ARL-TN-1068 ● July 2021

Extending OpenNMT's TensorFlow Lite to Include Transformer Models

by Gerardo Cervantes and Stephen LaRocca

ARL-TN-1068 ● July 2021

Extending OpenNMT's TensorFlow Lite to Include Transformer Models

Stephen LaRocca
Computational and Information Sciences Directorate,
DEVCOM Army Research Laboratory

Gerardo Cervantes
Advanced Resource Technologies, Inc.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* July 2021	**2. REPORT TYPE** Technical Note	**3. DATES COVERED (From - To)** May–July 2021

4. TITLE AND SUBTITLE Extending OpenNMT's TensorFlow Lite to Include Transformer Models	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gerardo Cervantes and Stephen LaRocca	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DEVCOM Army Research Laboratory ATTN: FCDD-RLC-IB Adelphi, MD 20783-1138	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-1068
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID IDs: Gerardo Cervantes, 0000-0002-4392-5017; Stephen LaRocca, 0000-0003-3341-5520

14. ABSTRACT
Since its release in 2017 the OpenNMT project has provided open development tools for Neural Machine Translation (NMT) including machine-learning inference with artificial neural-network models on Android platforms. Rapid advances in OpenNMT methods were achieved using TensorFlow since 2018; however, most of these advances were not deployable for use on Android platforms pending completion of the TensorFlow Lite library. The US Army Combat Capabilities Development Command Army Research Laboratory's Shareable Components project team closely tracked progress on TensorFlow Lite and succeeded in implementing a new method for converting OpenNMT models from standard TensorFlow to the Lite variant. Deployable on Android devices, these converted models provide important gains in execution speed while occupying less space. This extension adds more features to OpenNMT-tf, which allows the export of better-performing Transformer models onto Android. Beam search and <unk> replacement have also been added to this extension, which will generally help increase the performance.

15. SUBJECT TERMS
Machine Translation, deep learning, Android, open development tools, TensorFlow Lite

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Stephen LaRocca
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	19	**19b. TELEPHONE NUMBER (Include area code)** (301) 394-3198

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

1. Introduction 1

1.1 Purpose 1

1.2 Distribution 1

1.3 Requirements 1

1.4 Feature List 1

1.5 Exportable Model List 2

2. Exporting Instructions 2

2.1 Installing the Required packages 2

2.2 Training a Model 3

2.3 TensorFlow Lite Exporting 3

2.3.1 Modifiable Parameters 3

2.3.2 Data-Configuration File 3

2.3.3 File 3

3. Running the Model 4

3.1 Description 4

3.2 Running 4

3.3 Android Example 4

3.3.1 Android Dependencies 4

3.3.2 Reading Vocabulary Files 5

3.3.3 Converting Sentences to IDs 5

3.3.4 Converting IDs to a Sentence 6

3.3.5 Loading the Model 6

3.3.6 Running the Model 7

4. Features Added 8

4.1 Transformer Models 8

4.1.1 Description 8

4.1.2 Exporting Options 8

4.1.3 Limitations 8

4.2 Beam Search 8
4.2.1 Description 8
4.2.2 Compatibility 8
4.2.3 Exporting Instructions 8
4.3 <unk> Replacement 9
4.3.1 Description 9
4.3.2 Limitation 9
4.3.3 Exporting Instructions 9
4.4 Quantization 9
4.4.1 Types 9
4.4.2 Choosing a Quantization 9

5. Major Modification Descriptions 10
5.1 Transformer Relative 10
5.2 Beam Search 10
5.3 Dynamic-Range Quantization 10
5.4 <unk> Replacement 10

6. TensorFlow Lite Unit Tests 11
6.1 Test Descriptions 11
6.2 Parameterized 11
6.3 Arguments' Description 12

Distribution List 13

1. Introduction

1.1 Purpose

This is a pull request to OpenNMT-tf that adds useful features for creating TensorFlow Lite models. TensorFlow Lite allows the model to run on mobile environments. TensorFlow Lite lets you quantize the model; this decreases model size and inference time. TensorFlow Lite also allows you to make use of hardware accelerators on the phone for a significant speedup. This code extension adds different types of models that are known to be more accurate and other features to improve performance.

1.2 Distribution

The OpenNMT-tf open-source package can be found on GitHub at https://github.com/OpenNMT/OpenNMT-tf. This code is a pull request to the open-source library to be reviewed by the OpenNMT developers to be added to the repository.

1.3 Requirements

Required tools for exporting an OpenNMT-tf model to TensorFlow Lite are as follows:

- Python 3.5+
- Python pip
- OpenNMT-tf package, https://pypi.org/project/OpenNMT-tf/
- TensorFlow 2.5+

1.4 Feature List

TensorFlow Lite

- Can now export Transformer models.
- Beam search added for Transformer models.
- <unk> replacement has been added with a caveat. If the word that it is being replaced with is not in the source vocabulary, then it will remain as an unknown word (<unk>).

- Dynamic-range quantization parameter added (*tflite_dynamic_range*). In previous versions, using the parameter *tflite* automatically did dynamic-range quantization. See https://www.tensorflow.org/lite/performance/post_training_quantization).
- Added an additional TensorFlow Lite test. Better testing can now be done because you can specify parameters when testing, as well as quantization if applicable.

Export

- If no model to load is found, an error is now shown.

1.5 Exportable Model List

New

- Transformer and Transformer Big
- Transformer Relative and Transformer Big Relative
- Transformer Shared Embeddings and Transformer Big Shared Embeddings

Other

- Luong Attention
- NMTSmallV1, NMTMediumV1, and NMTBigV1

2. Exporting Instructions

2.1 Installing the Required packages

You need to have Python installed on the computer, and the version of Python should be 3.5 or higher. For the following commands, make sure you have Python and pip on the PATH environment variable so you can use those commands from the command line.

To install the required Python packages, run the following commands:

```
pip install –upgrade pip
pip install OpenNMT-tf
pip install TensorFlow
```

If there were no errors, you should have all of the required Python packages.

2.2 Training a Model

Instructions on training a model can be found on this page: https://opennmt.net/OpenNMT-tf/quickstart.html.

2.3 TensorFlow Lite Exporting

Instructions for advanced users on exporting can be found in file *docs/serving.md* of the code.

To export a model to TensorFlow Lite, run the following command:

onmt-main --model_type Transformer --config data.yml export --export_dir ~/output --export_format tflite

The conversion will take 1 to 2 minutes to complete.

2.3.1 Modifiable Parameters

You can adjust the command based on your needs:

--model_type should be the same command you used to train your model.

--config should be the path to the data-configuration file.

--export_dir is the directory where the TensorFlow Lite file will be created.

--export_format should be set to *tflite* unless you want to apply quantization to the model. Possible TensorFlow Lite quantization options are {*tflite_float16*, *tflite_dynamic_range*}.

2.3.2 Data-Configuration File

In the data-configuration file, you will want to make sure the model directory, source_vocabulary, and target_vocabulary paths are correctly set. A full list of parameters you can modify can be found here: https://opennmt.net/OpenNMT-tf/configuration.html.

2.3.3 File

If the conversion is successful, there will be an *opennmt.tflite* model file in the export directory.

3. Running the Model

Whenever you are running the model, you will have to use the same vocabulary files you used when training the model. The location of these vocabulary files is specified in the data-configuration file.

3.1 Description

The vocabulary files consist of a list of words of the language separated by new line characters. Each word in the language is given a unique number, referred to as the ID of the word. Words not in the vocabulary are shown as <unk> and the ID is equivalent to the size of the vocabulary.

3.2 Running

The model requires an array of integers; each integer is an ID from the vocabulary file. To convert a sentence to IDs, you should search each word in the sentence into the ID and put them in an array. The returned translations from the model are integer IDs; you can turn them into a sentence using the other vocabulary file.

3.3 Android Example

Provided in this document is Android Java code for running the models.

3.3.1 Android Dependencies

Adding these dependencies in the Android Studio Gradle file (build.gradle) will allow TensorFlow Lite models to be run on Android. The following lines provide Version 2.5 for TensorFlow; you should use the same version of TensorFlow that you used for converting the model.

```
implementation                    'org.tensorflow:tensorflow-lite:2.5.0'
implementation      'org.tensorflow:tensorflow-lite-select-tf-ops:2.5.0'
implementation 'org.tensorflow:tensorflow-lite-support:2.5.0'
```

Optionally, you can add a Google Guava dependency so that you can use their implementation of a HashBiMap, which is used in the example code. A HashBiMap is a bidirectional HashMap that is useful for going from a word to an ID and from an ID to a word.

```
implementation 'org.google.guava:guava:30.1-jre'
```

3.3.2 Reading Vocabulary Files

The following Android Java code snippet produces a HashBiMap that maps a word to the unique ID given in the file:

```
    public HashBiMap<String, Integer> readVocab(InputStream file){
        try {
            BufferedReader brFile = new BufferedReader(new InputStreamReader(file));
            HashBiMap<String, Integer> vocab = HashBiMap.create();
            String wordRead = brFile.readLine();
            int index = 0;
            while(wordRead != null){
                vocab.put(wordRead, index);
                index += 1;
                wordRead = brFile.readLine();
            }
            return vocab;
        }
        catch(IOException e) {
            return null;
        }
    }
```

You will have to do this twice for both vocabulary files, and the HashBiMap will give you a straightforward way to translate between words and IDs.

3.3.3 Converting Sentences to IDs

The following Android Java code converts a sentence into IDs. This will be used to get the IDs for the sentence you want to translate:

```
    private int[] textToIds(String text, HashBiMap<String, Integer> vocab){
        String[] words = text.split(" ");
        ArrayList<Integer> idsList = new ArrayList<>();
        //Unknown ID is the same as vocabulary size
        int unknownId = srcVocab.size();
        for(String word : words){
            Integer id = vocab.get(word);
            //Use Unknown ID if ID retrieved was null
            id = id == null ? unknownId : id;
            idsList.add(id);
        }
        //Turns Integer[] to int[]
        return
idsList.stream().filter(Objects::nonNull).mapToInt(i            ->
i).toArray();
    }
```

3.3.4 Converting IDs to a Sentence

The following Android Java code converts IDs to a sentence. This will be used to create the sentence after running the translation:

```
    private String idsToText(int[] ids, HashBiMap<Integer, String>
inverseVocab){
        StringBuilder sentence = new StringBuilder();
        for(int id : ids){
            String word = inverseVocab.get(id);
            //Word isn't in the vocabulary file
            if(word == null){
                word = "<unk>";
            }
            //Don't include blank words or end sentence tokens
            if("<blank>".equals(word) || "</s>".equals(word)){
                continue;
            }
            sentence.append(word).append(" ");
        }
        return sentence.toString();
 }
```

3.3.5 Loading the Model

The following code is used to load the model and assumes you have stored the saved model to the assets folder. The code will look at the assets folder to find the model you specified. The *modelPath* variable should be set to the location of the model. The *NUM_LITE_THREADS* should be set to an integer that tells it how many threads to use when running the model:

```
        AssetManager                 assetManager                  =
this.context.getResources().getAssets();
    ByteBuffer buffer = loadModelFile(assetManager, modelPath);
        if (buffer == null){
            Log.e("nmt-tf", "Could not load model");
            return;
        }
        Interpreter.Options opt = new Interpreter.Options();
        opt.setNumThreads(NUM_LITE_THREADS);
        tflite = new Interpreter(buffer, opt);
```

3.3.6 Running the Model

The following code will run the model with the sentence "*Hello World*". The *tflite* variable is the Interpreter defined in the loading of the model code example:

```
        int[] input_ids = sentenceToIds("Hello World");
        int[] output_ids = new int[250];
        tflite.run(input_ids, output_ids);
        String translatedSentence = IdsToSentence(output_ids);
        System.out.println("Translated      Sentence:      "      +
translatedSentence);
```

The previous code snippet will print out the translated sentence. The variable *output_ids* is an initialized array that contains the results of the model after you run the model. The variable should be the same size as *tflite_output_size*, which was set when creating the TensorFlow Lite model.

4. Features Added

4.1 Transformer Models

Transformer models can now be exported with TensorFlow Lite.

4.1.1 Description

Transformer models give outstanding performance in natural language processing tasks. These models have been quickly replacing recurrent-neural-network models due to the models being easier to parallelize and their state-of-the-art results.

4.1.2 Exporting Options

These models can be converted with the <unk> replacement option or/and with beam search by editing the data-configuration file.

There are some limitations to be aware of when exporting Transformer models.

4.1.3 Limitations

- Transformer models cannot be quantized.
- Transformer Relative models with Beam Search do not export to TensorFlow Lite.

4.2 Beam Search

4.2.1 Description

Beam Search is a way to improve the results of a model by searching through the output of the model to one that gives a better likelihood.

4.2.2 Compatibility

Transformer models with Beam Search can be TensorFlow Lite exported, with an exception that the Relative Transformer models cannot be converted with Beam Search. Recurrent-neural-network models with beam search do not work due to a TensorFlow Lite exporting error with Luong Attention in the TensorFlow-Addons package.

4.2.3 Exporting Instructions

The *beam_width* parameter in the configuration variable should be set to a value greater than 1 so that you can run with Beam Search enabled.

After the beam width is set, you can run the TensorFlow export command on a Transformer model to get a model with Beam Search.

4.3 <unk> Replacement

4.3.1 Description

The <unk> replacement replaces unknown words that are not found in the vocabulary, and it replaces them with a word in the source sentence with the highest attention.

4.3.2 Limitation

This TensorFlow Lite implementation is limited because the word it is being replaced with must be in the source vocabulary or else it will replace it with an unknown ID.

4.3.3 Exporting Instructions

To run with <unk> replacement the variable *replace_unknown_target* must be set to true in the configuration file.

After the variable is set to true you can run the export to TensorFlow Lite command to get a model that runs with <unk> replacement

4.4 Quantization

Models can be exported with quantization by changing the export format in the export command.

4.4.1 Types

- tflite—No quantization
- tflite_float16—Float 16 quantization
- tflite_dynamic_range—Dynamic-range quantization

4.4.2 Choosing a Quantization

If you are going to run the model with a graphics processing unit, the recommended quantization to choose is float16 since there will be a considerable increase in performance. With float16 quantization you should expect a larger file size when comparing with dynamic-range quantization.

If you are running with a central processing unit, the recommended quantization is Dynamic Range as the models will run faster with this and the file size will be the smallest it can be.

5. Major Modification Descriptions

5.1 Transformer Relative

opennmt/layers/transformer.py in MultiHeadAttention:

Due to the TensorFlow Issue no. 42410, getting word embeddings code had to be modified. This only affects the Transformer Relative models.

5.2 Beam Search

opennmt/utils/decoding.py in BeamSearch:

When running Beam Search with TensorFlow Lite, it preallocates the *parent_ids* variable. It preallocates the variable with the size provided in the *tflite_output_size* variable provided in the configuration file.

opennmt/utils/decoding.py in DecodingStrategy:

Makes a function to choose the decoding strategy when running using TensorFlow Lite. This will provide the tflite_output_size to the Beam Search Decoder so it can do the preallocating it has to do.

models/sequence_to_sequence.py in SequenceToSequence:

There are some modifications to _dynamic_decode function so that TensorFlow Lite exporting can work with Beam Search. In this file, it is modified to only get the best result from beam search instead of giving the best five sentences.

5.3 Dynamic-Range Quantization

opennmt/utils/exporters.py in TFLiteExporter:

Added export format *tflite_dynamic_range*. Added the extra option and now it will not automatically quantize when given the *tflite* format option.

5.4 <unk> Replacement

opennmt/utils/decoding.py in dynamic_decode:

Attention is modified so it works during decode; this is only available in TensorFlow 2.5+ because it uses TensorArrays of multiple dimensions.

models/sequence_to_sequence.py in SequenceToSequence:

When running regularly it runs using the words; when running with TensorFlow Lite it runs with IDs instead. It also pads the source ids to be the same size as the output during the <unk> replacement. When finding which words in the target are unknown words, it uses the unknown ID.

6. TensorFlow Lite Unit Tests

The tests can be found in the file *opennmt/tests/tflite_test.py*.

6.1 Test Descriptions

- **testTFLiteOutput**—Compares the output of running normally versus running the TensorFlow Lite compatible function to make sure that they produce the same output.
- **testTFLiteInterpreter (new)**—Converts model to TensorFlow Lite and runs the converted model with the Python TensorFlow Lite Interpreter.

6.2 Parameterized

The parameterized library is required to run the tests. This library is useful as it allows you to give a unit test many parameters and give it as a list so it runs with different parameters each time.

TestTFLiteOutput parameters example

```
@parameterized.expand(
    [
        [catalog.NMTBigV1, {}],
        [catalog.NMTBigV1, {"replace_unknown_target": True}],
        [catalog.NMTBigV1, {"beam_width": 3}],
    ]
)
```

TestTFLiteInterpreter parameters example

```
@parameterized.expand(
    [
     [catalog.TransformerBase, {"replace_unknown_target": True}],
        [catalog.TransformerBase, {"beam_width": 3}],
        [catalog.TransformerBase, {}, 'dynamic_range'],
    ]
)
```

6.3 Arguments' Description

The testTFLiteOutput now takes in optional parameters from the data-configuration file, so it can now test with beam search and <unk> replacement.

The testTFLiteInterpreter can be provided with the model you want to test, the data-configuration file (optional) to test parameters like <unk> replacement and beam search, and the quantization to convert the model with (optional).

1	DEFENSE TECHNICAL
(PDF)	INFORMATION CTR
	DTIC OCA
1	DEVCOM ARL
(PDF)	FCDD RLD DCI
	TECH LIB
1	DEVCOM ARL
(PDF)	FCDD RLC IB
	S LAROCCA
1	ADVANCED RESOURCE TECHNOLOGIES, INC.
(PDF)	G CERVANTES