UM–SJTU Joint Institute

Advanced Embedded System

(ECE4730J)

Mid-Check Report

ELMA: Encrypted Offloading for Embedded NLP Applications

Group 3

Name: Yihua Liu ID: 518021910998
Name: Yiming Ju ID: 518370910059
Name: Shuocheng Chen ID: 517021911139

Date: November 8, 2021

Contents

1 Overview

The project can be divided into three parts, including the full model on the remote server, the lite model on local embedded system and the link between the server and the embedded device.

2 Methodology

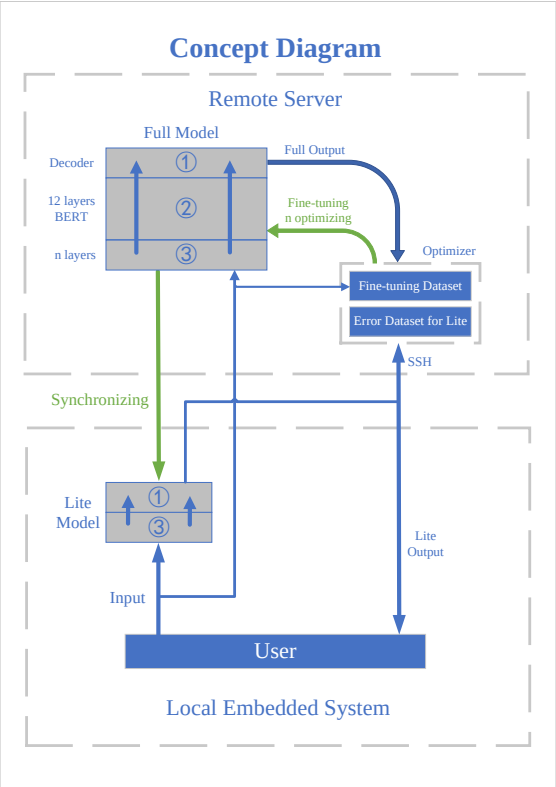

Figure 1. The diagram of the concept design.

2.1 Full ALBERT Model

ALBERT[2] is a model based on BERT with better performance. A pre-trained model of ALBERT[2] will be used in our project and it will be fine-tuned on the dataset SQuAD[1]. It is expected to be 12 layers in the model.

2.2 Lite EdgeBERT Model

Since the size of 12 layers is still too large for embedded systems, a lite model with less layers will be preferred. Thus an abridged ALBERT model with about 5 layers will be used in our embedded system. Its advantages are to prevent over-fitting and reduce computation. Between each two layers, there will be a classification layer so as to check whether the output is accurate enough to satisfy the need. In this way, complex sentences will be with larger model while simple

sentences will be with smaller one. Depending on the structural and contextual complexity, a given input can exit earlier or later.
To evaluate the standard quantitatively, a concept named entropy is defined below [3]

$$H(x) = -\sum p(x) \log p(x) = \ln(\sum_{k=1}^{n} e^{x_k}) - \frac{\sum_{k=1}^{n} x_k e^{x_k}}{\sum_{k=1}^{n} x_k} \tag{1}$$

Then **Algorithm 1** indicates how to realize early exit in lite model.

Algorithm 1 Early Exit

```
for layer i from 1 to n do
  if H(z_i) < E_T then
    return z_i
  end if
end for
if the question is solved then
  return z_n
else
  return unsolvable question to the remote server
end if
```

where z_i is the output of each transformer layer and E_T is the threshold to exit. Once the value of entropy is smaller than the threshold, the model will exit early and output the result.

2.3 Link between two models

The link between the full model and the lite model is bidirectional. At the end of the lite model, the output will be checked whether the question is answerable locally. Otherwise it will be uploaded to the remote server as the data in error data set for lite model, which is to make the model more corresponding to the local needs. In turn, the local lite model will be optimized through synchronizing the model, which is trained by the unsolvable questions from lite model.

3 Implementation and Validation

Our system is similar to an edge computing system, while only one edge device is applied as an embedded platform. validation environment contains two parts: the embedded platform and the cloud platform.

3.1 Validation Environment of Embedded Platform

NVIDIA Jetson TX2 serves as the embedded platform. Jetson TX2 is an embedded computing device for artificial intelligence.

3.1.1 Hardware Specifications

The embedded platform uses the following hardware specifications:

1. GPU: 256-core NVIDIA Pascal GPU architecture with 256 NVIDIA CUDA cores
2. CPU: Dual-core 64-bit NVIDIA Denver 2 CPU and quad-core ARM Cortex-A57 MPCore

3. Memory: 8 GiB 128-bit LPDDR4
4. Storage: 32 GiB eMMC 5.1 and 512 GiB SATA 3.0

3.1.2 Development Environment

NVIDIA provides a software development kit for Jetson platforms called JetPack SDK. The embedded platform is based on JetPack 4.6, which includes the following components:

1. Ubuntu 18.04.6 LTS (bionic) with Linux kernel 4.9.253-tegra
2. CUDA Toolkit 10.2 for L4T
3. `cuDNN` 8.2.1
4. `TensorRT` 8.0.1-1+cuda10.2 (8.0.1.6)

3.1.3 Package Dependencies

Since TensorFlow does not provide builds for AArch64 (ARMv8) architecture with GPU support, TensorFlow is manually compiled, built, and installed from GitHub source of branch `v2.7`.

1. `gcc (Ubuntu/Linaro 7.5.0-3ubuntu1 18.04) 7.5.0`
2. Python 3.6.9
3. Anaconda 4.10.1 (Python 3.8.8, GCC 10.2.0)
4. Bazel 3.7.2 (from `npm` package `@bazel/bazelisk` 1.10.1)

3.2 Validation Environment of Cloud Platform

The cloud platform is based on VMware vSphere Bitfusion servers. Virtual desktops (clients) are provided individually by JI IT Office and the access to GPU resources is guaranteed by Bitfusion utilities in the Virtual Desktops.

3.2.1 Hardware Specifications

The embedded platform uses the following hardware specifications:

1. GPU: 4*2 Tesla V100 32GiB GPU with NVIDIA Volta architecture and 640 NVIDIA tensor cores on Bitfusion Cloud Server
2. CPU: Quad-core Intel Xeon Gold 6226 @ 2.70GHz
3. Memory: 8GiB 32-bit VMware Virtual Memory
4. Storage: 200GiB VMware Virtual Disk

3.2.2 Development Environment

VMware vSphere Bitfusion client provides the following components:

1. Ubuntu 20.04.3 LTS (focal) with Linux kernel 5.11.0-38-generic

To run dated packages, e.g, the original release of Albert, two sets of environments are installed to satisfy different requirements.

1. Original Release of Google Albert
   (a) CUDA Toolkit 10.0
   (b) CuDNN 7.4.0
2. Transformer Repository from Hugging Faces
   (a) CUDA Toolkit 11.2
   (b) CuDNN 8.1.0

3.2.3 Package Dependencies

Common NLP requirements, e.g., numpy and sentencepiece, are omitted.

1. Original Release of Google Albert
   (a) Python 3.6.15
   (b) Tensorflowgpu 1.15.2
   (c) CuPy 9.5.0 for CUDA 10.0
2. Transformer Repository from Hugging Faces
   (a) Python 3.8.10
   (b) Tensorflow 2.7.0
   (c) CuPy 9.5.0 for CUDA 11.2

4 Problems and Solutions

4.1 Embedded Platform

4.1.1 Compability and Dependency

1. Problem: Bazel does not provide binaries for AArch64/ARMv8.
   Solution:
   For TensorFlow 1.15.2 requires Bazel 0.26.1, so we have to manually compile and build Bazel by Scratch

```
sudo apt-get install build-essential openjdk-11-jdk python zip unzip
cd bazel-0.26.1-dist
find -name '*.sh' -exec chmod a+x '{}' \;
sudo -E env EXTRA_BAZEL_ARGS="--host_javabase=@local_jdk//:jdk" bash
↪  ./compile.sh
```

   We have to manually change the permission settings of all the shell scripts because they are not executable by default after extracting from the distribution zip file.
   For TensorFlow 2.7, we can directly install Bazel 3.7.2 from bazelisk 1.10.1

```
npm install -g @bazel/bazelisk
```

2. Problem: TensorFlow does not provide binaries for AArch64/ARMv8.
   Solution: manually compile and build TensorFlow.
   Take TensorFlow 2.7 as example,

```
git clone https://github.com/tensorflow/tensorflow.git
cd tensorflow
git checkout r2.7
sudo .configure
sudo -E bazel --output_user_root=/ssddata/bazel build --config=dbg
↪  --config=cuda //tensorflow/tools/pip_package:build_pip_package
```

3. Problem: TensorFlow 1.15.2 pre-built configuration configuration asks for Open MPI, ComputeCPP, and `clang` support.
   Solution:

```
tar -xf ComputeCpp-CE-2.7.0-aarch64-linux-gnu.tar
cd ComputeCpp-CE-2.7.0-aarch64-linux-gnu
sudo mv ComputeCpp-CE-2.7.0-aarch64-linux-gnu /usr/local
sudo mv /usr/local/ComputeCpp-CE-2.7.0-aarch64-linux-gnu
↪  /usr/local/computecpp
sudo sed -i '\$a deb http://apt.llvm.org/bionic/ llvm-toolchain-bionic
↪  main\ndeb-src http://apt.llvm.org/bionic/ llvm-toolchain-bionic
↪  main' /etc/apt/sources.list
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
↪  15CF4D18AF4F7421
sudo apt update
sudo apt install clang-format clang-tidy clang-tools clang clangd
↪  libc++-dev libc++1 libc++abi-dev libc++abi1 libclang-dev libclang1
↪  liblldb-dev libllvm-ocaml-dev libomp-dev libomp5 lld lldb llvm-dev
↪  llvm-runtime llvm python-clang
gunzip -c openmpi-4.1.1.tar.gz | tar xf -
cd openmpi-4.1.1
sudo ./configure --prefix=/usr/local
sudo make all install
```

4. Problem: TensorFlow 1.15.2 pre-build configuration __main__.UserInputError: SYCL / CUDA / ROCm are mututally exclusive. At most 1 GPU platform can be configured.
   Solution: Do not choose ROCm, OpenCL SYCL, and CUDA simultaneously. Only CUDA is needed.
5. Problem: TensorFlow pre-built configuration requires specifying CUDA compute capabilities.
   Solution: According to CUDA GPUs — NVIDIA Developer, the CUDA compute capability of Jetson TX2 is 6.2.
6. Problem: TensorFlow pre-built configuration ERROR: Config value download_clang_use_lld is not defined in any .rc file.

Solution: use `gcc` instead of `clang`. All of the tested build configuration for Linux provided by TensorFlow use `gcc`.

7. Problem: TensorFlow 1.15.2 building ValueError: dictionary update sequence element #9 has length 1; 2 is required.
   Solution: At Line 355 of third_part/gpus/find_cuda_config.py, specify `cudnn_version` to be 8.

8. Problem: TensorFlow 1.15.2 building Illegal ambiguous match on configurable attribute "deps" in //tensorflow/tools/pip_package:included_headers_gather.
   Solution: Again use `gcc` instead of `clang`.

9. Problem: TensorFlow 1.15.2 building error: overriding 'virtual void ... noexcept'.
   Solution:
   Change the prototype of function `log` at Line 26 of tensorflow/compiler/tf2tensorrt/utils/trt_logger.h from

```
void Logger::log(Severity severity, const char* msg)
```

   to

```
void log(Severity severity, nvinfer1::AsciiChar const* msg) noexcept
```

10. Problem:
    TensorFlow 1.15.2 building error: declaration of 'void* createInferRuntime_INTERNAL(void*, int)' has a different exception specifier.
    Solution: Change the prototype of function `getLogger` at Line 19 and Line 26 of tensorflow/compiler/tf2tensorrt/stub/NvInfer_5_0.inc from

```
nvinfer1::ILogger* getLogger()
```

    to

```
nvinfer1::ILogger* getLogger() noexcept
```

    and change the prototype of function `getPluginRegistry` at Line 33 and Line 40 of tensorflow/compiler/tf2tensorrt/stub/NvInfer_5_0.inc from

```
nvinfer1::IPluginRegistry* getPluginRegistry()
```

    to

```
nvinfer1::IPluginRegistry* getPluginRegistry() noexcept
```

11. Problem: TensorFlow 1.15.2 building cudnnGetRNNDescriptor error.
    Solution: Change `cudnnGetRNNDescriptor` into `cudnnGetRNNDescriptor_v6` or change

```
RETURN_IF_CUDNN_ERROR(cudnnGetRNNDescriptor(
    /*handle=*/cudnn.handle(), /*rnnDesc=*/rnn_desc,
    /*hiddenSize=*/&hidden_size_v,
    /*numLayers=*/&num_layers_v,
    /*dropoutDesc=*/&dropout_desc,
    /*inputMode=*/&input_mode,
    /*direction=*/&direction,
    /*mode=*/&mode,
    /*algo=*/&algo,
    /*dataType=*/&data_type));
```

    into

```
cudnnDataType_t data_type, mathPrec;
cudnnRNNMode_t cellMode;
cudnnRNNBiasMode_t biasMode;
cudnnMathType_t mathType;
int32_t inputSize, projSize;
uint32_t auxFlags;
RETURN_IF_CUDNN_ERROR(cudnnGetRNNDescriptor_v8(
    rnn_desc,
    &algo,
    &cellMode,
    &biasMode,
    &direction,
    &input_mode,
    &data_type,
    &mathPrec,
    &mathType,
    &inputSize,
    &hidden_size_v,
    &projSize,
    &num_layers_v,
    &dropout_desc,
    &auxFlags));
```

12. Problem: TensorFlow 1.15.2 building error: 'cudnnConvolutionFwdPreference_t' was not declared in this scope.
    Solution: Migrate to TensorFlow 2.7 because this incompatibility is un-resolvable. According to Table 7. API functions and data types that were removed of Nvidia cuDNN Documentation, `cudnnConvolutionFwdPreference_t` is removed since cuDNN version 8.

13. Problem: the recommended GCC version is 7.3.1 but this version is not installed.
    Solution:

```
sudo mkdir /etc/apt/sources.list.d
sudo add-apt-repository ppa:jonathonf/gcc
sudo apt-get update
sudo apt install gcc-7
```

If the directory /etc/apt/sources.list.d does not exist, it will prompt "no such file or directory" error when adding APT repository.

14. Problem: TensorFlow 2.7 building crosstool_wrapper_driver_is_not_gcc failed.
    Solution: remove option `--config=v1` of `bazel build`.

4.1.2 Network

1. Problem: we need to connect between Jetson TX2 and personal computers.
   Solution: Use SSH connections.
2. Problem: SSH connections are only available in local networks.
   Solution: Applying a public IP.
3. Problem: `rsync` is only supported file transportation between Linux hosts.
   Solution: Use SFTP/SCP connections.
4. Problem: could not download some packages or connect to some websites.
   Solution: use network tools to configure Internet connections.
5. Problem: `sudo` commands does not inherit user environment paths and variables to connect to the Internet.
   Solution: Add option `-E` to make `sudo` to inherit user environment paths and variables.

4.1.3 Disk Space

1. Problem: when building TensorFlow runs out of disk space.
   Solution: add an external SSD and mount it at /ssddata. When running `bazel`, add option `--output_user_root=/ssddata/bazel`.
2. Problem: after disk space runs out, booting the device will fail with error

```
cp: not writing through dangling symlink 'etc/resolv.conf'
cgroup: cgroup2: unknown option "nsdelegate"
tegra-i2c 3180000.i2c: no acknowledge from address 0x36
regmap_util_write_table_8:regmap_util_write_table:-121[    2.855855]
↪  ov5693 2-0036: could not read otp bank
ov5693 2-0036: Error -121 reading otp data
ov5693 2-0036: board setup failed
ov5693 probe of 2-0036 failed with error -121
using random self ethernet address
using random host ethernet address
using random self ethernet address
using random host ethernet address
```

```
CPU1: shutdown
CPU2: shutdown
```

Solution: since USB connection has a fixed IP, so use this IP to establish SSH connection between VMware Ubuntu and Jetson TX2. After the connection set up, not only SSH remote operations can be done, Jetson TX2 itself can also show a command line interface. The interface may also be called by Alt+F2 or Ctrl+Alt+F2. Through them, remove more than 1 GiB space. Before we have set up a mounted disk /ssddata, so executing

```
sudo mv ~/.cache/bazel /ssddata
```

It can free more than 8 GiB disk space.

4.2 Cloud Platform

4.2.1 Memory

1. Problem: Memory is too small for CPU training
   Solution: Add 10 GiB virtual memory in swap
2. Problem: Tensor cannot be allocated to GPU because of an exhaustion of resourcess Solution: Half the number of batch from 48 to 24.

5 Current Progress

5.1 TensorFlow Deployment in Local

After we find that TensorFlow 1.15.2 is not compatbile with cuDNN 8 and TensorRT 8, we are currently building TensorFlow 2.7 by Bazel 3.7.2 on Jetson TX2.

Alternatively, we have just found Nvidia has provided wheels for AArch64, Nvidia container 21.7 - 21.9, Python 3.6, JetPack 4.6 TensorFlow 1.15.5 and 2.6.0. If manually compiled and built version of TensorFlow succeeds, we will adopt it because it may gives better performance, but if not, we will adopt the Nvidia version. Besides, it shows us possibilities to run ALBERT on Jetson TX2. ALBERT requires TensorFlow version 1.15.2, which is close to version 1.15.5.

5.2 Fine-tuning ALBERT in Remote

5.2.1 Task Selection

The pre-trained model is from https://tfhub.dev/google/albert_large/3

The dataset SQuAD is a classic and frequently-used dataset from https://rajpurkar.github.io/SQuAD-explorer/Group

```
python -m albert.run_squad_v2
↪  --albert_config_file=albert/material/albert_large/albert_config.json
↪  --output_dir=albert/outputs/ --train_file=albert/material/train-v2.0.json
↪  --predict_file=albert/material/dev-v2.0.json
↪  --vocab_file=albert/material/albert_large/30k-clean.vocab
↪  --train_feature_file=train_feature_file.tf
↪  --predict_feature_file=predict_feature_file.tf
↪  --predict_feature_left_file=predict_left_feature_file.tf
↪  --init_checkpoint=albert/material/albert_large/model.ckpt-best.index
↪  --spm_model_file=albert/material/albert_large/30k-clean.model
↪  --do_lower_case --max_seq_length=384 --doc_stride=128
↪  --max_query_length=64 --do_train --do_predict --train_batch_size=48
↪  --predict_batch_size=8 --learning_rate=5e-5 --num_train_epochs=5.0
↪  --warmup_proportion=.1 --save_checkpoints_steps=5000 --n_best_size=20
↪  --max_answer_length=30
```

5.2.2 Trial Run

Our progress stuck on fine-tuning on SQuAD V2 for long, albeit converting the dataset to tensorflow feature file was completed initially. Many configurational and environmental problems emerged when fine-tuning was attempted and its adaptation to GPU was enforced. Currently, multiple models are being fine-tuned in parallel, which utilizes the GPU resources maximally.

6 Future Plan

6.1 Embedded Platform

- Deploy `EdgeBERT` on Jetson TX2
- Develop speech-to-text application on Jetson TX2

6.2 Cloud Platform

- Synchronize the lite model with the full model
- Update the training set automatically with the unanswerable questions on lite model
- Utilize fully the Transformer repository from Huggingface to establish a customized model improved from original ALBERT, in order to enable its embedded applicability and include EdgeBERT functions.

6.3 System Design

- Configure connections between audio devices and the embedded platform
- Set up a user interface
- Since EdgeBERT is a hardware/software co-design, or more specifically hardware/algorithm co-design, the entropy assessment, DVFS, etc. requires hardware C++ simulation

and HLS. EdgeBERT uses SystemC and a SystemC/C++ library called `matchlib` to synthesize HLS into RTL, but Xilinx provides a commercial solution to this feature.

References

[1] Stanford NLP Group. *Stanford Question Answering Dataset*. Version 2.0. Stanford, California: Stanford University, 2021. URL: https://rajpurkar.github.io/SQuAD-explorer.

[2] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[3] Thierry Tambe et al. *EdgeBERT: Sentence-Level Energy Optimizations for Latency-Aware Multi-Task NLP Inference*. 2021. arXiv: 2011.14203 [cs.AR].