tinyML® Summit

Miniature dreams can come true…

March 28-30, 2022 | San Francisco Bay Area

TINY ML

www.tinyML.org

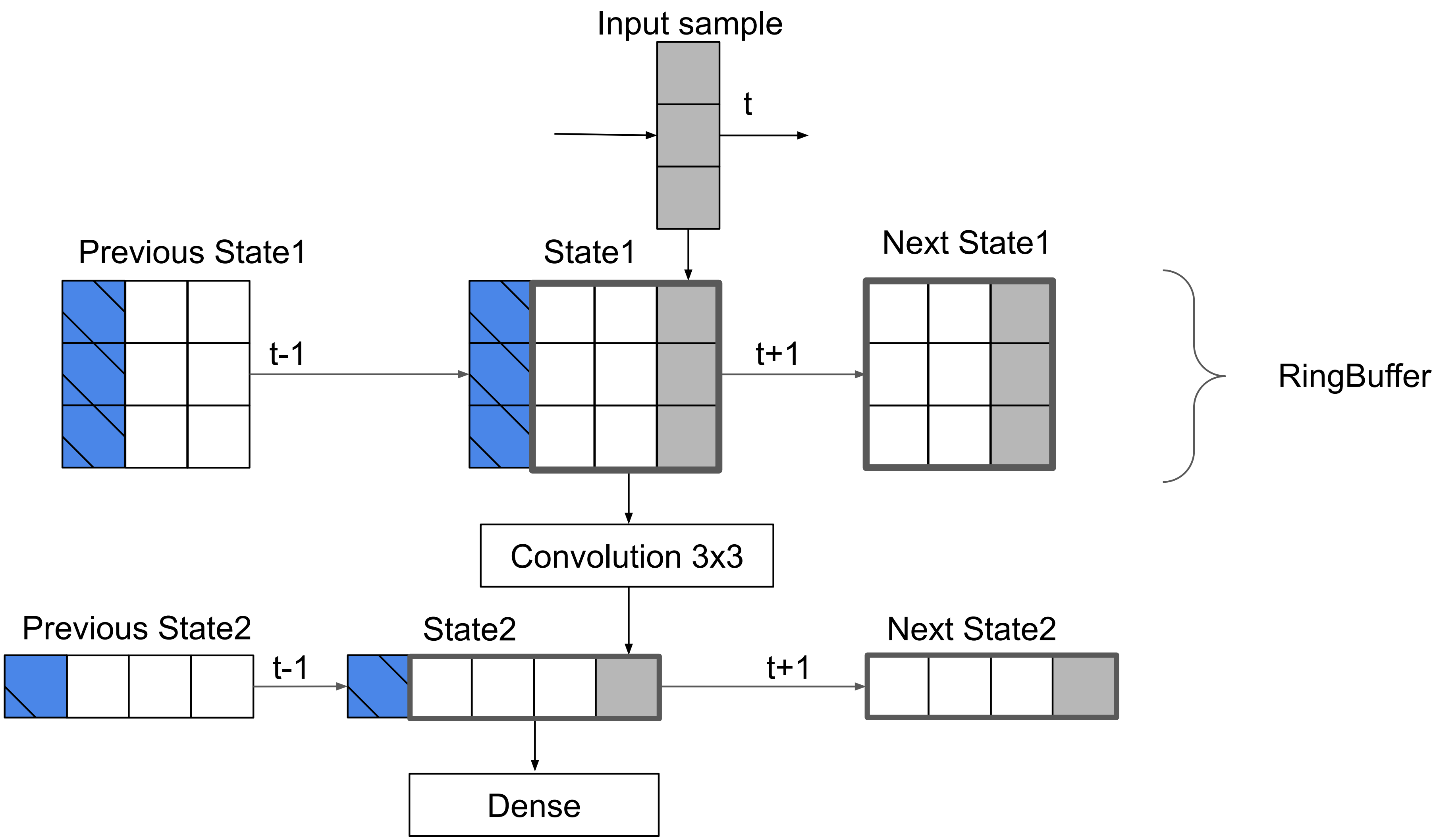
Speech models design and deployment on device.

Oleg Rybakov
rybakov@google.com
Catalyst team

Agenda:

- Performance (latency, memory footprint, power consumption):
  - Streaming
    - Functional API
    - Subclass API
  - Quantization
    - Post training quantization
    - Quantization aware training
      - Fake
      - Native
- Applications:
  - Hotword detection:
    - Multihead self attention
    - Matchbox
  - Streaming multihead self attention

Model streaming (using ring buffers: State1, State2):



Functional API:

Non streaming model:

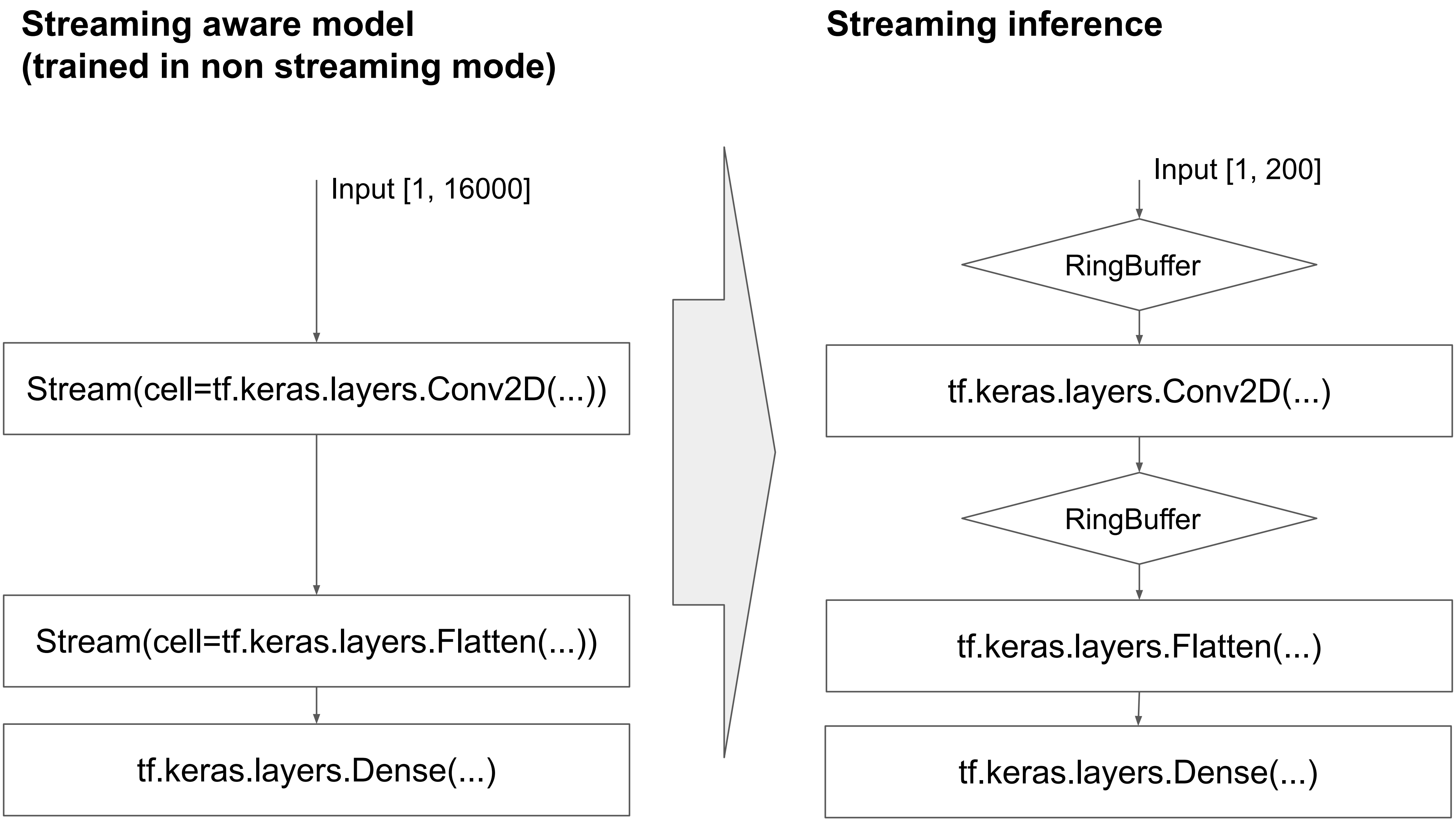
```
net = tf.keras.layers.Conv2D(...)(net)
net = tf.keras.layers.Flatten(...)(net)
net = tf.keras.layers.Dense(...)(net)
```

Streaming aware model:

```
net = Stream(cell=tf.keras.layers.Conv2D(...))(net)
net = Stream(cell=tf.keras.layers.Flatten(...))(net)
net = tf.keras.layers.Dense(...)(net)
```

Applications:

- Real-time Speech Frequency Bandwidth Extension. Yunpeng Li et al
- SoundStream: An End-to-End Neural Audio Codec. Neil Zeghidour, et al
- Streaming keyword spotting on mobile devices. Oleg Rybakov, et al
- Real time spectrogram inversion on mobile phone. Oleg Rybakov, et al

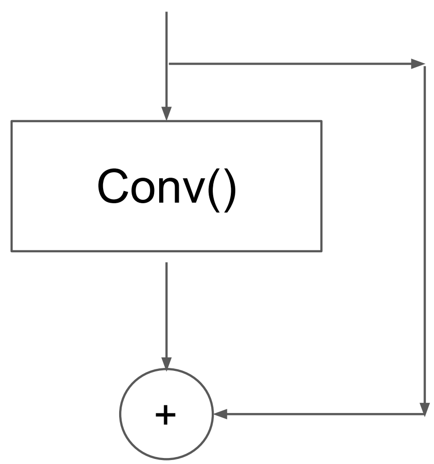

Streaming aware model
(trained in non streaming mode)
Streaming inference
Input [1, 16000]
Stream(cell=tf.keras.layers.Conv2D(...))
Stream(cell=tf.keras.layers.Flatten(...))
tf.keras.layers.Dense(...)
Input [1, 200]
RingBuffer
tf.keras.layers.Conv2D(...)
RingBuffer
tf.keras.layers.Flatten(...)
tf.keras.layers.Dense(...)

Subclass API, lingvo, keras

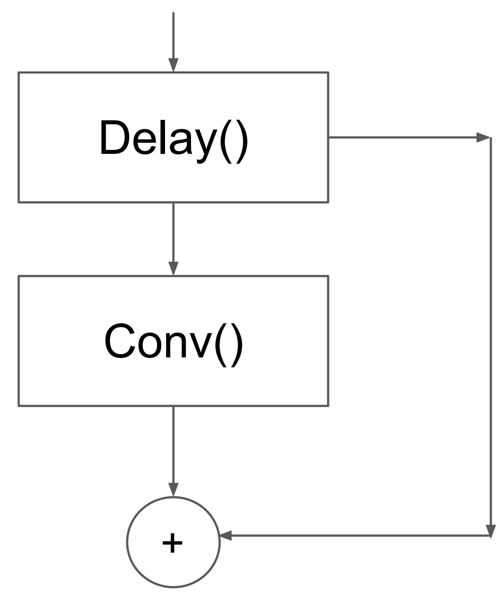
```
def __init__(self, label_count, apply_quantization, **kwargs):

  self.conv = ring_buffer.RingBuffer(quantize.quantize_layer(tf.keras.layers.Conv2D()))  # Create quantization, streaming aware layer

def call(self, inputs): # Forward propagation for model training

  net = self.conv(inputs)

  return net

def stream_inference(self, inputs, states): # Inference in streaming mode

  net, output_state = self.conv(net, state=states)

  return net, output_state
```

Edge cases: causal vs non causal conv

Causal Conv



Non causal Conv



Why model quantization?

- Speed up computation.
- Reduce memory footprint.
- Reduce power consumption.

Post training quantization(PTQ) with TFLite

- Pros:
  - Dynamic quantization easy to use, works on most models.
  - With int8 quantization can give speed up: 1.5x…4x e.g. on hotword models.
  - Supports full model quantization, but needs representative data sets.
- Cons:
  - Does not support lower bits e.g. int4.
  - There is numerical difference between float and quantized models.
  - On more complicated model e.g. auto-regressive models numerical difference can increase.

Fake quantization aware training(QAT)

- Pros:
  - Supports lower than 8bits quantization (inference engine also has to support it).
  - Does not need calibration data for full model quantization (vs post training quantization).
  - Easy to use on functional tf.keras.
- Cons:
  - During model training it uses fake quantization (uses float ops), if results of float summation and multiplication does not fit into 23 bits mantissa there will be a numerical difference between forward propagation in training and inference modes (when integer runs with int ops).
  - After model training it needs post training quantization step which will convert float ops to int ops.

$$\frac{\text{matmul}\left(\textbf{float}\left(\text{int8(weight * scale\_w)}\right),\ \textbf{float}\left(\text{int8(act * scale\_a)}\right)\right)}{\text{scale\_w * scale\_a}}$$

"Emulates dequantization"

int8(x) = np.round(np.clip(x, -127, 127))

Native quantization aware training(QAT)

- Pros:
  - What you train is what you serve: no need in post training quantization step.
  - Speed up not only for inference but for training too (will use int8, int4 ops).
  - Forward propagation during training and inference are numerically the same.
- Cons:
  - User will have to manage quantized types and variables (need int8/int4 types), including custom gradients for int ops.

$$\frac{\text{matmul}\left(\textbf{int8}(\text{weight} * \text{scale\_w}),\ \textbf{int8}(\text{act} * \text{scale\_a})\right)}{\text{scale\_w} * \text{scale\_a}}$$

Observations:

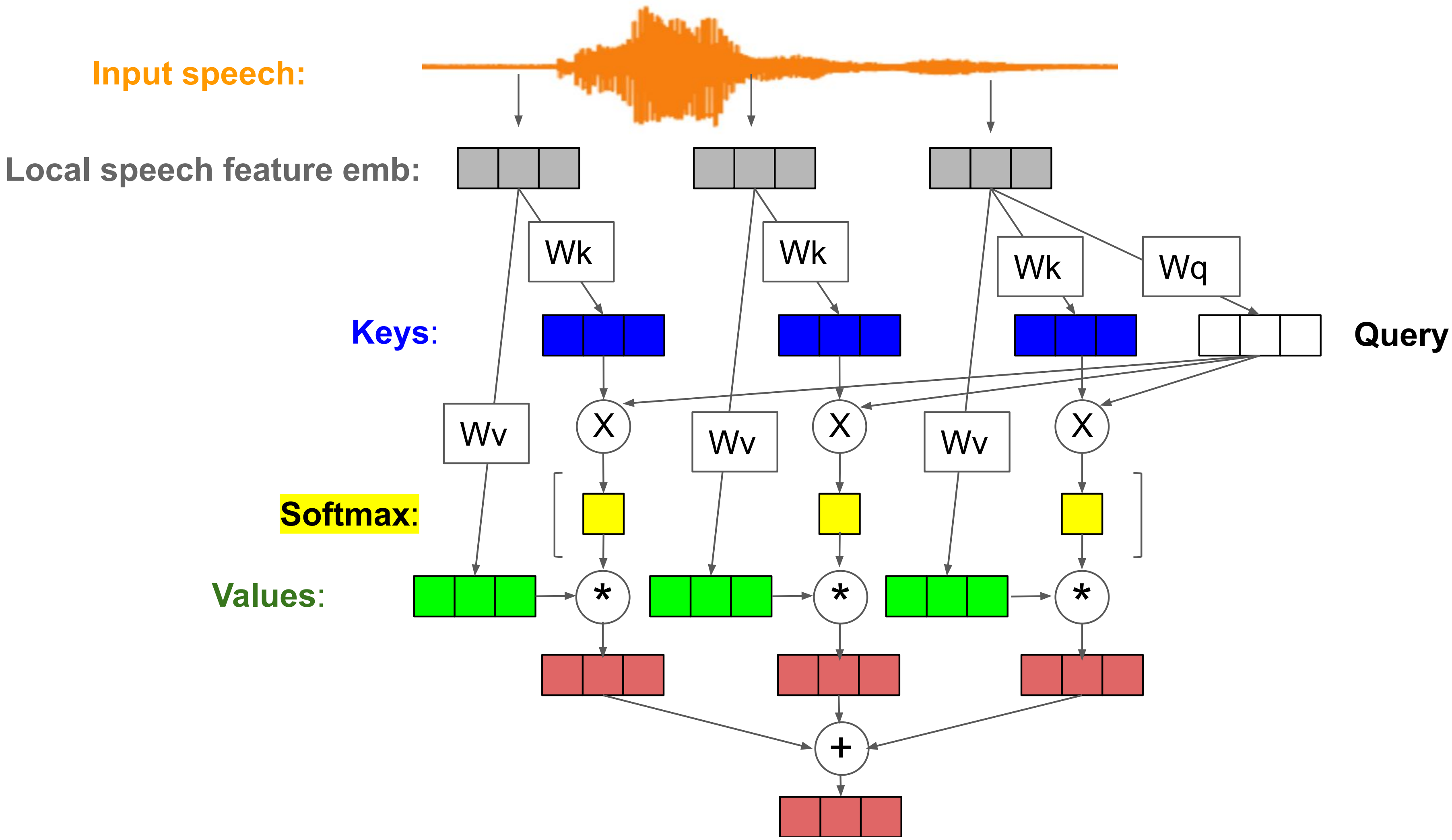
- int8 activation + int8 weights. QAT and PTQ works.
- int8 activation + int4 weights. only QAT works on most models.
- int4 activation + int4 weights. only QAT mostly works on very large models, can require some model tuning.
- < int4 needs significant model modifications. e.g. PokeBNN

Applications: Hotword detection

Google Speech commands data V2 with 12 labels:
”yes”, ”no”, ”up”, ”down”, ”left”, ”right”, ”on”, ”off”, ”stop”, and ”go”; “silence” “unknown”

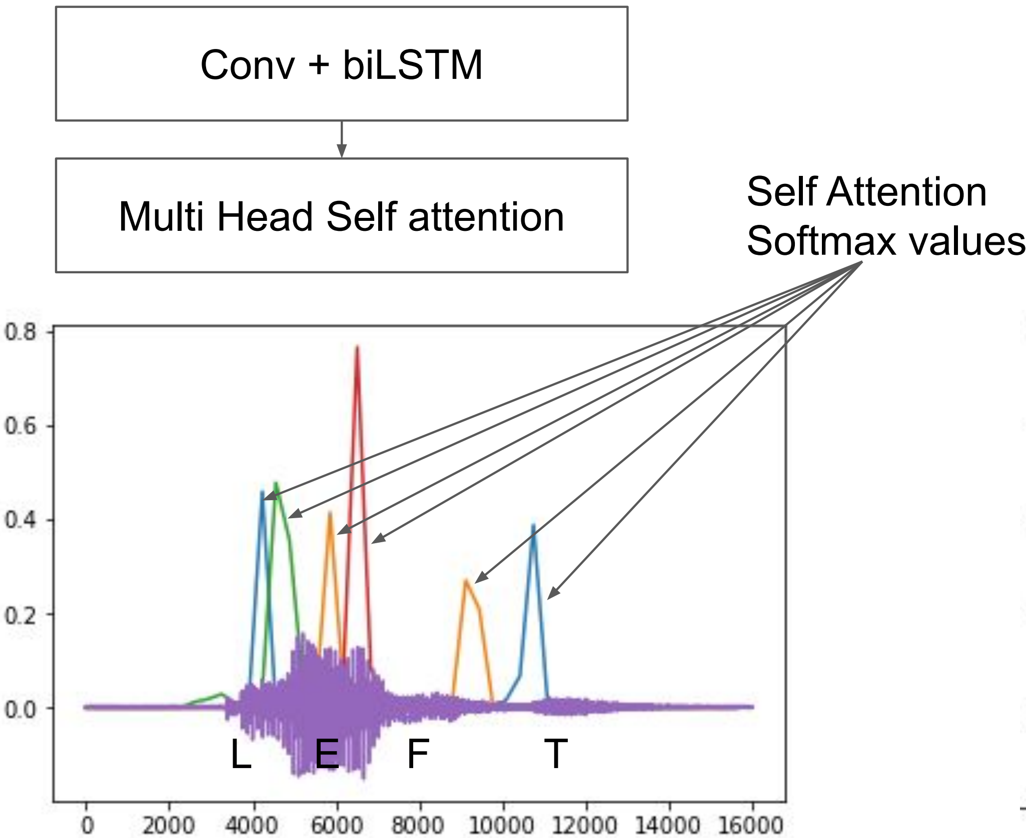
Model	**Pixel4 CPU[ms] Latency of processing 1 sec in non streaming**	**Pixel4 CPU[ms] Latency of processing 20ms in streaming mode**	**Accuracy[%]**	**# Parameters**
MHAtt-RNN (non causal)	13	N/A	98.4	750K
Matchbox (non causal)	3.0	N/I	98.0	75K
Matchbox (causal)	3.0	0.2	97.4	75K

Transformer single head self attention (causal)

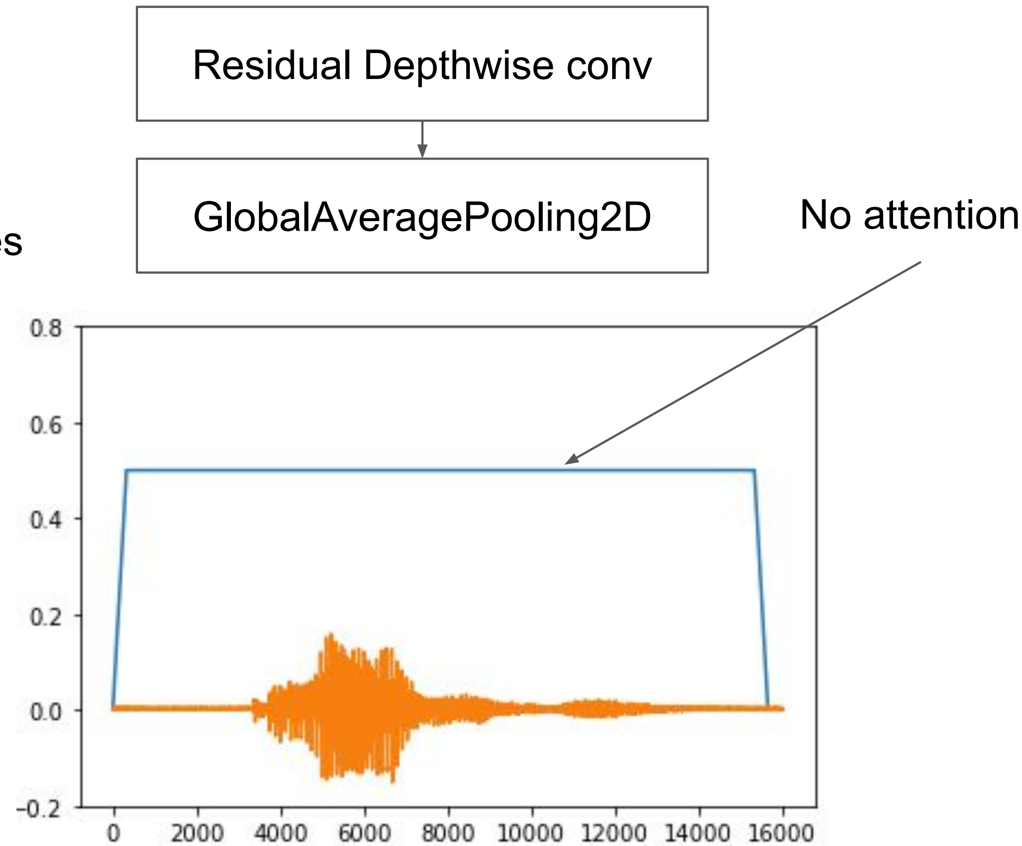


Hotword detection

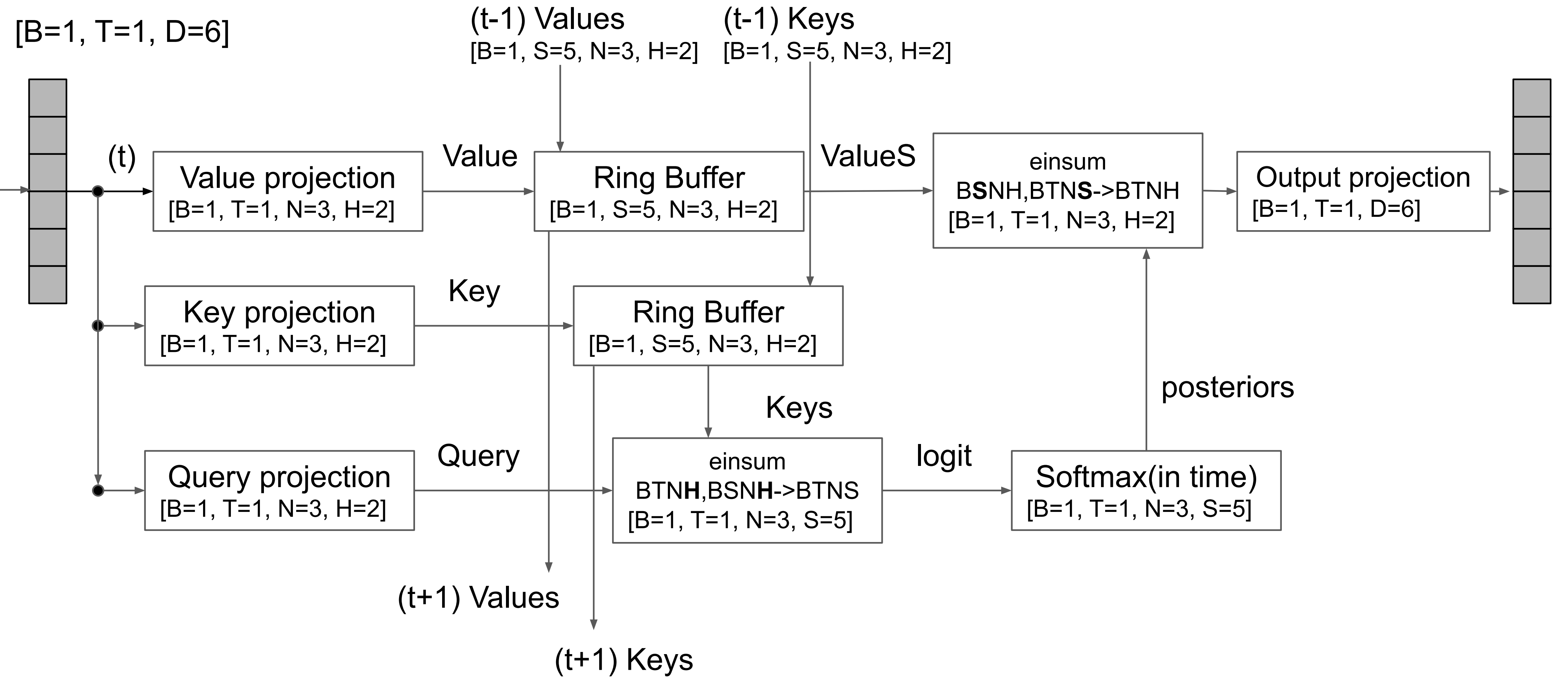
MHAtt-RNN (non causal)



Matchbox (non causal)



Streaming Multihead Self Attention



Conformer encoder for ASR, S2S.

Processing Audio every 20ms with Speech frontend and Conformer Encoder (17 conformer layers with causal local attention = 60).

conformer encoder	**Pixel6 CPU[ms] with TFLite**
float32	31ms/960MB
post-quantized (int8)	13ms/160MB

References

- Yunpeng Li et al Real-time Speech Frequency Bandwidth Extension.
- Oleg Rybakov, et al Streaming keyword spotting on mobile devices.
- Neil Zeghidour, et al SoundStream: An End-to-End Neural Audio Codec.
- Oleg Rybakov, et al Real time spectrogram inversion on mobile phone.
- Somshubra Majumdar, et al MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition.
- Anmol Gulati et al Conformer: Convolution-augmented Transformer for Speech Recognition.
- Benoit Jacob et al, Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference
- AmirAli Abdolrashidi et al, Pareto-Optimal Quantized ResNet Is Mostly 4-bit
- Zhewei Yao et al, HAWQ-V3: Dyadic Neural Network Quantization
- Yichi Zhang et al, PokeBNN: A Binary Pursuit of Lightweight Accuracy
- Hao Wu, LOW PRECISION INFERENCE ON GPU
- Pete Warden, Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition
- https://www.tensorflow.org/lite
- https://github.com/tensorflow/lingvo/blob/master/lingvo/core/conformer_layer.py
- https://github.com/tensorflow/lingvo/blob/master/lingvo/core/batch_major_attention.py
- https://github.com/google-research/google-research/blob/master/kws_streaming
- https://www.tensorflow.org/model_optimization/guide/quantization/training_comprehensive_guide

Appendix

Design patterns for Stream layer

- **Buffer**: RNN layers: GRU, LSTM
- **Ring buffer**: Conv1D, Conv2D, DepthwiseConv2D, Flatten, GlobalMaxPooling2D, GlobalAveragePooling2D layers
- **Remainder buffer**: Conv1DTranspose
- …

tinyML Summit 2022 Sponsors

TINY ML SUMMIT

Copyright Notice