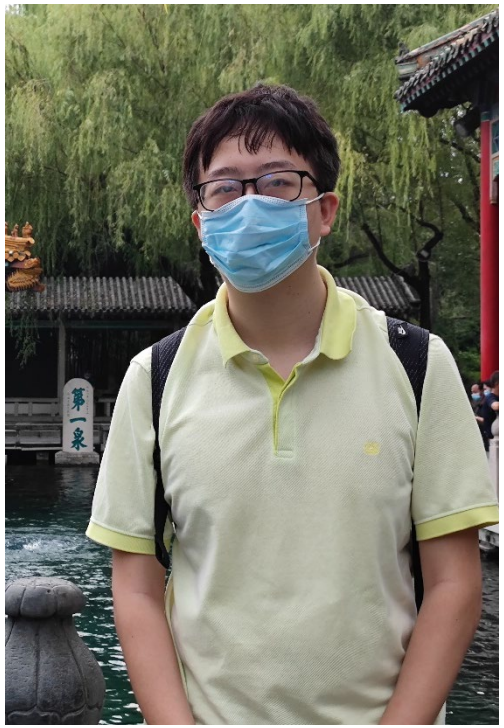
ELMA: Early-Exit Offloading for Embedded Question Answering Applications

Capstone Design
Dec 15, 2021

Group:	3
Instructor:	Prof. An Zou
Sponsor:	UM-SJTU Joint Institute
Group Member:	Yihua Liu, Shuocheng Chen, Yiming Ju

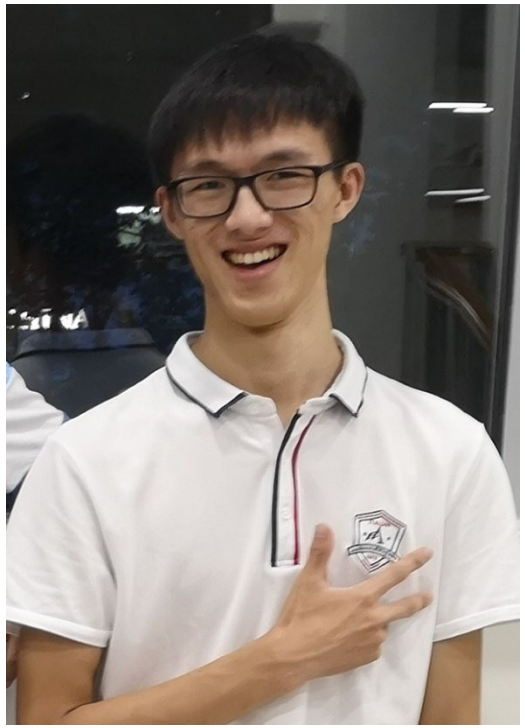
Team Members

JOINT INSTITUTE
交大密西根学院



Yihua Liu

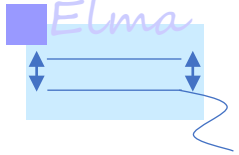
Shuocheng Chen



Yiming Ju

Overview

- Introduction
- Design Specifications
- Concept Generation & Selection
- Design Description
- Implementation & Validation
- Discussion & Conclusion



1. Introduction

Introduction

JOINT INSTITUTE
交大密西根学院

NLP

Embedded System

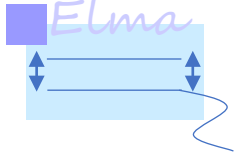
16000 words / s

0.2 s

Cloud

Large computation

High performance



2. Design Specifications

Design Specifications

Customers Requirements

Functionality	Question answering on embedded systems	Similar accuracy
Efficiency	Network latency	Faster prediction
Security	Personal sensitive data	A safer method
Cost	Need network connection	Work without network

Design Specifications

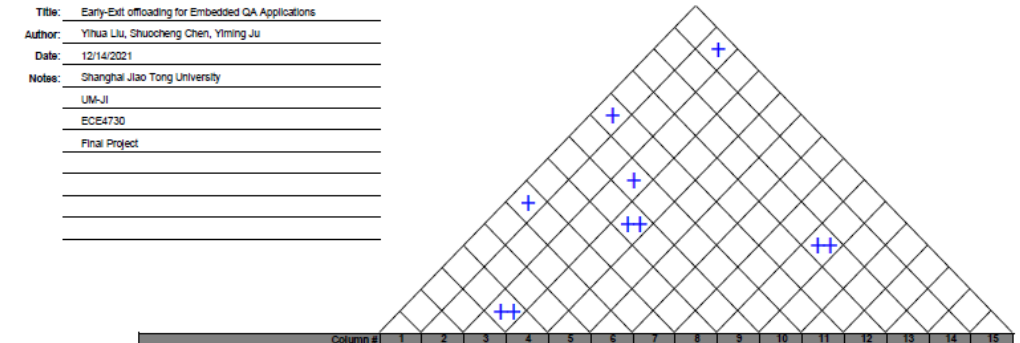
Engineering Specifications

Functionality	Train on the cloud and predict locally	Offloading
Efficiency	The whole model for prediction is large	Early exit
Security	Encryption and decryption consume resources	Simplify
Cost	Storage and computation on the cloud	Work remotely

Design Specifications

JOINT INSTITUTE 交大密西根学院

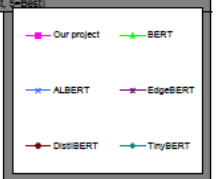
Title: Early-Exit offloading for Embedded QA Applications
Author: Yihua Liu, Shuocheng Chen, Yiming Ju
Date: 12/14/2021
Notes: Shanghai Jiao Tong University
UM-JI
ECE4730
Final Project

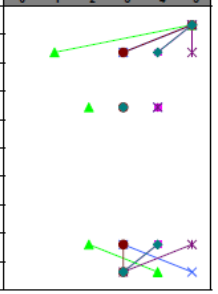


Legend

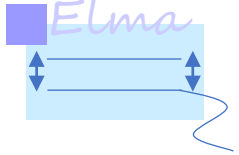
Θ	Strong Relationship	9
O	Moderate Relationship	3
▲	Weak Relationship	1
++	Strong Positive Correlation	
+	Positive Correlation	
−	Negative Correlation	
▼	Strong Negative Correlation	
▼	Objective Is To Minimize	
▲	Objective Is To Maximize	
x	Objective Is To Hit Target	

Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	Training set on the cloud	Prediction offloading	Prediction time	A more flexible model	Simplify the process of encryption and decryption	The cost of storage and computation on the cloud	Dataset Size on the cloud	Training time	Number of model parameters	Power consumption on embedded system	Power consumption on the cloud	Average response time to a request for prediction	Space occupation	Network latency	Support prediction free of network	Our project	BERT	ALBERT	EdgeBERT	DistilBERT	TinyBERT
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	x	x	▼	x	x	▼	▲	▼	▼	▼	▼	▼	▼	▼	x	Competitive Analysis (0=Worst, 5=Best)						
1	9	13.5	10.0	Run the task question answering	Θ															5	5	5	5	5	5
2	9	13.5	10.0	Can be applied on embedded systems		Θ		Θ												4	1	3	5	3	4
3	9	5.4	4.0	Lower network latency	O													Θ							
4	9	12.2	9.0	Similar or better performance	O		Θ	O	Θ		Θ	Θ				Θ				4	2	4	4	3	3
5	9	10.8	8.0	Protect personal sensitive data	Θ				Θ																
6	9	8.1	6.0	Work without a network connection	O	O													Θ						
7	9	10.8	8.0	Save power	Θ	Θ		O		Θ				Θ	Θ										
8	9	6.8	5.0	Have a good transportability		O		Θ																	
9	9	10.8	8.0	Occupy less space on embedded systems		▲							Θ				Θ			4	2	3	5	3	4
10	9	8.1	6.0	Have a good documentation for reference	▲	Θ		Θ												3	4	5	3	3	3
				Target or Limit Value			1s				100,000	60h		10W		0.5s	32G								
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	3	9	10	9	6	6	3	7	8	9	5	10	9	6	6						
				Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9						
				Weight / Importance	401.4	347.3	109.5	324.3	206.8	97.3	109.5	109.5	97.3	97.3	97.3	109.5	97.3	48.6	73.0						
				Relative Weight	17.3	14.9	4.7	13.9	8.9	4.2	4.7	4.7	4.2	4.2	4.2	4.7	4.2	2.1	3.1						





Powered by QFD Online (http://www.QFDOnline.com)



3. Concept Generation & Selection

Concept Generation & Selection

JOINT INSTITUTE
交大密西根学院

Dataset selection

SQuAD2.0
The Stanford Question Answering Dataset

Environment Selection

NVIDIA
CUDA

Develop Kit Selection

Concept Generation & Selection

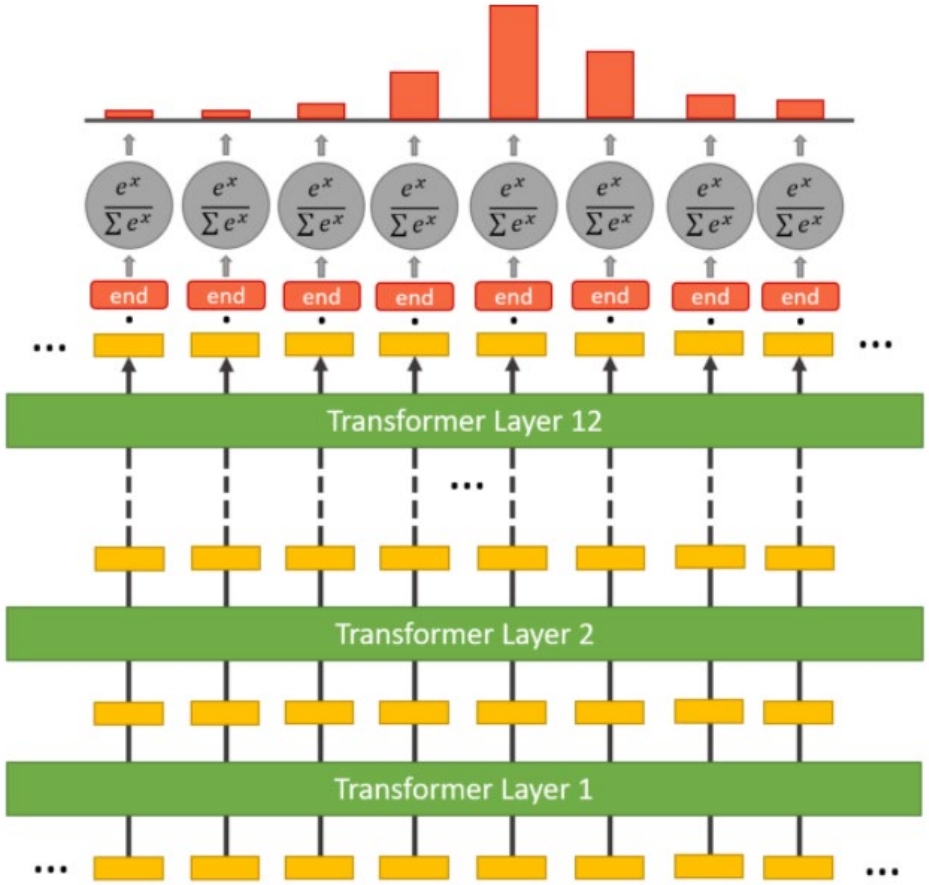
BERT vs. ALBERT Tensorflow vs. PyTorch

Criterion	Weight(%)	BERT				ALBERT			
		Tensorflow		PyTorch		Tensorflow		PyTorch	
		Score	Rating	Score	Rating	Score	Rating	Score	Rating
Performance	0.35	8	2.8	8	2.8	9	3.15	9	3.15
Space occupation	0.35	4	1.4	4	1.4	7	2.45	7	2.45
Flexibility	0.2	3	0.6	8	1.6	3	0.6	8	1.6
Derivative	0.1	7	0.7	5	0.5	7	0.7	5	0.5
Total			5.5		6.3		6.9		7.7

ALBERT + Pytorch

Concept Generation & Selection

ALBERT

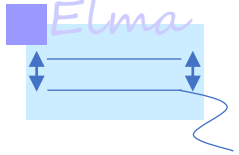


Parameter Sharing → Save memory space

Embedding Factorization

Sentence Order Prediction

→ Better performance

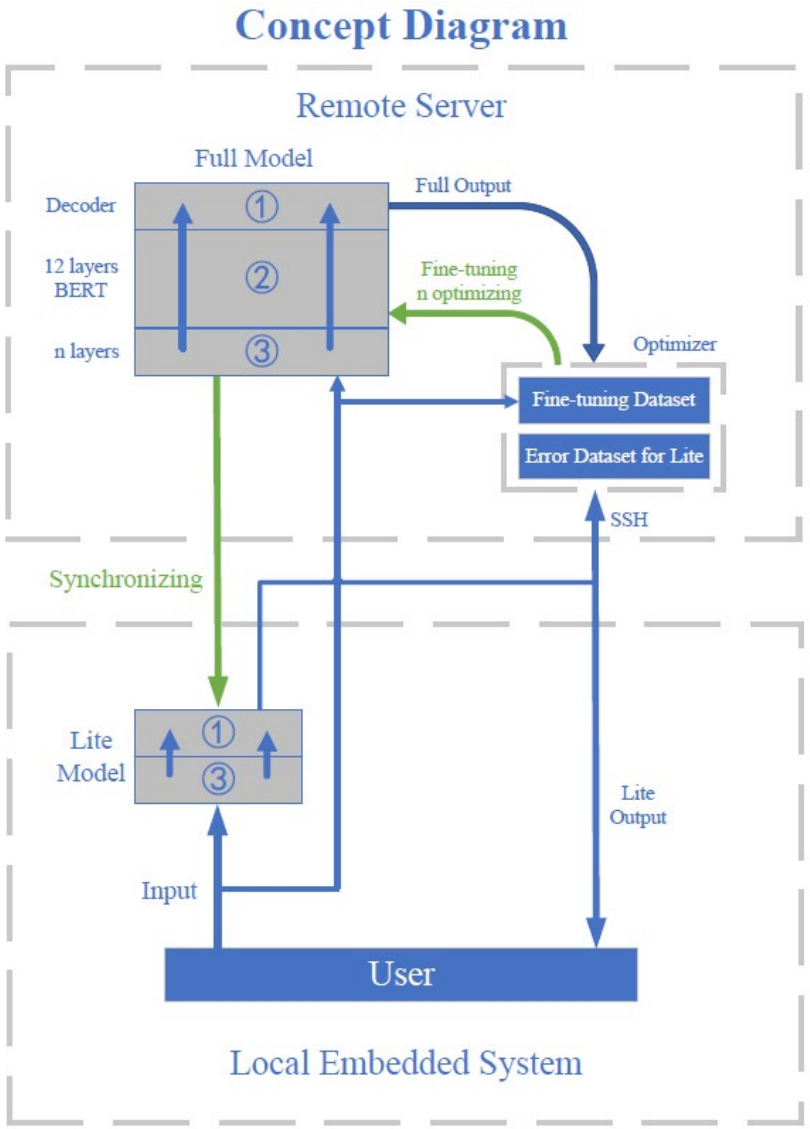


4. Design Description

Design Description

Remote Server： Full model

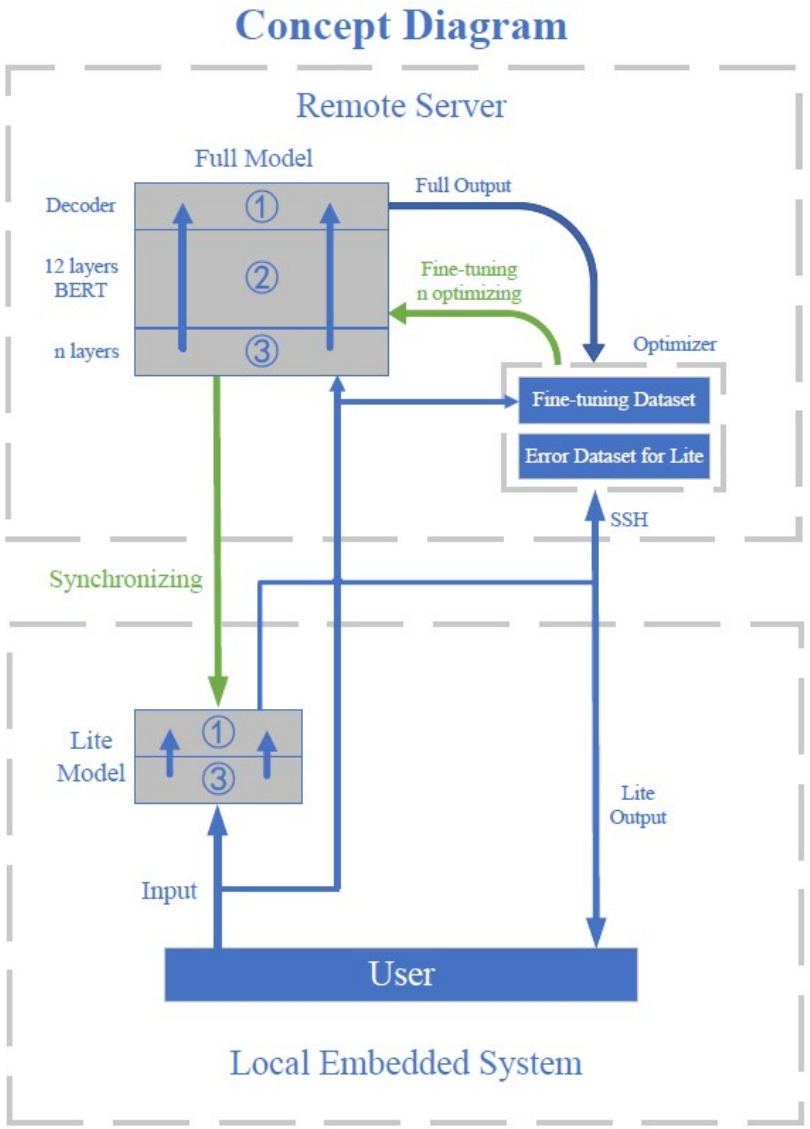
Embedded System: Lite model



Design Description

Remote Server： Full model

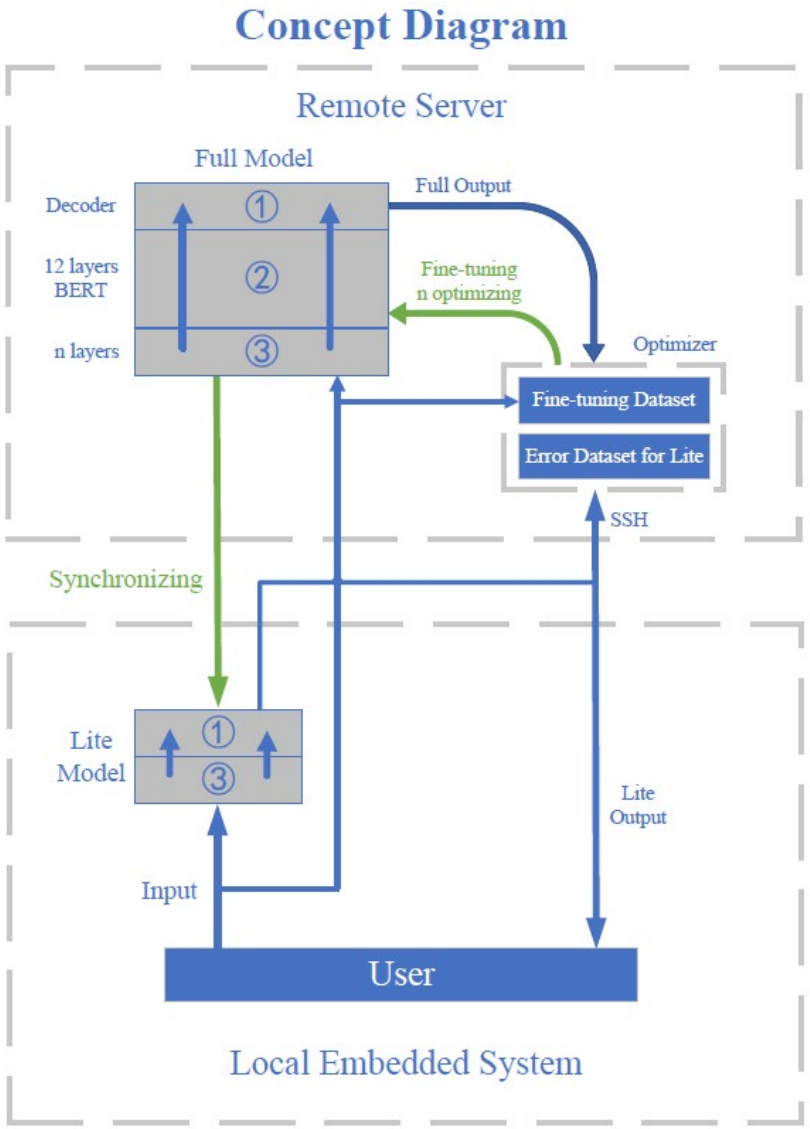
- 12 layers ALBERT
- Optimizer
- Fine-tuning



Design Description

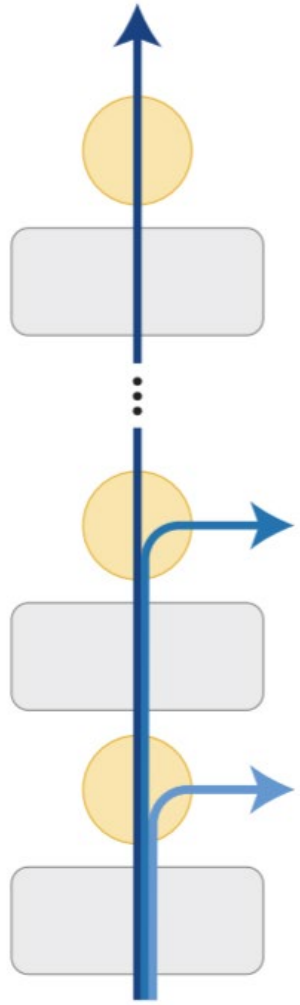
Embedded System: Lite model

- **Early-Exit**
- **Synchronization**

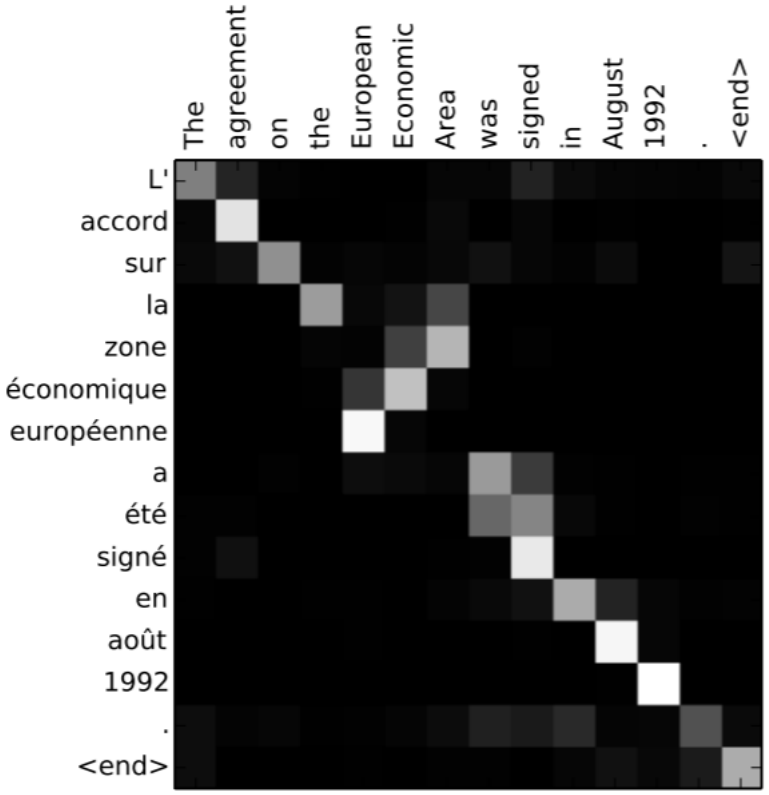


Early Exit

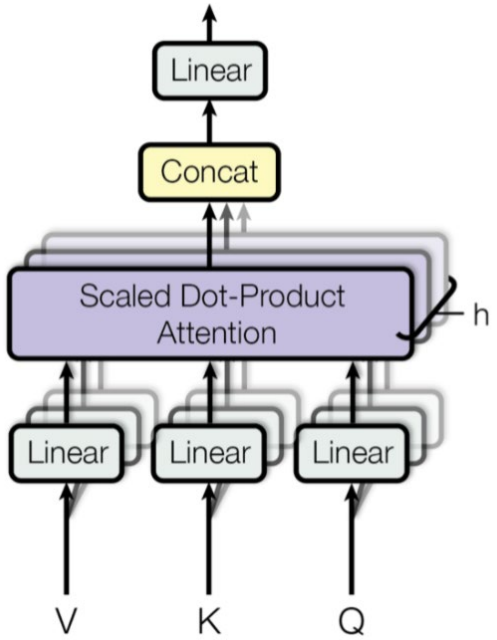
Input: x_n(transformer output),E_T(threshold of entropy)
Output: z_N(output of last transformer) or unsolvable
For n = 1…N **do**
    $z_n = f_{exit}(x_n)$
    $y_n = softmax(z_n)$
    $e_n = entropy(y_n)$
    if e < E_T **then**
        return $argmax(y_n)$
    End if
End for
If the question is solvable
    return z_N
Else
    return unsolvable question to the server



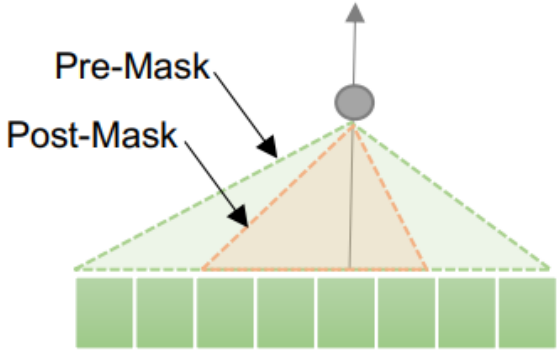
Adaptive Attention



Attention Mechanism



Multi-Head Attention



Adaptive Attention Span

Fp16 Quantization

Floating Point defined by IEEE 754

Numerical form:

$$V_{10} = (-1)^{S} * M * 2^{E}$$

Sign bit ***s*** determines whether number is negative or positive

Significand (mantissa) ***M*** usually a fractional value in range [1.0,2.0)

Exponent ***E*** weights value by a (-/+) power of two

Analogous to scientific notation

Addition

$$(\pm s1 \times b^{e1}) + (\pm s2 \times b^{e2}) = (\pm s1 \times b^{e1}) + (\pm s2 / b^{e1-e2}) \times b^{e1}$$
$$= (\pm s1 \pm s2 / b^{e1-e2}) \times b^{e1} = \pm s \times b^{e}$$

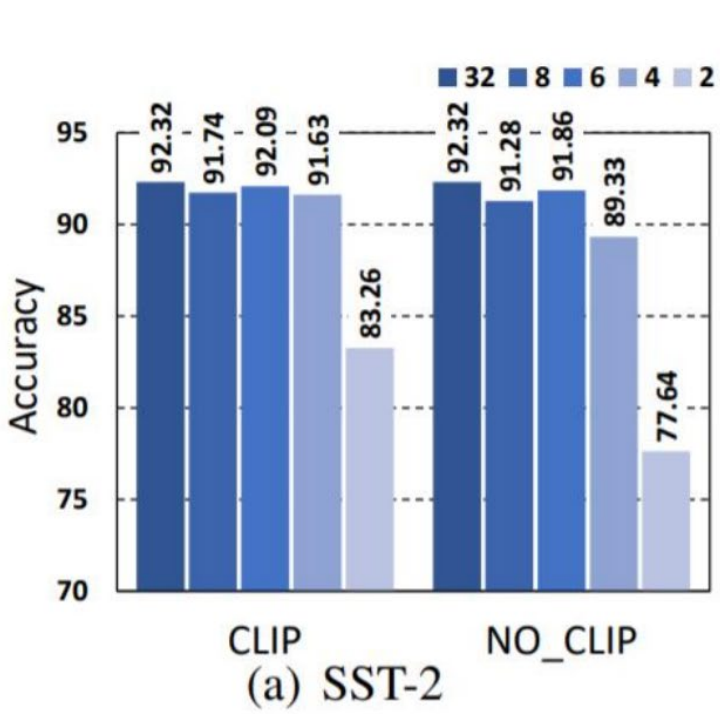
Multiplication

$$(\pm s1 \times b^{e1}) \times (\pm s2 \times b^{e2}) = (\pm s1 \times s2) \times b^{e1+e2}$$

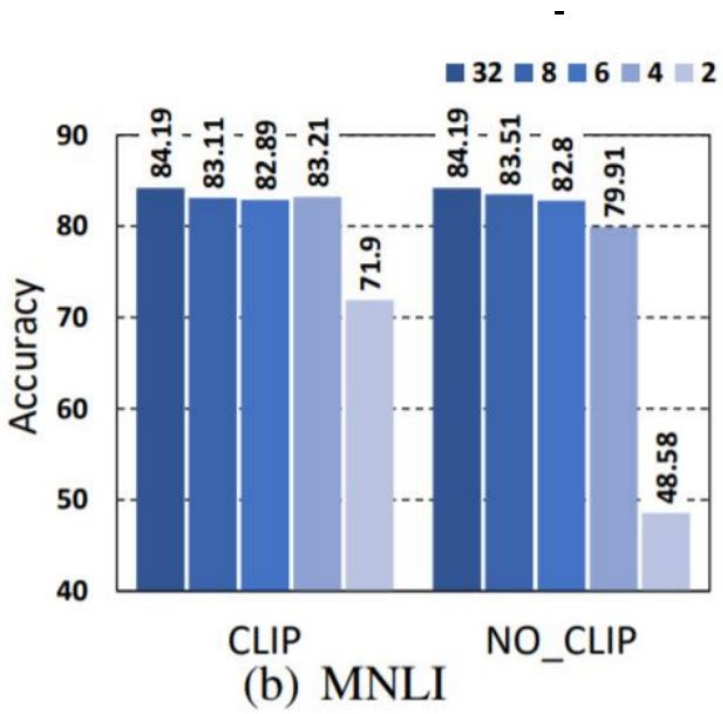
Division

$$(\pm s1 \times b^{e1}) / (\pm s2 \times b^{e2}) = (\pm s1 / s2) \times b^{e1-e2}$$

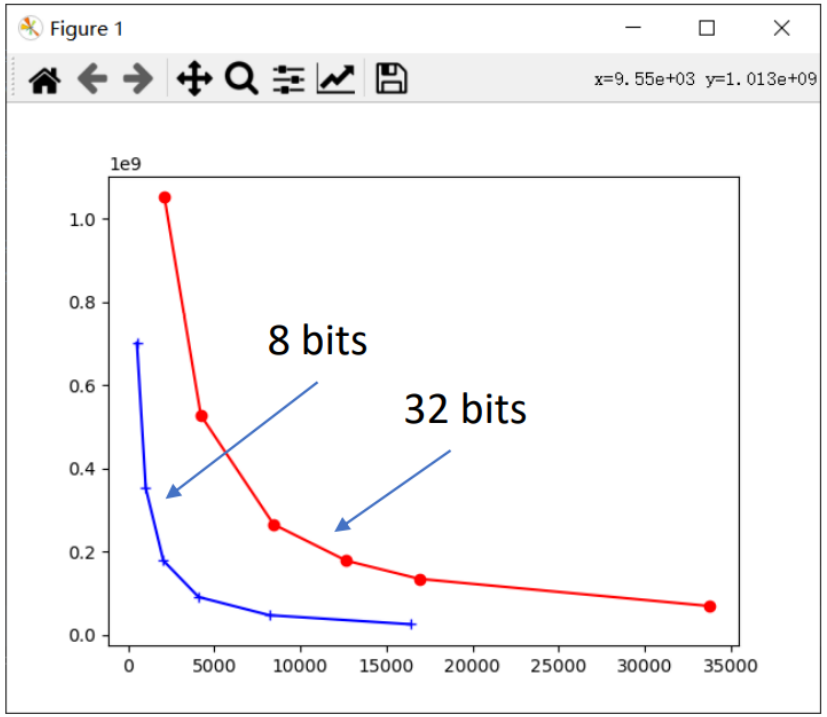
Fp16 Quantization



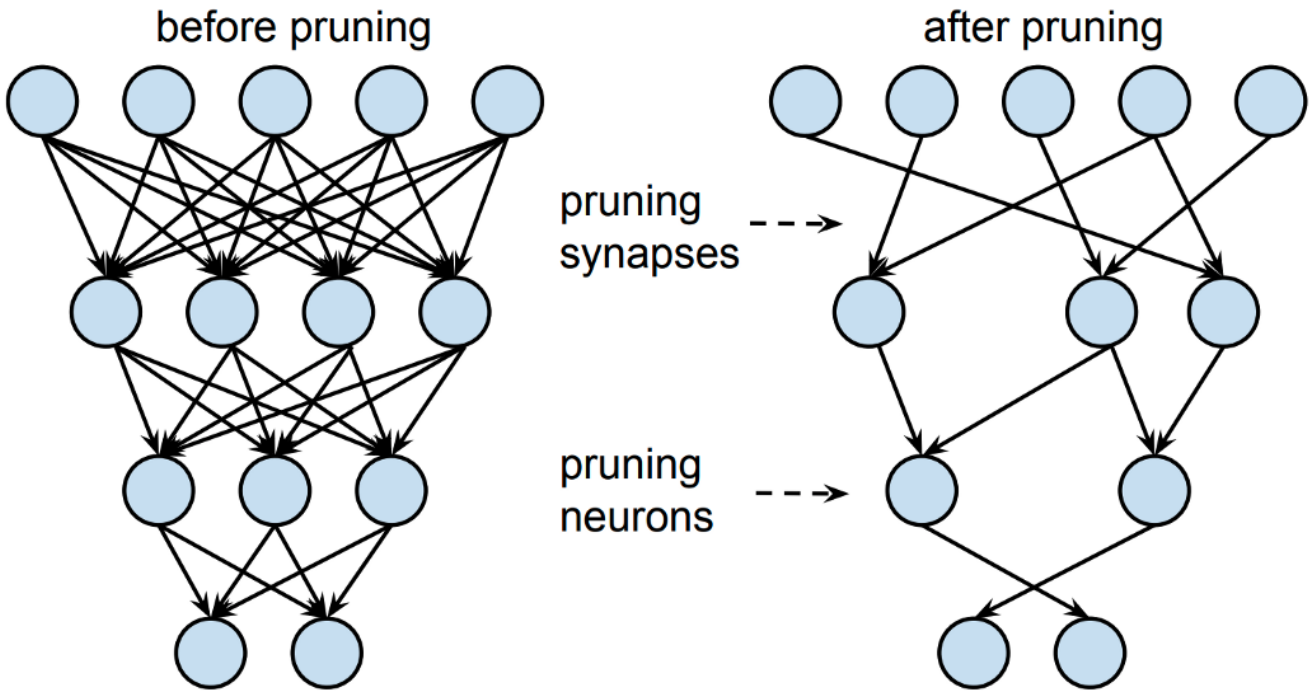
(a) SST-2



(b) MNLI



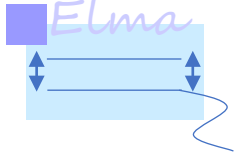
Network Pruning



Method:

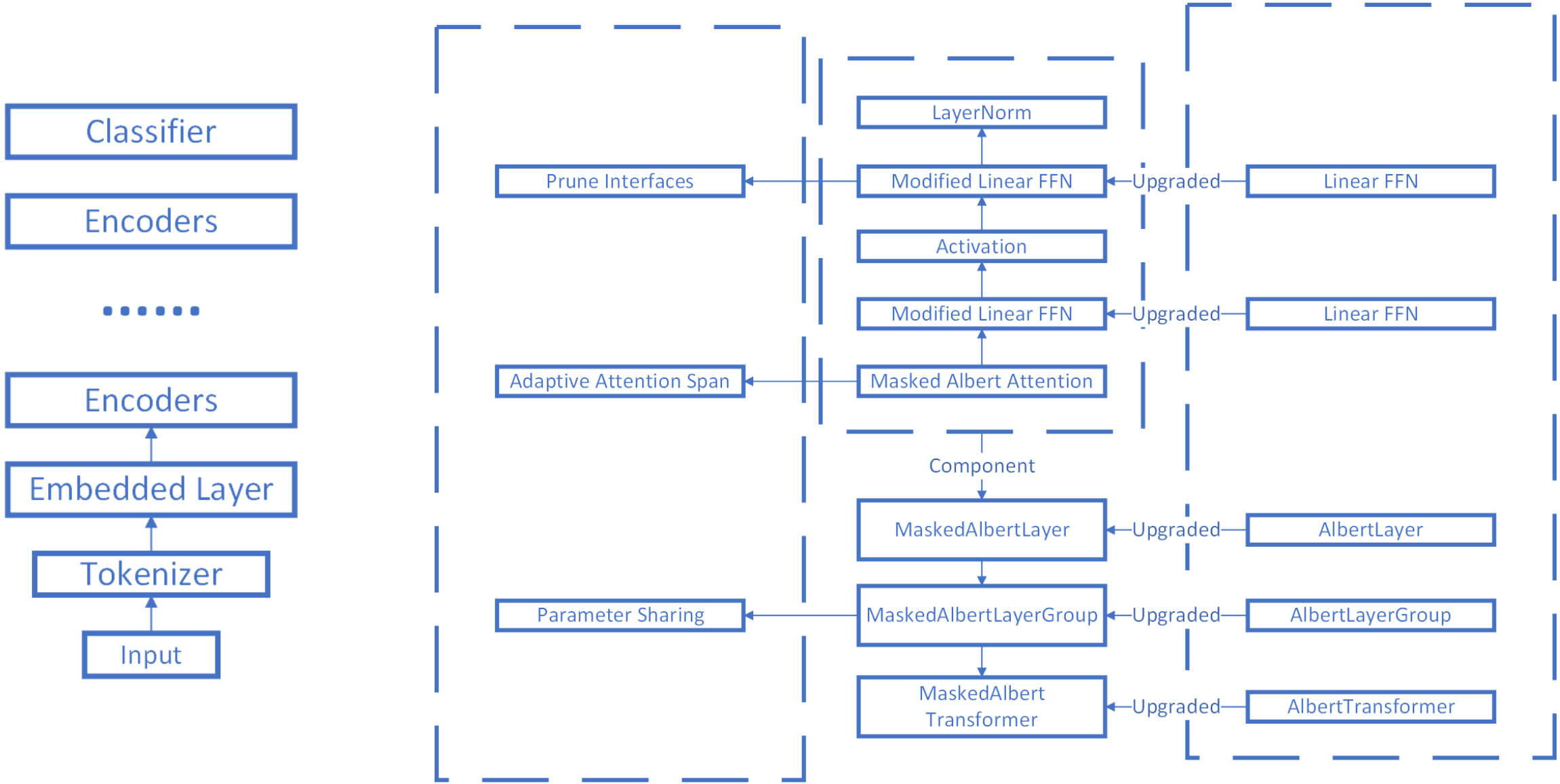
Movement Pruning

Magnitude Pruning

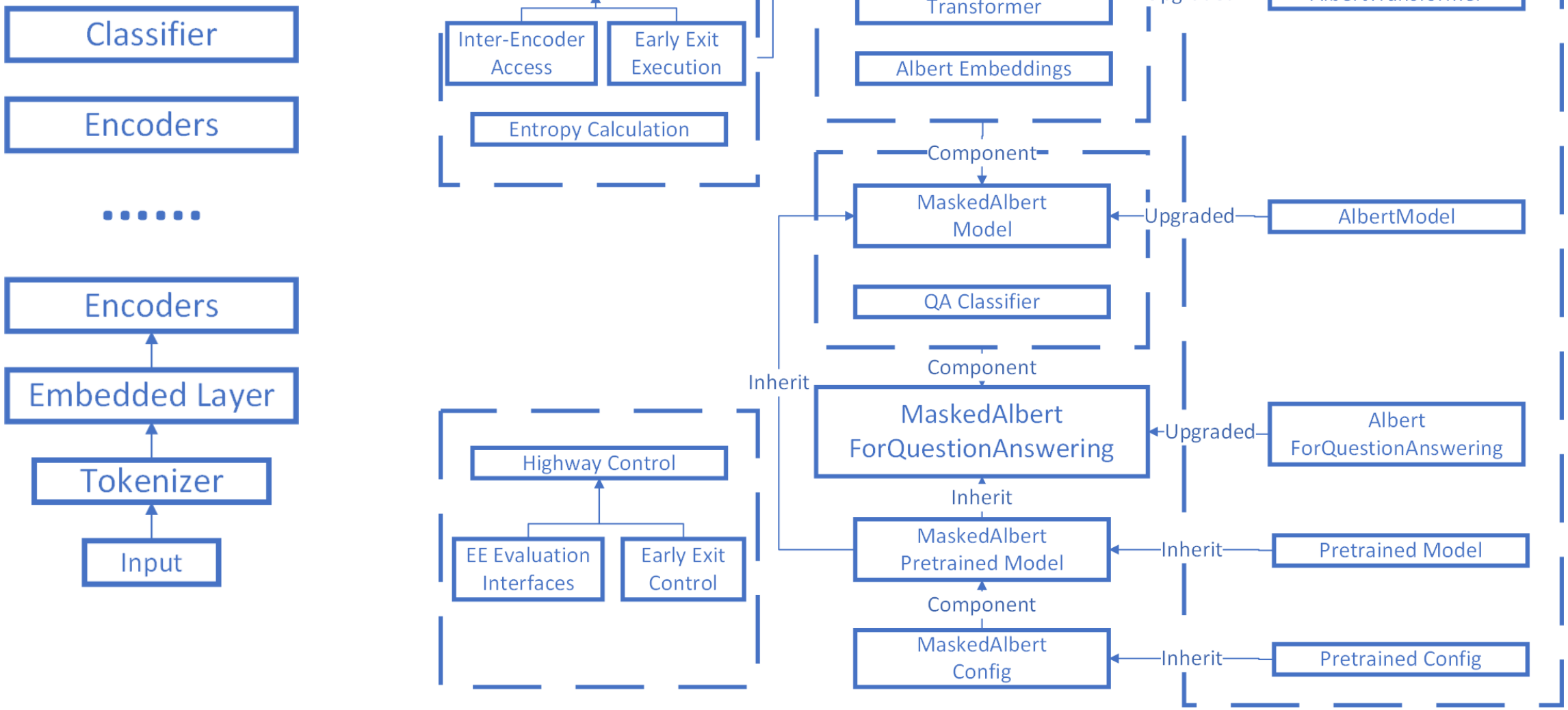


5. Implementation & Validation

Implementation



Implementation



Implementation

FP16

```
if fp16:
        with amp.scale_loss(loss, optimizer) as scaled_loss:
                scaled_loss.backward()
else:
        loss.backward()
```

Magnitude Prune

```
mask = MagnitudeBinarizer.apply(inputs=tensor, threshold=threshold)
pruned_model[name] = tensor * mask
```

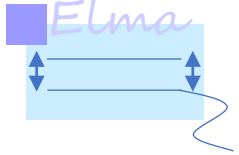
Validation

JOINT INSTITUTE 交大密西根学院

$$\text{exact} = (\text{norm}(\text{orig answer}) == \text{norm}(\text{pred answer}))$$
$$\text{f1} = (2 * \text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

63 Aug 03, 2020	AMBERT-H (single model) *ByteDance*	76.710	79.659
63 Aug 03, 2020	AMBERT-S (single model) *ByteDance*	76.563	79.776
64 Jan 05, 2019	synss (single model) *bert_finetune*	76.055	79.329
65 May 21, 2021	mgrc *single model*	75.344	78.381
65 Apr 05, 2021	BERT-Base-L (single model) *Anonymous*	75.457	78.232
66 Dec 18, 2018	ARSG-BERT (single model) *TRINITI RESEARCH LABS, Active.ai* https://active.ai	74.746	78.227
66 Aug 29, 2020	BERT-Base-V (single model) *Anonymous*	75.073	77.805
66 Nov 05, 2018	MIR-MRC(F-Net) (single model) *Kangwon National University, Natural Language Processing Lab. & ForceWin, KP Lab.*	74.791	77.988
67 Aug 06, 2020	BERT-Base-DT (single model) *Anonymous*	74.769	77.706
68 Dec 03, 2020	BERT-Base-V2 *single model*	74.656	77.404

Metrics	Values
HasAns_exact	65.72199730094466
HasAns_f1	71.02531166019085
HasAns_total	5928
NoAns_exact	84.44070647603027
NoAns_f1	84.44070647603027
NoAns_total	5945
exact	75.09475280047165
f1	77.74261328405724
Total	11873

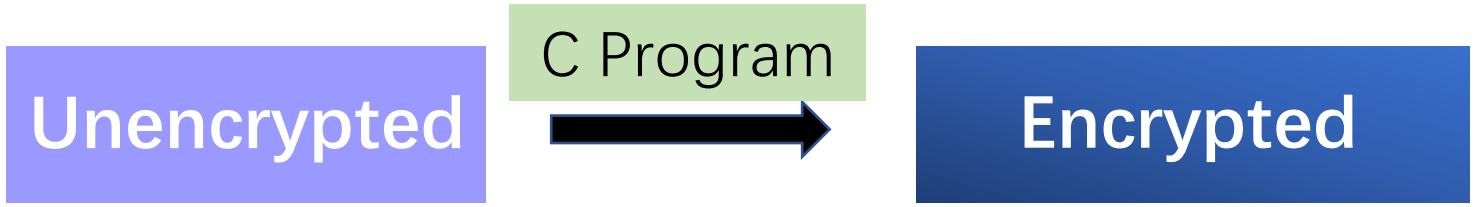


6. Discussion & Conclusion

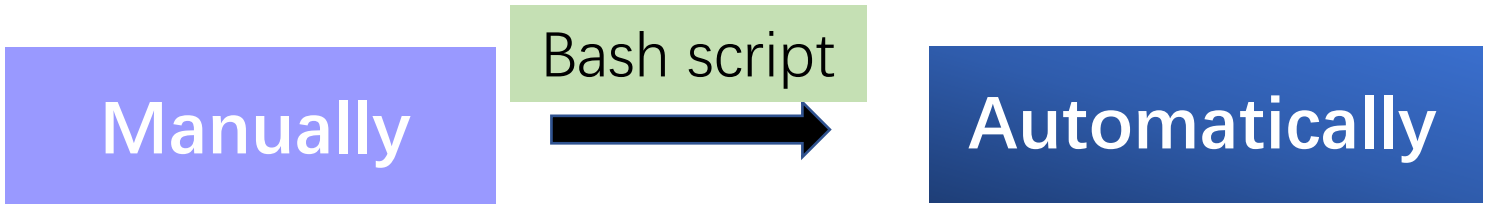
Discussion

Functionality

- Encryption: Localized encryption



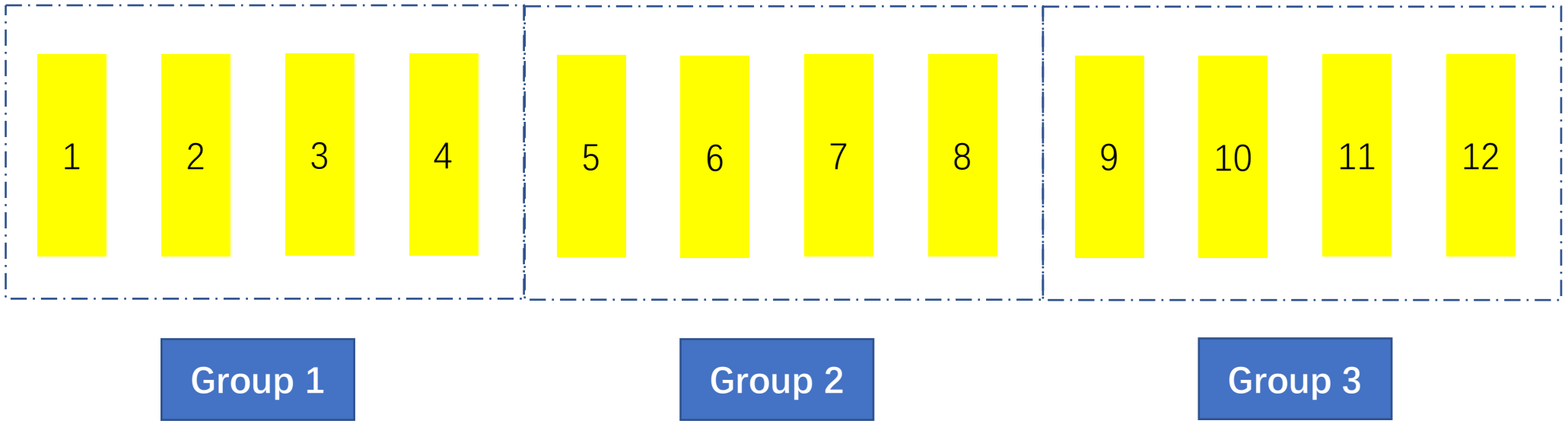
- Synchronization: Network communication automation



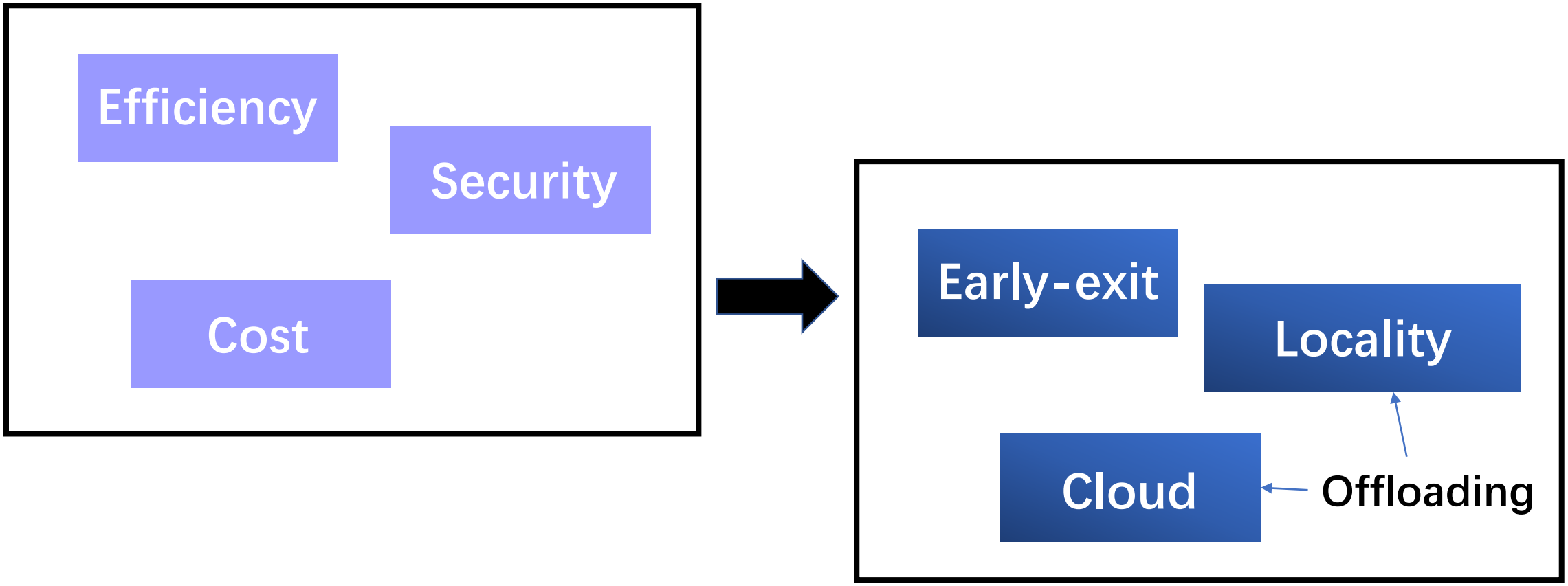
Discussion

Optimization

- Distillation rate
- Choice of entropy
- Introduce a true error dataset
- Parameter sharing within each of the layer groups



Conclusion


Efficiency
Security
Cost
Early-exit
Locality
Cloud
Offloading

Thanks!

ELMA: Early-Exit Offloading for Embedded Question Answering Applications

Group:	3
Instructor:	Prof. An Zou
Sponsor:	UM-SJTU Joint Institute
Group Member:	Yihua Liu, Shuocheng Chen, Yiming Ju