Transformer Translation Model

This is an implementation of the Transformer translation model as described in the Attention is All You Need paper. The implementation leverages tf.keras and makes sure it is compatible with TF 2.x.

Warning: the features in the `transformer/` folder have been fully intergrated into nlp/modeling. Due to its dependencies, we will remove this folder after the model garden 2.5 release. The model in `nlp/modeling/models/seq2seq_transformer.py` is identical to the model in this folder.

Contents

Walkthrough

Below are the commands for running the Transformer model. See the Detailed instructions for more details on running the model.

```
# Ensure that PYTHONPATH is correctly defined as described in
# https://github.com/tensorflow/models/tree/master/official#requirements
export PYTHONPATH="$PYTHONPATH:/path/to/models"

cd /path/to/models/official/legacy/transformer

# Export variables
PARAM_SET=big
DATA_DIR=$HOME/transformer/data
MODEL_DIR=$HOME/transformer/model_$PARAM_SET
VOCAB_FILE=$DATA_DIR/vocab.ende.32768

# Download training/evaluation/test datasets
python3 data_download.py --data_dir=$DATA_DIR

# Train the model for 100000 steps and evaluate every 5000 steps on a single GPU.
# Each train step, takes 4096 tokens as a batch budget with 64 as sequence
# maximal length.
python3 transformer_main.py --data_dir=$DATA_DIR --model_dir=$MODEL_DIR \
    --vocab_file=$VOCAB_FILE --param_set=$PARAM_SET \
    --train_steps=100000 --steps_between_evals=5000 \
    --batch_size=4096 --max_length=64 \
    --bleu_source=$DATA_DIR/newstest2014.en \
    --bleu_ref=$DATA_DIR/newstest2014.de \
```

```
    --num_gpus=1 \
    --enable_time_history=false

# Run during training in a separate process to get continuous updates,
# or after training is complete.
tensorboard --logdir=$MODEL_DIR
```

Detailed instructions

0. **Environment preparation**

   Add models repo to PYTHONPATH

   Follow the instructions described in the Requirements section to add the models folder to the python path.

   Export variables (optional)

   Export the following variables, or modify the values in each of the snippets below:

   ```
   PARAM_SET=big
   DATA_DIR=$HOME/transformer/data
   MODEL_DIR=$HOME/transformer/model_$PARAM_SET
   VOCAB_FILE=$DATA_DIR/vocab.ende.32768
   ```

1. **Download and preprocess datasets**

   data_download.py downloads and preprocesses the training and evaluation WMT datasets. After the data is downloaded and extracted, the training data is used to generate a vocabulary of subtokens. The evaluation and training strings are tokenized, and the resulting data is sharded, shuffled, and saved as TFRecords.

   1.75GB of compressed data will be downloaded. In total, the raw files (compressed, extracted, and combined files) take up 8.4GB of disk space. The resulting TFRecord and vocabulary files are 722MB. The script takes around 40 minutes to run, with the bulk of the time spent downloading and ~15 minutes spent on preprocessing.

   Command to run:

   ```
   python3 data_download.py --data_dir=$DATA_DIR
   ```

   Arguments:

   - `--data_dir` : Path where the preprocessed TFRecord data, and vocab file will be saved.
   - Use the `--help` or `-h` flag to get a full list of possible arguments.

2. **Model training and evaluation**

   transformer_main.py creates a Transformer keras model, and trains it uses keras model.fit().

   Users need to adjust `batch_size` and `num_gpus` to get good performance running multiple GPUs.

   Note that: when using multiple GPUs or TPUs, this is the global batch size for all devices. For example, if the batch size is `4096*4` and there are 4 devices, each device will take 4096 tokens as a batch budget.

   Command to run:

```
python3 transformer_main.py --data_dir=$DATA_DIR --model_dir=$MODEL_DIR \
    --vocab_file=$VOCAB_FILE --param_set=$PARAM_SET
```

Arguments:

- `--data_dir` : This should be set to the same directory given to the `data_download` 's `data_dir` argument.
- `--model_dir` : Directory to save Transformer model training checkpoints.
- `--vocab_file` : Path to subtoken vocabulary file. If data_download was used, you may find the file in `data_dir` .
- `--param_set` : Parameter set to use when creating and training the model. Options are `base` and `big` (default).
- `--enable_time_history` : Whether add TimeHistory call. If so, --log_steps must be specified.
- `--batch_size` : The number of tokens to consider in a batch. Combining with `--max_length` , they decide how many sequences are used per batch.
- Use the `--help` or `-h` flag to get a full list of possible arguments.

Using multiple GPUs

You can train these models on multiple GPUs using `tf.distribute.Strategy` API. You can read more about them in this guide.

In this example, we have made it easier to use is with just a command line flag `--num_gpus` . By default this flag is 1 if TensorFlow is compiled with CUDA, and 0 otherwise.

- --num_gpus=0: Uses tf.distribute.OneDeviceStrategy with CPU as the device.
- --num_gpus=1: Uses tf.distribute.OneDeviceStrategy with GPU as the device.
- --num_gpus=2+: Uses tf.distribute.MirroredStrategy to run synchronous distributed training across the GPUs.

Using Cloud TPUs

You can train the Transformer model on Cloud TPUs using `tf.distribute.TPUStrategy` . If you are not familiar with Cloud TPUs, it is strongly recommended that you go through the quickstart to learn how to create a TPU and GCE VM.

To run the Transformer model on a TPU, you must set `--distribution_strategy=tpu` , `--tpu=$TPU_NAME` , and `--use_ctl=True` where `$TPU_NAME` the name of your TPU in the Cloud Console.

An example command to run Transformer on a v2-8 or v3-8 TPU would be:

```
python transformer_main.py \
  --tpu=$TPU_NAME \
  --model_dir=$MODEL_DIR \
  --data_dir=$DATA_DIR \
  --vocab_file=$DATA_DIR/vocab.ende.32768 \
  --bleu_source=$DATA_DIR/newstest2014.en \
  --bleu_ref=$DATA_DIR/newstest2014.end \
  --batch_size=6144 \
  --train_steps=2000 \
  --static_batch=true \
  --use_ctl=true \
```

```
  --param_set=big \
  --max_length=64 \
  --decode_batch_size=32 \
  --decode_max_length=97 \
  --padded_decode=true \
  --distribution_strategy=tpu
```

Note: `$MODEL_DIR` and `$DATA_DIR` must be GCS paths.

Customizing training schedule

By default, the model will train for 10 epochs, and evaluate after every epoch. The training schedule may be defined through the flags:

- Training with steps:
  - `--train_steps` : sets the total number of training steps to run.
  - `--steps_between_evals` : Number of training steps to run between evaluations.

Compute BLEU score during model evaluation

Use these flags to compute the BLEU when the model evaluates:

- `--bleu_source` : Path to file containing text to translate.
- `--bleu_ref` : Path to file containing the reference translation.

When running `transformer_main.py` , use the flags: `--bleu_source=$DATA_DIR/newstest2014.en --bleu_ref=$DATA_DIR/newstest2014.de`

Tensorboard

Training and evaluation metrics (loss, accuracy, approximate BLEU score, etc.) are logged, and can be displayed in the browser using Tensorboard.

```
tensorboard --logdir=$MODEL_DIR
```

The values are displayed at localhost:6006.

Implementation overview

A brief look at each component in the code:

Model Definition

- transformer.py: Defines a tf.keras.Model: `Transformer` .
- embedding_layer.py: Contains the layer that calculates the embeddings. The embedding weights are also used to calculate the pre-softmax probabilities from the decoder output.
- attention_layer.py: Defines the multi-headed and self attention layers that are used in the encoder/decoder stacks.
- ffn_layer.py: Defines the feedforward network that is used in the encoder/decoder stacks. The network is composed of 2 fully connected layers.

Other files:

- beam_search.py contains the beam search implementation, which is used during model inference to find high scoring translations.

Model Trainer

transformer_main.py creates an `TransformerTask` to train and evaluate the model using tf.keras.

Test dataset

The newstest2014 files are extracted from the NMT Seq2Seq tutorial. The raw text files are converted from the SGM format of the WMT 2016 test sets. The newstest2014 files are put into the `$DATA_DIR` when executing `data_download.py`