Lab 2: Benchmarking

The goal of this lab is for you to benchmark and compare model inference efficiency on your devices. **You should benchmark 2*N* models or model variants, where *N* is the size of your group (so, two models per person.)** For now, if you don't have appropriate evaluation data in place that's fine; you can provide pretend data to the model for now and just evaluate efficiency.

Ideally, the models you benchmark will be related to and useful for your class project, but at the very least the efficiency metrics should be useful.

Include any code you write to perform this benchmarking in your Canvas submission (either as a link to a folder on github, in a shared drive, zip, etc).

Group name: How Do You Turn This On

Group members present in lab today: Yi-Ting Yeh, Ting-Rui Chiang

1: Models

1. Which models and/or model variants will your group be benchmarking? Please be specific.
    a. Transformer Model: Transformer ASR model whose encoder is Conformer and decoder is hybrid of CTC and attention models. We adjust CTC weights in {0, 0.3, 1.0}. When CTC weight=0, the model uses only attention decoder. When CTC weight = 1, the model uses only CTC decoder.
    b. LSTM Model: LSTM ASR model with CTC decoder. CTC weights in {0, 0.3, 1.0}
2. Why did you choose these models?

The Transformer model is a state-of-the-art ASR model with respect to WER, but its model size is huge. Therefore, we also consider a smaller model LSTM model. LSTM model uses a pure LSTM encoder and a hybrid decoder and can serve as a baseline of end-to-end ASR systems. While the best performance of ASR systems is usually achieved when using the hybrid of CTC deocder and attention decoder, we are interested in the performance differences of using only one decoder. Therefore, for each of the two models, we test 3 variants that use only CTC decoder, attention decoder, or a hybrid of two decoders.

3. For each model, you will measure parameter count, inference latency, and energy use. For latency and energy, you will also be varying a parameter such as input size or batch size. What are your hypothesis for how the models will compare according to these metrics? Explain.
    a. We hypothesize the Transformer model will have a better WER rate but drastically larger parameter counts, inference latency, and energy use than the LSTM model.
    b. In terms of the decoder choice, we expect the CTC decoder will be more efficient than the attention decoder since it does not have an overhead of computing attention matrix.
    c. If we increase the input size or batch size, Transformer models should be benefited more since its computation can be easily parallized. It is because it don't have any recurrent architectures.

2: Parameter count

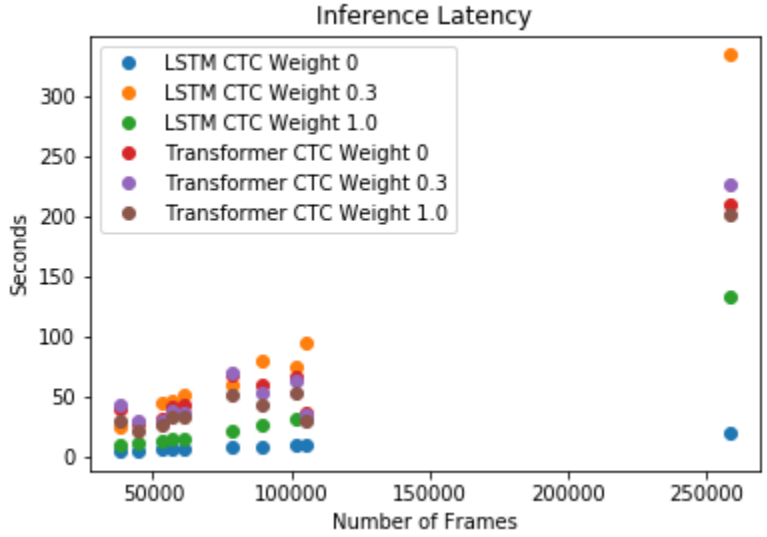
1. Compute the number of parameters in each model.
    a. Transformer-hybrid: There are 116146960 parameters in the full hybrid model. CTC decoder has 2565000 parameters and the attention decoder has 30350216 parameters.
    b. LSTM: There are 8248145 parameters in the full hybrid model. CTC decoder has 8667 parameters and the attention decoder has 1459318 parameters.
2. Does this number account for any parameter sharing that might be part of the model you're benchmarking?
    a. In both the Transformer and LSTM models in our experiments, there are no commonly used parameter sharing techniques such as tied embedding. Therefore, the number reflects the actual parameter counts in the model.
3. Any difficulties you encountered here? Why or why not?
    a. No, It is fairly easy to count a number of parameters since we use the ESPNet library to run all models.

3: Latency

1. Compute the inference latency of each model. You should do this by timing the forward pass only.
   For each model, we use the averaged time of inferencing 100 samples.
    a. Transformer-hybrid: The latency is 99.378, 91.180, and 103.448 when we use CTC weights 0, 1, and 0.3 respectively.
    b. LSTM model: The latency is 10.265, 51.851, and 143.83 when we use CTC weights 0, 1, and 0.3 respectively.

2. Repeat this, but try varying one of: batch size, input size, other. Plot the results

Here we vary the number of input frames. By doing this experiment, we want to observe the model behaviors on different input sizes.



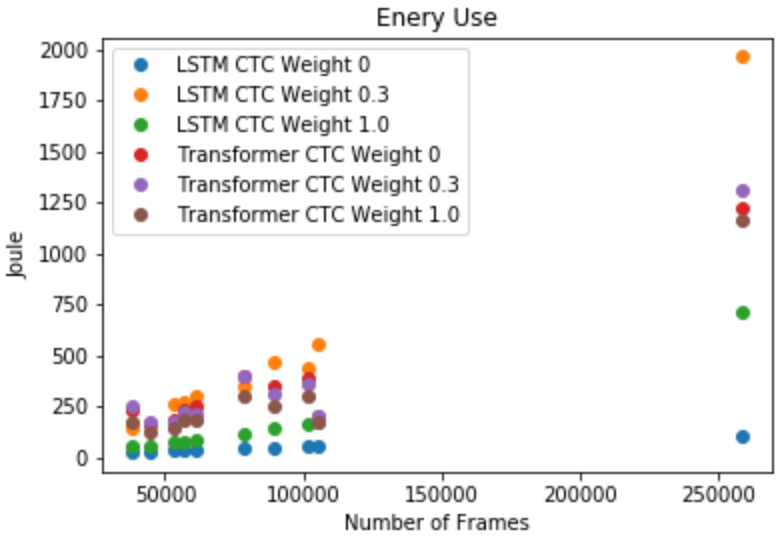
3. Any difficulties you encountered here? Why or why not?
    a. We have difficulty executing the ASR model on our device. We use ESPNet for our ASR model. In the library, it uses torch.stft for data preprocessing. Unfortunately, the implementation of torch.stft relies on MKL, which is a x86-only library. Therefore, we have to replace torch.stft with another implementation by using librosa. It requires more modification to make the replacement support batch size greater than 1.

4: Energy use

1. Compute the energy use of each model. You can use the powertop tool on RPi and Jetson (must be run as root):

For each model, we use the averaged watt of 60 samples after running models for 60 seconds.

    a. Transformer-hybrid: The energy use is 5.82, 5.787, and 5.778 watt when we use CTC weights 0, 1, and 0.3 respectively. If we times watt with the averaged inference time, it is 579.07, 527.66, and 597.72 Joule for CTC weights 0, 1, and 0.3.
    b. LSTM model: The energy use is 5.786, 5.3789, and 5.885 watt when we use CTC weights 0, 1, and 0.3 respectively. If we times watt with the averaged inference time, it is 59.39, 278.90, and 845.72 for CTC weights 0, 1, and 0.3.
2. Plot the results with respect to the varied input frames:



3. Any difficulties you encountered here? Why or why not?
    a. Due to the library version issue, we can't do batched inference on our device which is an ARM architecture.
    b. The energy usage is the average of 60 samples in 60 seconds estimated by the command powertop. We have no idea how accurate it is.

5: Discussion

1. Analyze the results. Do they support your hypotheses? Why or why not?

In general, the results are aligned with our hypotheses, which we expect the Transformer and attention decoder to have a higher latency and energy use. The only outlier is the LSTM model using the hybrid decoder. It has the largest latency and energy use. We think it is because the inference time of LSTM increases drastically with respect to the input length, as shown in the figure in Section 4.2. Therefore, the LSTM model with the hybrid decoder suffers the most when dealing with long input size.

We also experimented with models on the real dataset Librispeech. We use the clean test set and test models with the hyperparamter setting same as the above experiments. Transformer models have 54.18, 4.33 and 3.23 WER when we use CTC weights 0, 0.3 and 1.0 respectively. The LSTM model with all of the settings has WER larger than 99, which indicates it can't give us any reasonable results. It seems that the Transformer model has the best trade-off between actual performance and efficiency.