**wav2letter++**: A Fast Open-Source Speech Recognition Framework

**Vineel Pratap, Avni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, Ronan Collobert**

**INTRODUCTION**

- wav2letter++ is a fast, open-source deep learning speech recognition framework.
- Written entirely in C++ and backed by the efficient ArrayFire tensor library.
- Scales linearly to 64 GPUs for models with 100+ million parameters.
- Over 2x faster in some cases than other optimized frameworks for training end-to-end neural networks for speech recognition.

**DESIGN**

- The design of wav2letter++ is motivated by three requirements:
  - It must efficiently train models on datasets containing thousands of hours of speech.
  - Make expressing and incorporating new network architectures, loss functions, and other operations easy, and
  - Make the path from model research to deployment straightforward, requiring as little new code as possible while maintaining the flexibility needed for research.

**Flashlight Machine Learning Library**

- **ArrayFire Tensor Library**
  - ArrayFire [1] is a highly-optimized tensor library that supports CPU, GPU, and OpenCL backends.
  - Uses just-in-time (JIT) code generation to combine series of simple operations into a single kernel call.
  - Less verbose and relies on fewer C++ idiosyncrasies.

- **Flashlight Machine Learning Library**
  - A standalone machine learning library that:
    - Extends ArrayFire with autograd, NN modules, distributed training, etc. to support neural network training.
    - Extends the core ArrayFire CUDA back-end with more efficient cuDNN operations including convolutions and RNN operations.
  - wav2letter++ library is built on top of flashlight.

**wendlin++**

- **Autograd**
  - Variable forward(const Variables x) {
    auto hidden = matmul(weights[i], x);
    hidden = max(hidden, 0f);
    return matmul([weights[i], hidden]);
  }

- **Optimization**
  - Variable criterion(const Variables yhat, const Variables y) {
    auto probe = sigmoid(yhat);
    return -(y * log(probe) + (1 - y) * log1p(-probe));
  }

  ```
  for (const auto &xy : trainSet)
  criterion(yhat, y).backward();
  for (auto &w : weights) {
    w -= lr * w.grad();
    w.zeroGrad(); //set gradient to zero
  }
  ```

  Example: one layer MLP trained with binary cross-entropy and SGD, using autograd

**Data Preparation and Feature Extraction**

- wav2letter++ supports multiple audio file formats (e.g. wav, flac,.. / mono, stereo / int, float) and several feature types including raw audio, a linearly scaled power spectrum, log-Mels (MFSC) and MFCCs.
- Data loading computes features on the fly prior to each network evaluation.
- To make this efficient while training models, we load the audio and compute the features asynchronously and in parallel with inference.

**Models**

- We support several end-to-end sequence models with loss functions including Connectionist Temporal Classification (CTC) [6], wav2letter’s AutoSegmentationCriterion (ASG) criterion [2], and sequence-to-sequence models with attention (seq2seq).
- Adding a new sequence criterion is particularly easy; ASG and CTC are already efficiently implemented in C++.
- Since the flashlight library we use provides dynamic graph construction and automatic differentiation, building new layers or other primitive operations requires little effort.

**Training and Scale**

- Flexibility for the user to experiment with different features, architectures, and optimization parameters. Hackable to the core.
- Training can be run in three modes:
  - train (flat-start training)
  - continue (continuing with a checkpoint state)
  - fork (for e.g. transfer learning)
- We scale wav2letter++ to larger datasets with data-parallel synchronous and asynchronous SGD and provide a simple framework with which to create custom distributed optimization schemes.

**Decoding**

- The wav2letter++ decoder is performance-optimized beam-search decoder which:
  - Supports any type of language model which exposes the interface required by our decoder including n-gram LMs and any other stateless parametric LM
  - Supports online decoding, where emissions are streamed into the decoder.

**BENCHMARKS**

**Training Performance by Component**

- wav2letter++

**Decoding Speed and Throughput**

**REFERENCES**

3. Daniel Povey, Avni Hannun, and Gabriel Synnaeve, “Wav2letter: AutoSegmentationCriterion (ASG)” criterion [2], and sequence-to-sequence models with attention (seq2seq). Adding a new sequence criterion is particularly easy; ASG and CTC are already efficiently implemented in C++.
4. Vitaliy Liptchinsky, Vineel Pratap, Tensor Processing Unit (TPU) architecture.

**Distributed Training Scales Linearly**

<table>
<thead>
<tr>
<th>Name</th>
<th>WER (%)</th>
<th>Time/sample (ms)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPNet</td>
<td>7.20</td>
<td>1458</td>
<td>-</td>
</tr>
<tr>
<td>wav2letter++</td>
<td>5.00</td>
<td>1700</td>
<td>7.8</td>
</tr>
<tr>
<td>wav2letter++</td>
<td>4.91</td>
<td>140</td>
<td>3.9</td>
</tr>
</tbody>
</table>

Decoding performance on Librispeech dev-clean.