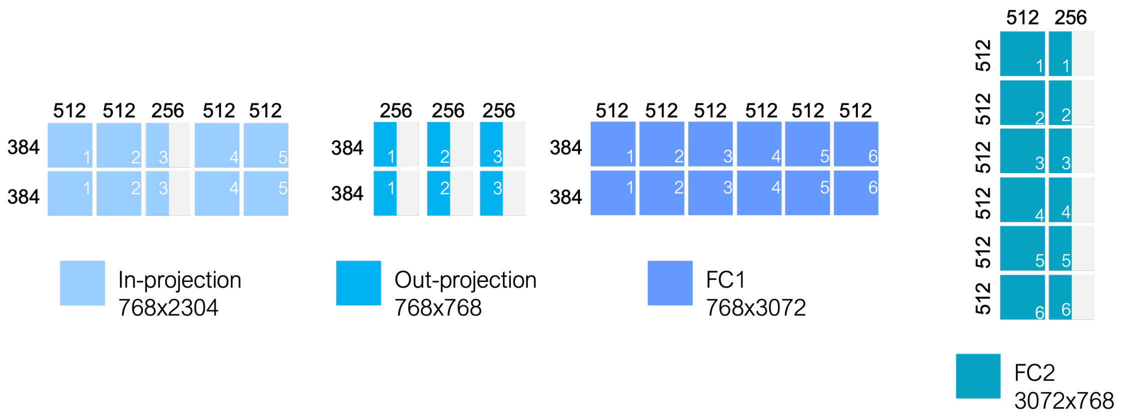


Figure 3.5: Distribution of rows and columns on-chip

As our goal is to be able to free tiles, the minimum rows to pruned for In-Projection, Out-Projection and FC1 is 256 over 768 (33%). In fact, removing 256 will free an entire row of tiles. The tiles will be implemented with 512 rows and 512 or 256 columns. For FC2, we should remove 512 over 3072 (17%). The corresponding distribution of rows and columns for each block is displayed Fig.3.6.

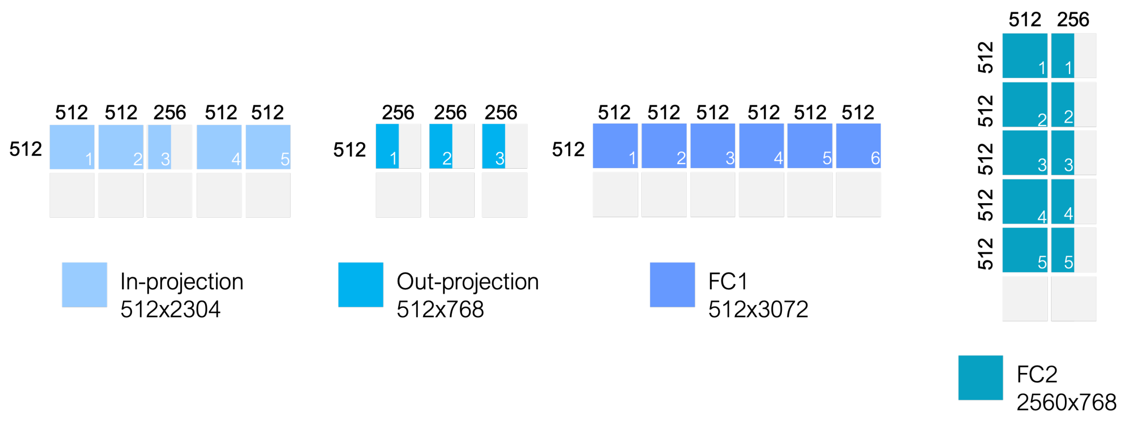


Figure 3.6: Potential distribution of rows and columns on-chip with pruning

Removing 256 rows from In-Projection, Out-Projection or FC1 will respectively free 5, 3 or 6 tiles. For FC2, the removal of 512 rows will free 2 tiles from FC2 block but also 2 tiles from FC1. In fact, the input of FC2 is directly related to the output of FC1 (Fig.2.5). As a consequence, removing 512 rows from FC2 will also remove 512 columns from FC1. At the end, 4 tiles could be removed. The following table summarizes the pruning capability of each block.

Table 3.1: Pruning capability

Block	# Rows	# Weights Removed	# Tiles	% Weights removed
In Proj	256/768	589,824	5	8.3%
Out Proj	256/768	196,608	3	2%
FC1	256/768	786,432	6	11%
FC2	512/3072	786,432	4	11%

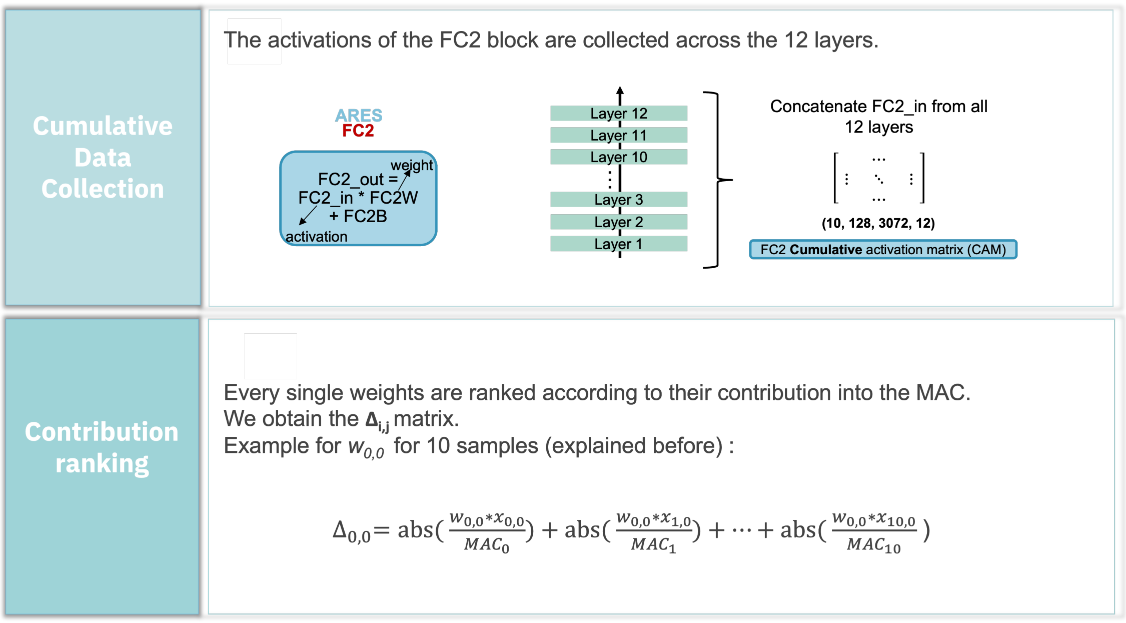
Finally, FC2 block shows a good resilience against pruning, the number of rows to remove from this block is less important in percentage with respect to the total number of rows of the matrix with respect to the other blocks (17% vs. 33%) so pruning would be less aggressive on FC2 and allows to free 4 tiles and 11% of the total number of weights of the model. FC2 has both good resilience and high pruning capability. Thus, to develop the pruning technique and be able to have consistent results for the study, FC2 has been chosen as block. 512 rows will be removed from this block, which correspond to 4 tiles in the chip implementation.

3.2 Row-wise weight pruning on FC2

As explained in the previous sub-section, 512 rows are removed from the FC2 block. In practice, the weights of the ALBERT model are implemented on one single chip. In fact, the weights are shared among the 12 layers. It is then possible to use only one chip to implement the weights that remain the same for all the layers. The chip is re-used with the output of one layer applied as the input for the next one. Consequently, to deploy the pruning technique, it has to be applied to all the layers. We will finally remove 512 rows from the FC2 block across the 12 layers.

Fig.3.7 describes the different steps to prune the FC2 block. All the computations are done with Python. After the identification of the rows to prune, we apply a mask to the FC2 weights to set at zero the 512 least important rows (Fig.3.8).

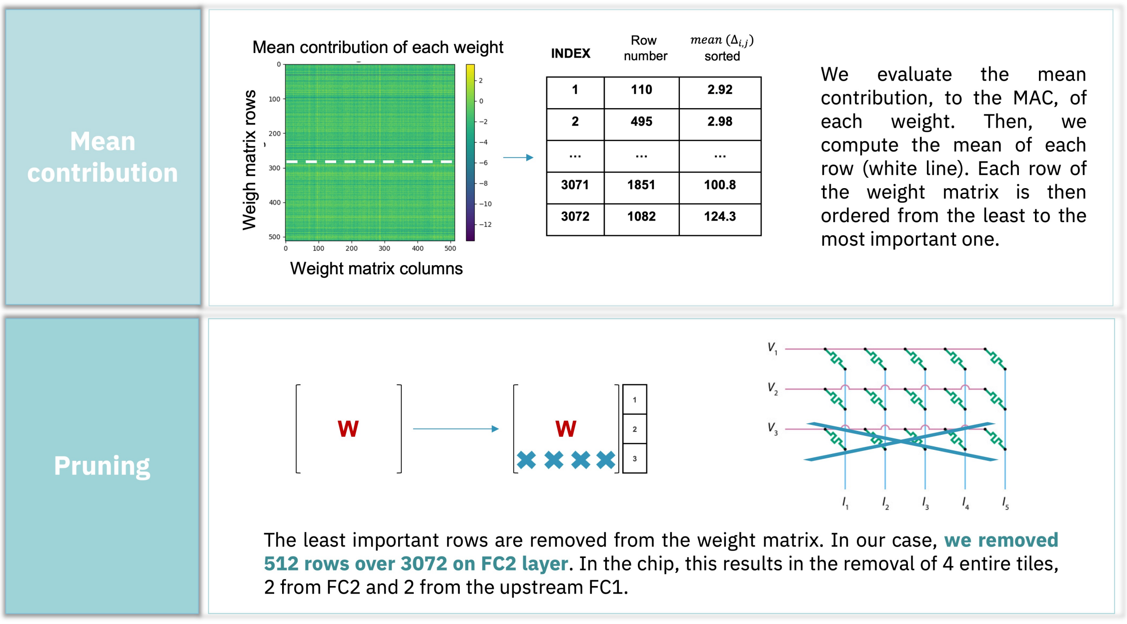
In this part, we studied the MAC operation to develop a pruning technique allowing to remove entire rows from a deep learning model, the ALBERT model in our case. Reducing the model size simplifies the implementation of the weights on-chip and their debug. In an Analog-AI application it also reduces propagation noise. The next chapter of this report will focus on the results we obtained on GLUE tasks. Key parameter of this study is the accuracy. In fact, pruning can degrade accuracy. In our case, we aim to keep the baseline accuracy reducing the model size as described before.


Cumulative Data Collection
ARES
FC2
weight
FC2_out = FC2_in * FC2W + FC2B
activation
Layer 12
Layer 11
Layer 10
Layer 3
Layer 2
Layer 1
Concatenate FC2_in from all 12 layers
(10, 128, 3072, 12)
FC2 Cumulative activation matrix (CAM)
Contribution ranking

The activations of the FC2 block are collected across the 12 layers.

Every single weights are ranked according to their contribution into the MAC.
We obtain the $\Delta_{i,j}$ matrix.
Example for $w_{0,0}$ for 10 samples (explained before) :

$$\Delta_{0,0}= \text{abs}(\frac{w_{0,0}*x_{0,0}}{MAC_0}) + \text{abs}(\frac{w_{0,0}*x_{1,0}}{MAC_1}) + \cdots + \text{abs}(\frac{w_{0,0}*x_{10,0}}{MAC_{10}})$$



INDEX	Row number	$mean\ (\Delta_{i,j})$ sorted
1	110	2.92
2	495	2.98
…	…	…
3071	1851	100.8
3072	1082	124.3

We evaluate the mean contribution, to the MAC, of each weight. Then, we compute the mean of each row (white line). Each row of the weight matrix is then ordered from the least to the most important one.

The least important rows are removed from the weight matrix. In our case, **we removed 512 rows over 3072 on FC2 layer**. In the chip, this results in the removal of 4 entire tiles, 2 from FC2 and 2 from the upstream FC1.

Figure 3.7: FC2 pruning steps

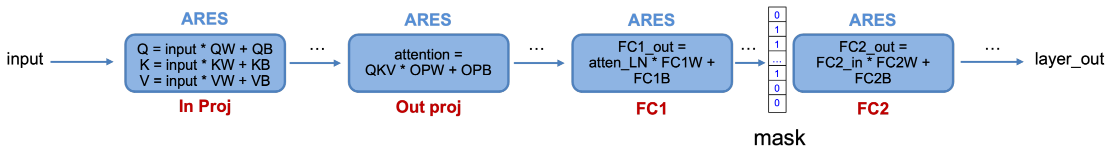


Figure 3.8: Mask application to FC2 weights

Chapter 4

Results on GLUE tasks

After pruning 512 rows on FC2 layer, the accuracy of the model has been evaluated on the GLUE dataset. The pruned model has been then trained. The last subsection will show the advantages of the improved routing of the chip using the pruned model.

Contents

1 Accuracy results

The ALBERT$_{\text{BASE}}$ model has been evaluated on the GLUE dataset. The model 'albert-v1' has been imported from HuggingFace. The Python code to perform fine-tuning and pruning has been built using content from AIHWKIT ([Rasch *et al.*, 2021]) and the following Jupyter notebook from Datasets repository from Hugging Face (hugging face notebook).

Before pruning the model, the accuracy on the different GLUE tasks have been evaluated. The following table summarizes the fine-tuning parameters used and the accuracy result for each task.

Table 4.1: Fine-Tuning Results

Task	Learning Rate	Epochs	ALBERT$_{\text{BASE}}$
CoLA	2e-5	20	0.54
MRPC	2e-5	20	0.8627
QQP	2e-5	5	0.9028
QNLI	1e-5	10	0.9059
MNLI	1e-5	5	0.8235
RTE	5e-6	20	0.7292
SST2	1e-5	10	0.8990
Average	-	-	**0.808**

Pruning has been performed on the fine-tuned model. 512 rows have been removed from the FC2 block, so also 512 columns from FC1. The accuracy with pruning obtained on validation dataset is compared with the one without pruning (SW). The result is displayed in Fig.4.1.

The average accuracy obtained on the GLUE dataset is 0.75. The average accuracy dropped down to 6%, from 0.81 to 0.75, compared to the baseline one. The results are encouraging but could be improved fine-tuning the pruned model.

2 Training

In order to close the gap in accuracy between the imported ALBERT model and the pruned one, training has been performed on the pruned model. The parameters, learning rate and epochs are the same as Table 4.1.

Fig.4.2 shows that training allows expected improvements. The average GLUE accuracy obtained with the pruned model is 0.803 whereas the average accuracy for the non-pruned model is 0.808. The baseline accuracy has been recovered. The pruning technique allows to reduce the model size by 11 percent and to potentially free 4 tiles in the chip implementation retaining baseline accuracy. Next section will focus on a possible improved routing using the pruned ALBERT model.

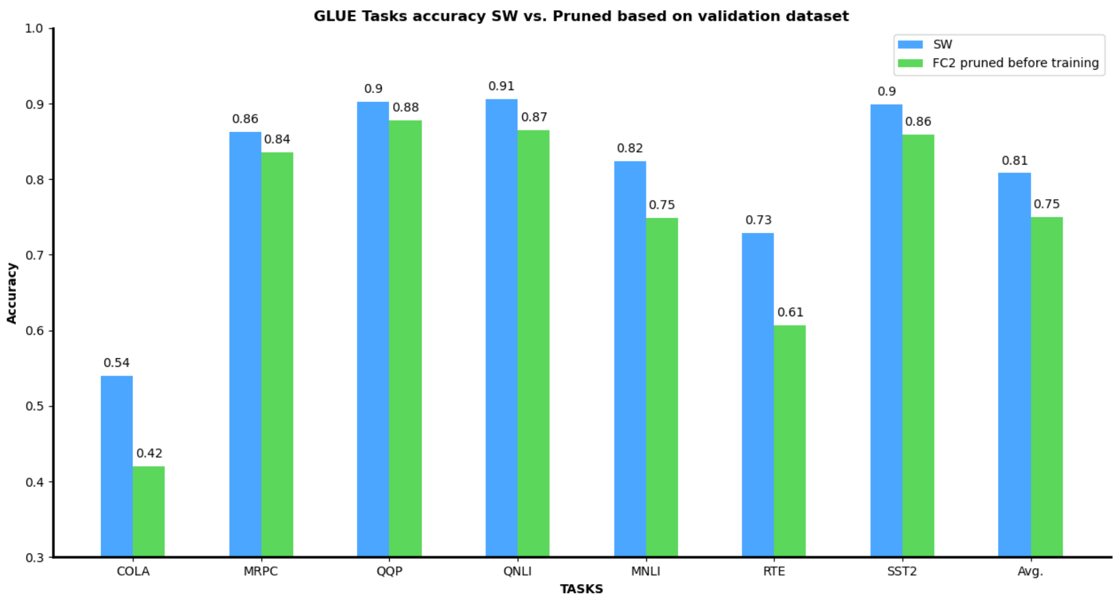


Figure 4.1: Glue Tasks accuracy results

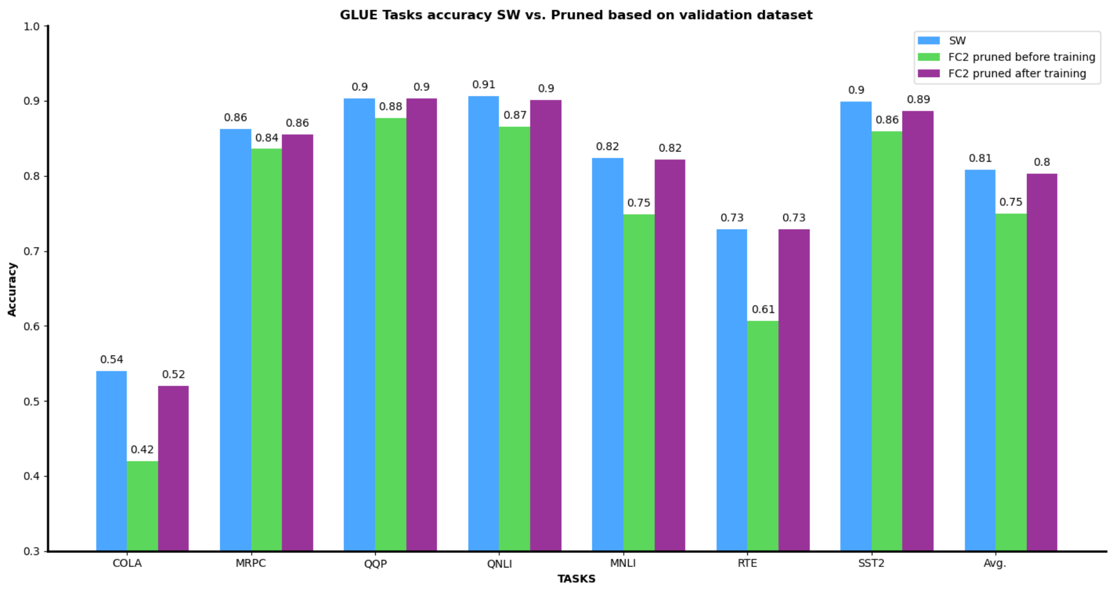


Figure 4.2: Glue Tasks accuracy results after pruned model fine-tuning

3 Improved routing

The previous sections show the complexity of ALBERT implementation on-chip. Pruning 512 rows on FC2 would free 4 tiles of the chip. Fig.4.3 suggests an improved routing.

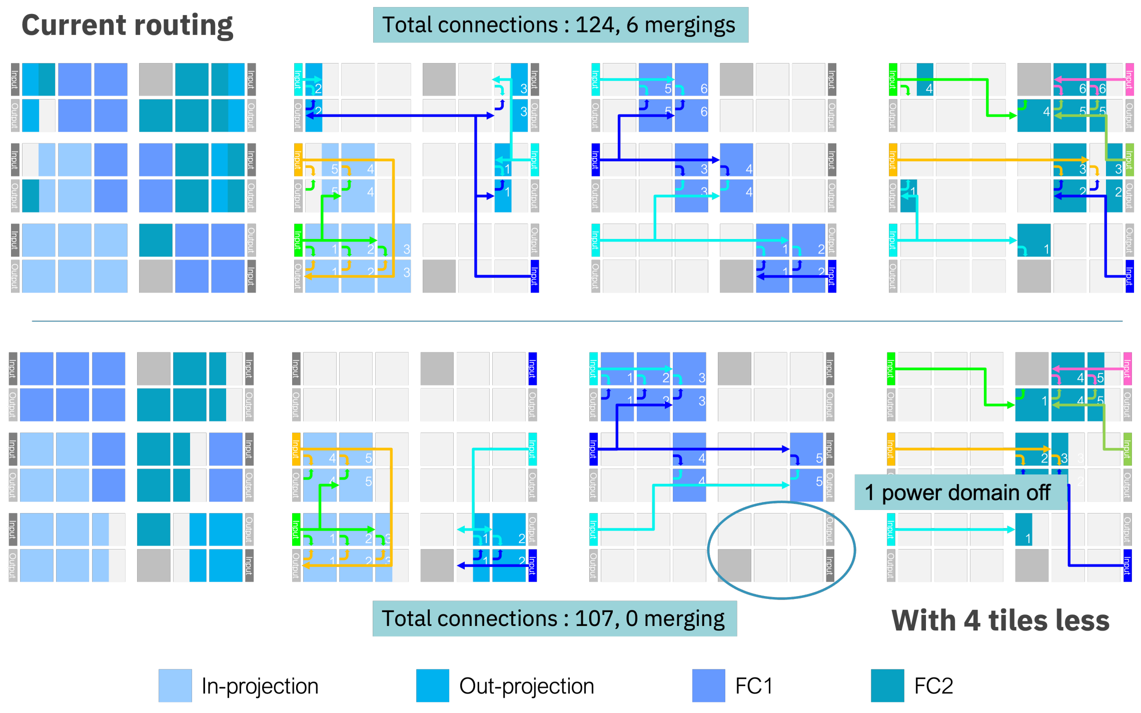


Figure 4.3: Current vs. Improved routing with 4 tiles less

The routing with 4 tiles removed is less compact than the one of the current route on-chip. It leads to a fewer number of connections, 107 instead of 124. This improvement could help reducing the noise as it usually increases with the propagation length. Pruning also allows to avoid merging. In fact, they are not shared tiles between different blocks on the improved routing whereas there are 6 in the current routing. Finally, the routing with 4 tiles less shows a power domain off that could help reducing power consumption. The overall distribution is easier, so would allow simpler debug. This routing is only a proposal and could be even more improved.

Conclusion

The exponential growth in the size of Large Language Models has forced researchers to seek innovations in order to sustain AI progress. In this context, new types of chips using In-Memory Computing have been developed such as the one from the Analog-AI group of IBM Research, Almaden. The chip is capable of operating MAC operation faster than conventional digital accelerators. The MAC operation on-chip is performed using the conductances of PCM devices. In order to demonstrate the performances of the chip, the ALBERT model is implemented on-chip. ALBERT is a transformer encoder-based model using weight sharing across its 12 layers. It is composed of 12M parameters whose 7.7M are implemented on-chip. The parameters programmed on-chip are the weights from the In-Projection, Out-Projection, FC1 and FC2 blocks. As the chip can contain up to 8.9M weights, the implementation of 7.7M weights is challenging due to signal routing constraints. As a consequence, a pruning technique to reduce the model size has been developed. This technique is based on the mean contribution of the weights into the MAC. It is applied to the FC2 block of ALBERT to remove 512 rows, which corresponds to 4 tiles less in the on-chip implementation. To validate the pruned model, its accuracy on the GLUE tasks has been computed. After the fine-tuning of the pruned model, baseline accuracy is recovered. According to the simulations, the pruned model could be implemented on-chip without reducing the performances of the chip. A proposal of an improved routing with 4 tiles less shows possible improvements in noise propagation, energy efficiency and allows simpler debug. This work could be further enhanced by fine-tuning the pruned model using hardware-aware training, specifically tailored for Analog-AI. Additionally, implementing the pruned model on-chip for experimental performance testing is a promising avenue for future research.

Bibliography

[Ambrogio *et al.*, 2023] AMBROGIO, Narayanan, O. F. M. H. N. Y. C. F. I. L. K. S. B. C. *et al.* (2023). An analog-ai chip for energy-efficient speech recognition and transcription. *Nature*.

[Chang *et al.*, 2023] CHANG, Y., WANG, X., WANG, J., WU, Y., ZHU, K., CHEN, H., YANG, L., YI, X., WANG, C., WANG, Y. *et al.* (2023). A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109.*

[Devlin *et al.*, 2018] DEVLIN, J., CHANG, M.-W., LEE, K. et TOUTANOVA, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

[Frankle *et al.*, 2019] FRANKLE, J., DZIUGAITE, G. K., ROY, D. M. et CARBIN, M. (2019). Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611.*

[Han *et al.*, 2015] HAN, S., POOL, J., TRAN, J. et DALLY, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

[Lan *et al.*, 2019] LAN, Z., CHEN, M., GOODMAN, S., GIMPEL, K., SHARMA, P. et SORICUT, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942.*

[Michel *et al.*, 2019] MICHEL, P., LEVY, O. et NEUBIG, G. (2019). Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

[Narayanan *et al.*, 2021] NARAYANAN, P., AMBROGIO, S., OKAZAKI, A., HOSOKAWA, K., TSAI, H., NOMURA, A., YASUDA, T., MACKIN, C., LEWIS, S. C., FRIZ, A. *et al.* (2021). Fully on-chip mac at 14 nm enabled by accurate row-wise programming of pcm-based weights and parallel vector-transport in duration-format. *IEEE Transactions on Electron Devices*, 68(12):6629–6636.

[Rasch *et al.*, 2021] RASCH, M. J., MOREDA, D., GOKMEN, T., LE GALLO, M., CARTA, F., GOLDBERG, C., EL MAGHRAOUI, K., SEBASTIAN, A. et NARAYANAN, V. (2021). A flexible and fast pytorch toolkit for simulating training and inference on analog crossbar arrays. *In 2021 IEEE 3rd international conference on artificial intelligence circuits and systems (AICAS)*, pages 1–4. IEEE.

[Touvron *et al.*, 2023] TOUVRON, H., MARTIN, L., STONE, K., ALBERT, P., ALMAHAIRI, A., BABAEI, Y., BASHLYKOV, N., BATRA, S., BHARGAVA, P., BHOSALE, S. *et al.* (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288.*

[Vaswani *et al.*, 2017] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł. et POLOSUKHIN, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wang *et al.*, 2018] WANG, A., SINGH, A., MICHAEL, J., HILL, F., LEVY, O. et BOWMAN, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461.*

[Wang *et al.*, 2019] WANG, Z., WOHLWEND, J. et LEI, T. (2019). Structured pruning of large language models. *arXiv preprint arXiv:1910.04732.*

[Wouters *et al.*, 2015] WOUTERS, D. J., WASER, R. et WUTTIG, M. (2015). Phase-change and redox-based resistive switching memories. *Proceedings of the IEEE*, 103(8):1274–1288.

[Ye *et al.*, 2019] YE, S., FENG, X., ZHANG, T., MA, X., LIN, S., LI, Z., XU, K., WEN, W., LIU, S., TANG, J. *et al.* (2019). Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm. *arXiv preprint arXiv:1903.09769.*

Gantt Diagram

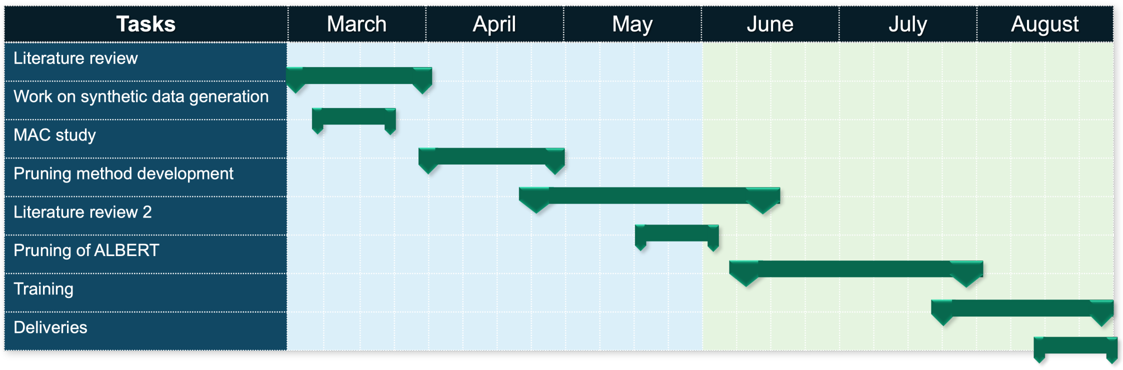


Figure 4.4: Gantt Diagram

Summary Sheet

Master Nanotech
2022/2023
IBM Research
650 Harry Rd, San Jose, USA

Pruning ALBERT transformer for Analog-AI

from March to August 2023

IBM Supervisor :
AMBROGIO Stefano
stefano.ambrogio@ibm.com
Phelma Supervisor :
PREJBEANU Liliana
liliana.buda@cea.fr

Student :
BOULHARTS Emma
41901635
emmaboulharts@gmail.com

Project Description

The Analog AI group at IBM Almaden studies in-memory computing to accelerate deep learning workloads, such as image recognition, speech, and language modeling. The intern work will consist of an exploration of possible pruning methods on ALBERT transformer, a compact model, part of the BERT transformer family, which performs Natural Language Processing on industry-relevant tasks (GLUE tasks).

Resources

IBM Research provides to the student an access to a private office with all the facilities for her work including a computer and second screen. The student has also access to the electrical laboratory managed by his supervisor and the Cognitive Computing Cluster of the company.

Surpervision

In addition to her supervisor, Stefano Ambrogio, the student is advised by the different research members of the team. Weekly Friday meetings allows to follow the progress of the student who has to give presentations regularly to the team for even deeper follow-up. The student also has access to conferences taking place in the Almaden offices.