Accelerating Transformer Neural Networks on FPGAs for High Energy Physics Experiments

Filip Wojcicki [1], Zhiqiang Que[1], Alexander D Tapper[2, 3], Wayne Luk[1]

1 Department of Computing, Imperial College London, London, UK
2 Department of Physics, Imperial College London, London, UK
3 CERN

IMPERIAL

2nd FPGA Developers' Forum

Motivation

- Particle physics needs real-time detectors

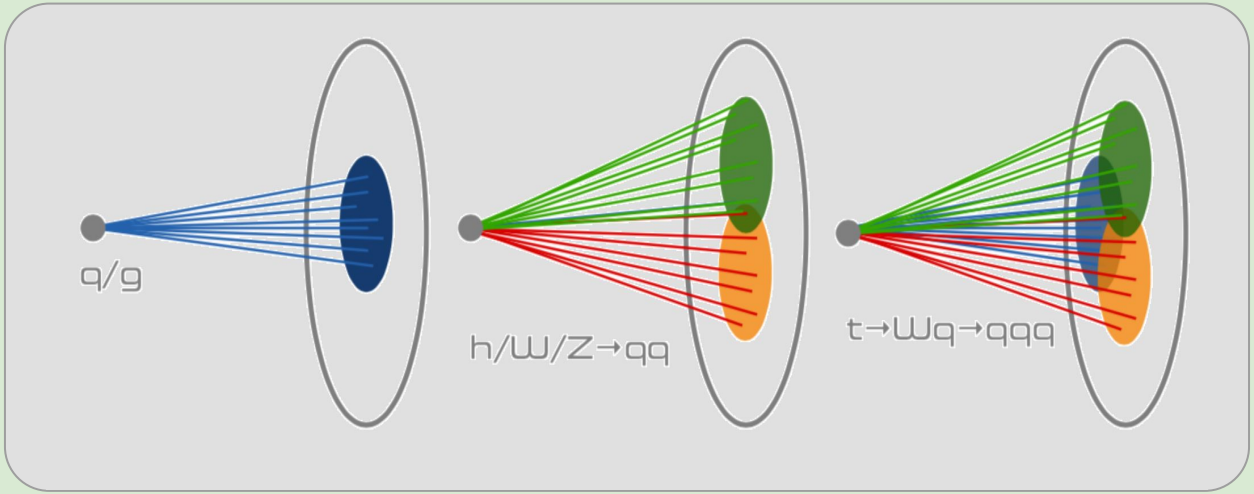
  → Requirement: low-latency & high-accuracy

- Transformer neural networks (TNNs) yield great results in various applications

  → Adapt to jet-tagging

Jet Tagging

- Associating particle jets with their origin
- Proton-proton collision at LHC with 5 jet classes

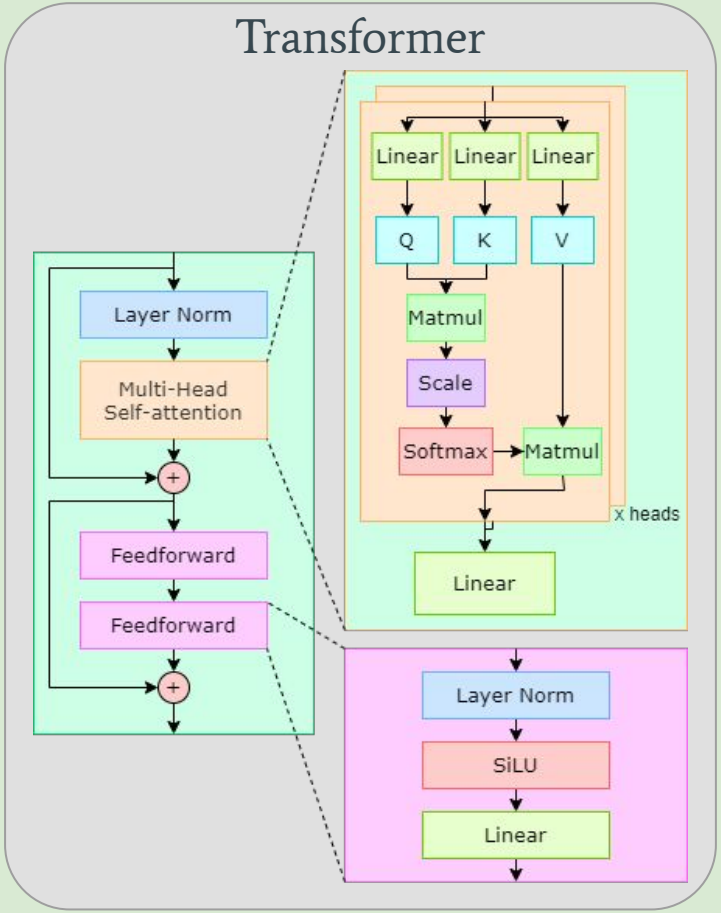


IMPERIAL

Fig: Eric A. Moreno, et al. JEDI-net: a jet identification algorithm based on interaction networks. The European physical journal. URL http://cds.cern.ch/record/2688535.

Transformer & Attention [1]

- Attention mechanism for global dependencies
- No recurrence, only residual connections
- Originally for natural language processing (NLP)
- State-of-the-art in different domains

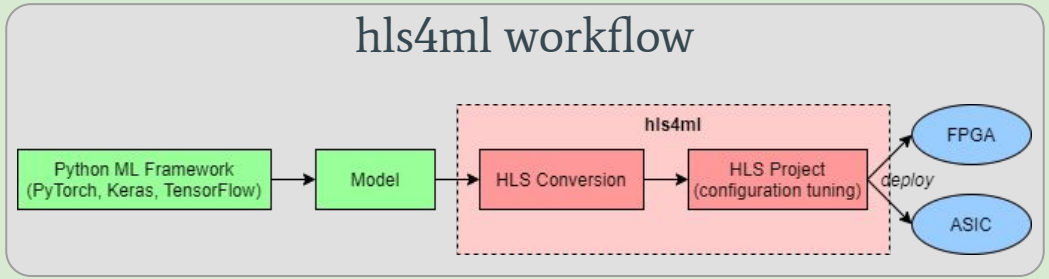


[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. Attention is all you need. Jun 12, 2017. URL https://arxiv.org/abs/1706.03762

hls4ml

- Open-source software-hardware codesign workflow for machine learning models
- No support for transformer and attention (still an ongoing effort in 2025)

  → Use existing template and extend the library



Challenges

- Transformers scale quadratically with latent dimension size

  → Use FPGAs as accelerators to meet latency constraints (L1 Trigger, µs latency)

Time complexity:

$$\mathcal{O}(h \cdot (N_Q N_K (\boldsymbol{d_Q} + \boldsymbol{d_V}) + \boldsymbol{d_Q^2}(N_Q + N_K) + \boldsymbol{d_V^2} N_K + N_Q \boldsymbol{d_V} \boldsymbol{d_{out}})))$$

Assuming $\boldsymbol{d_Q} \approx \boldsymbol{d_V} \gg \boldsymbol{d_{out}}$: $\mathrm{O}(\mathrm{h} \cdot \boldsymbol{d_Q^2})$

- FPGA applications take long time to develop and analyze

  → Make the implementation and design exploration easier

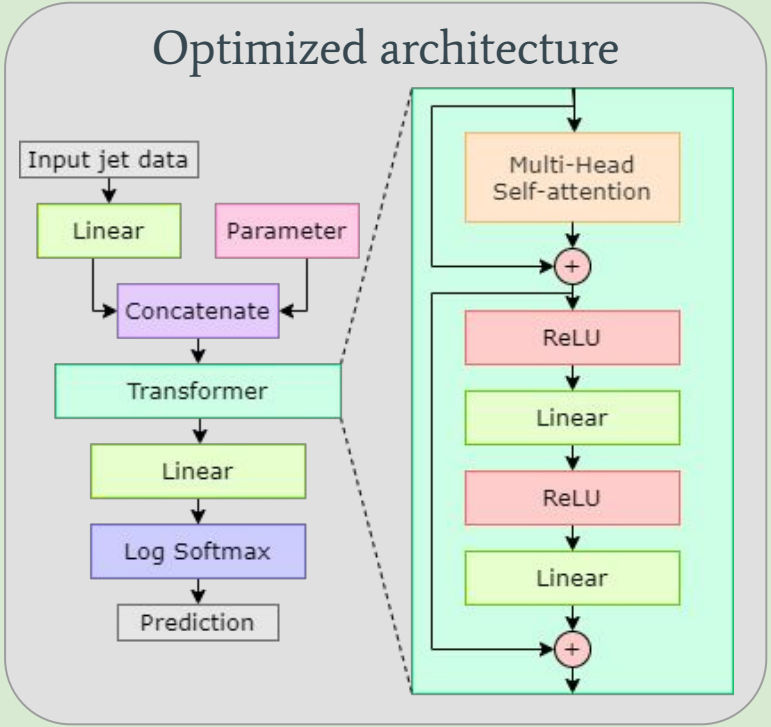
Novelties

- **N1:** Hardware-aware and ultra-low (nanosecond) latency transformer network design
- **N2:** FPGA-specific TNN optimizations (log softmax, matrix splitting)
- **N3:** Highly customizable post-training quantization search algorithm

N1: Ultra-Low Latency Architecture

High-level optimizations:

- Weight matrix splitting
- Functional encoding changed to learnable
- Hardware-friendly activation functions
- Binary scaling instead of division
- Removed normalization layers



IMPERIAL

N1: Inference Comparison

- FPGA design **10^3 - 10^5 x faster** than state-of-the-art designs on GPUs
- More scalable than designs with comparable latency [3]
- Hardware resources can support higher utilization

Evaluation on GPUs

Neural network	Inference per batch (ms)	Accuracy / aver. AUC	FLOPS	Parameters
DNN [2]	**1.0 ± 0.2**	0.760 / 0.941	**27 k**	14,725
CNN [2]	57.1 ± 0.5	0.740 / 0.911	400 k	205,525
GRU [2]	23.2 ± 0.6	0.750 / 0.912	46 k	15,575
JEDI-net [2]	121.2 ± 0.4	- / **0.959**	116 M	33,625
JEDI-net with $\sum O$ [2]	402.0 ± 1.0	- / 0.957	458 M	8,767
Proposed network	1.2 ± 0.1	**0.761** / 0.940	584 k	**2,605**

Evaluation on FPGA

Timing			
Cycles	**Frequency**	**Latency**	**Initiation interval**
18	200 MHz	90 ns	1
Total used (%)			
BRAM 18K	**DSP48E**	**Flip Flop**	**LUT**
12 (0.22%)	4,351 (35.41%)	58,942 (1.71%)	298,881 (17.30%)

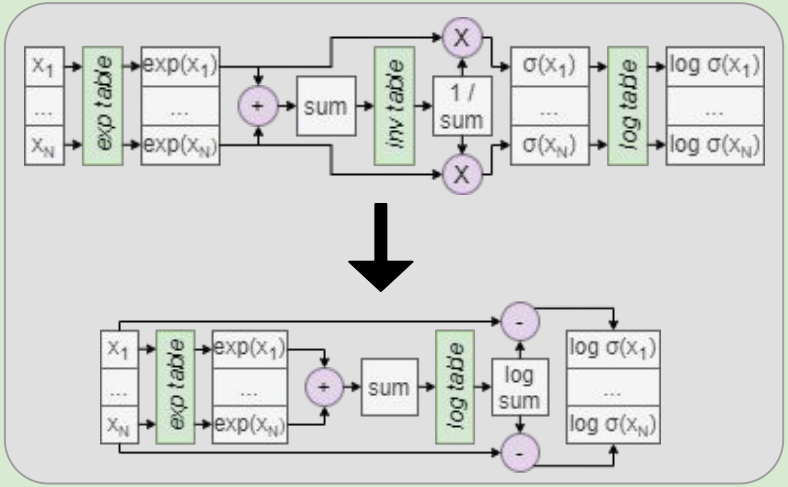
[2] Eric A. Moreno et al. JEDI-net: a jet identification algorithm based on interaction networks. The European physical journal. URL http://cds.cern.ch/record/2688535.

[3] Javier Duarte, et al. Fast inference of deep neural networks in FPGAs for particle physics. Technical Report 13. URL http://cds.cern.ch/record/2316331.

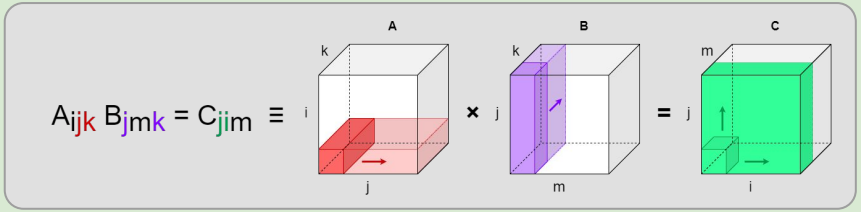
N2: Hardware Optimizations

- Log softmax equivalency formula:

$$\log(\sigma(x_i)) = log(e^{x_i} / \sum_{j=1}^{N} e^{x_j}) = x_i - \log(\sum_{j=1}^{N} e^{x_j})$$



- Tensor product using Einstein Summation



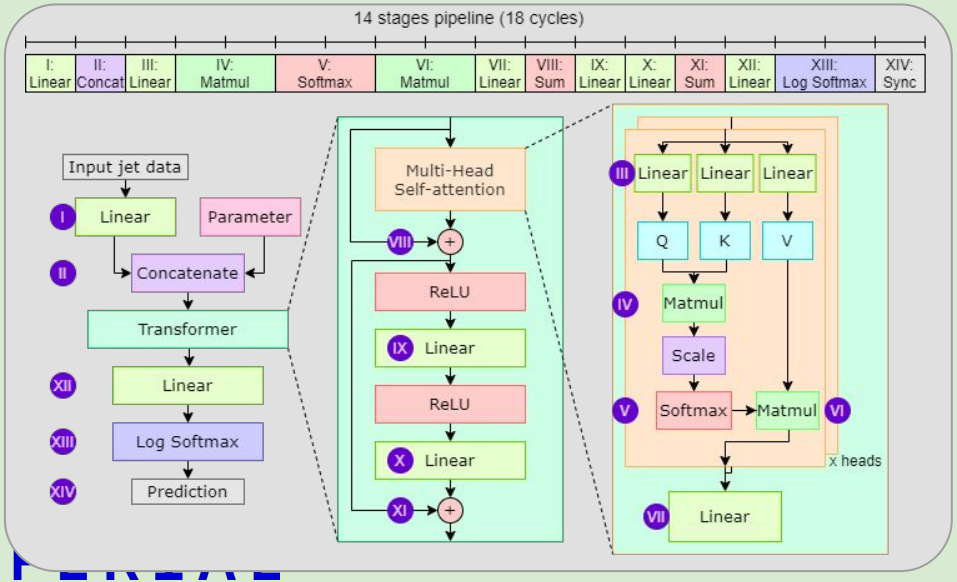

```
torch.einsum("qhc,khc->hqk", [A, B])
```

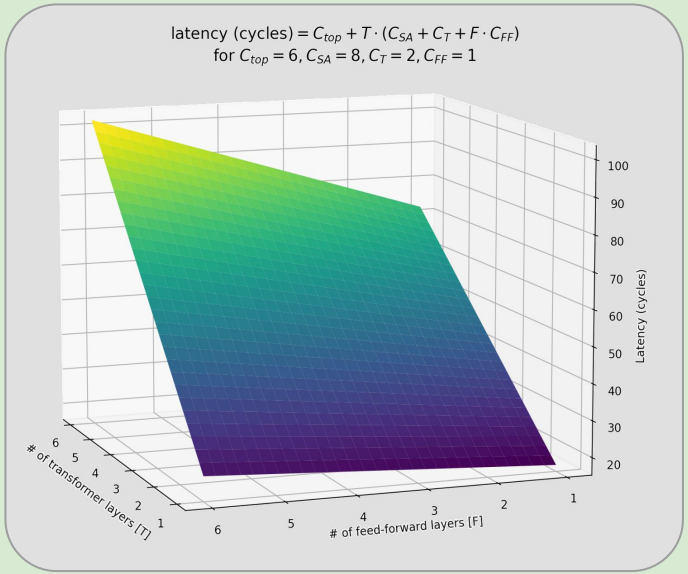
```
#define EINSUM_PIPELINE(DEPTH) {\
  if (io_type == io_parallel) {\
    if (pipeline_einsum_depth == DEPTH) {\
      #pragma HLS PIPELINE II=reuse_factor\
    } else {\
      #pragma HLS unroll \
    }}}

type_t out[H][K][Q];
if (io_type == io_serial) {
  #pragma HLS ARRAY_PARTITION variable=out cyclic factor=cycle_factor
} else if (io_type == io_parallel) {
  #pragma HLS ARRAY_PARTITION variable=out complete
}

Einsum_H: for (unsigned hh = 0; hh < H; hh++) {
  EINSUM_PIPELINE(3);
  Einsum_K: for (unsigned kk = 0; kk < K; kk++) {
    EINSUM_PIPELINE(2);
    Einsum_Q: for (unsigned qq = 0; qq < Q; qq++) {
      EINSUM_PIPELINE(1);
      Einsum_C: for (unsigned cc = 0; cc < C; cc++) {
        EINSUM_PIPELINE(0);
        if (cc == 0) out[hh][kk][qq] = 0;
        out[hh][kk][qq] += A[qq][hh][cc] * B[kk][hh][cc];}}}}
```

IMPERIAL

N2: Hardware Pipeline and Latency Analytical Model

- Directly mapped FPGA pipeline of the model for in-depth analysis
- Closed-form formula for TNN latency, validated by synthesis



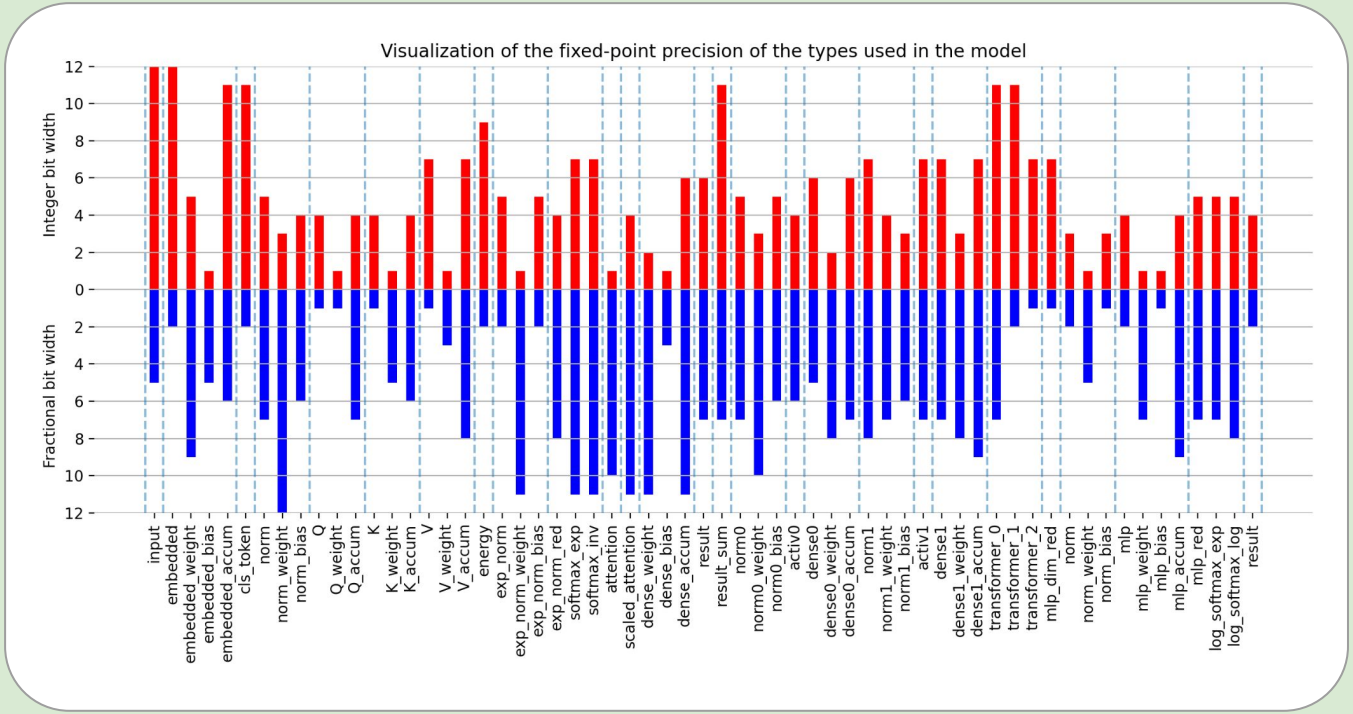


N3: Post-Training Quantization - Proposed Algorithm

- Proposed algorithm is significantly faster than many existing solutions
- Exploration is inherently time-consuming
- User-defined search priorities - accuracy vs hardware usage
- **64% total bit-width reduction compared to FP32** (with 2% accuracy decrease)

N3: Post-Training Quantization - Use Case

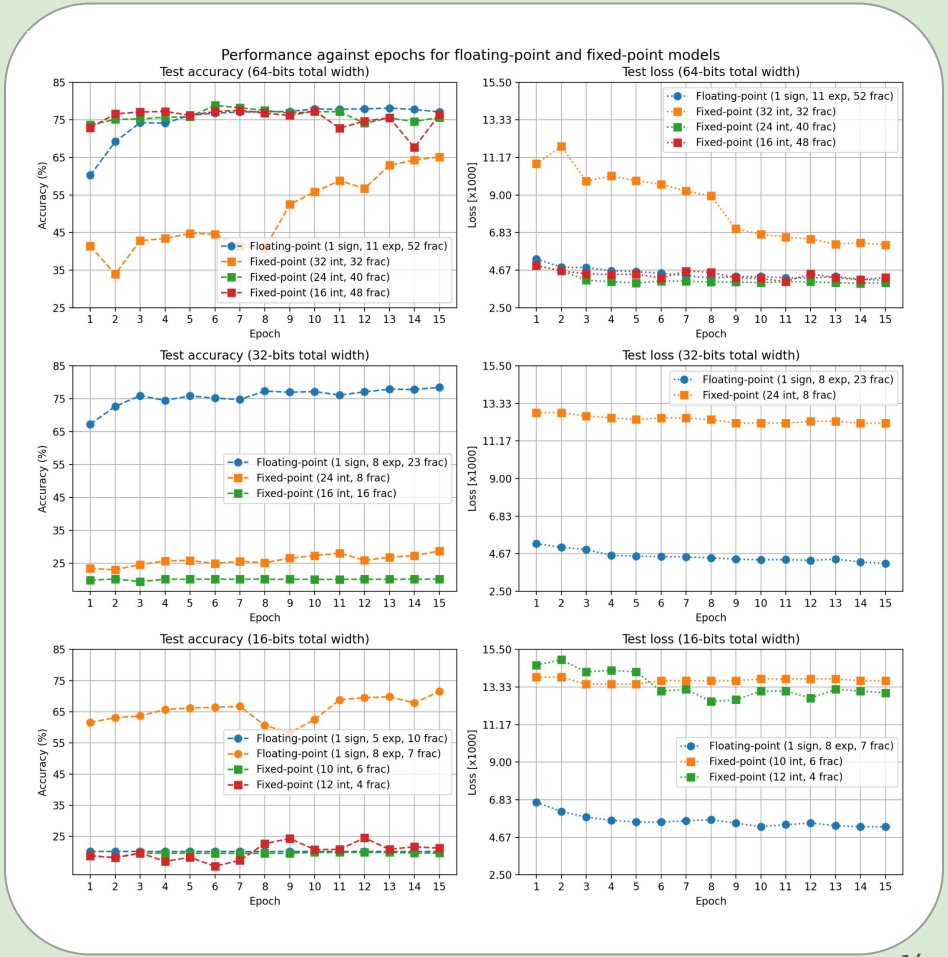
- Integer and fractional parts vary significantly
- Certain layer-dependant patterns exist
- Infeasible to reproduce manually



IMPERIAL

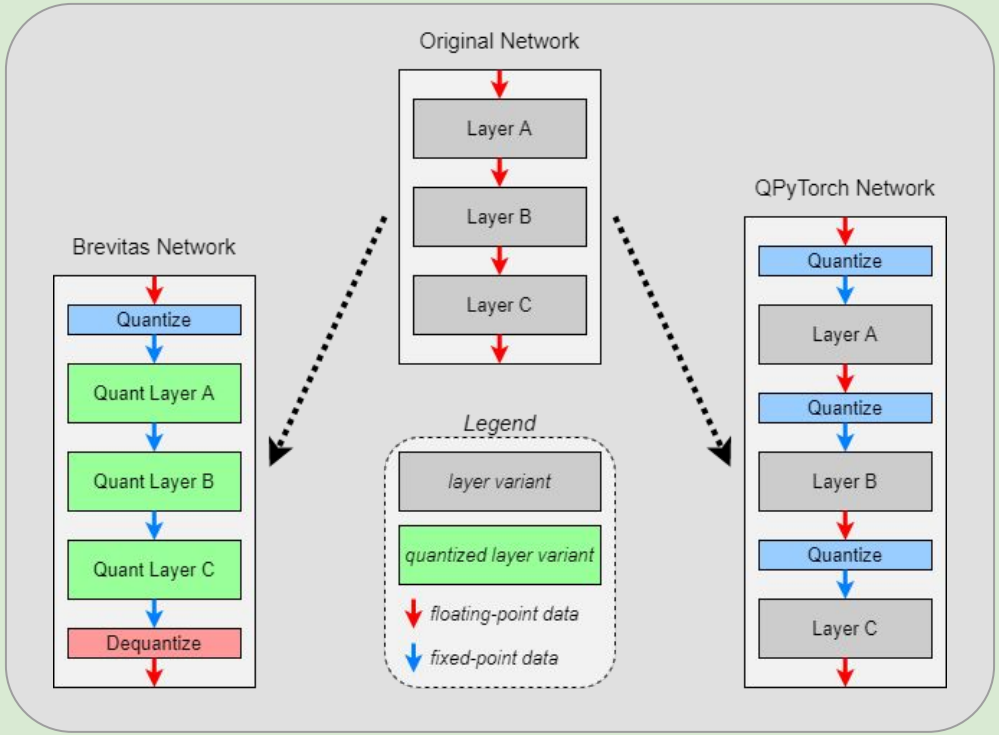
Quantization-Aware Training

- Fixed-point representation on FPGAs
- Existing frameworks:
  - PyTorch Eager Mode Quantization
  - PyTorch FX Graph Mode Quantization
  - Brevitas
  - **QPyTorch - best choice for novelties**



IMPERIAL

Quantization-Aware Training

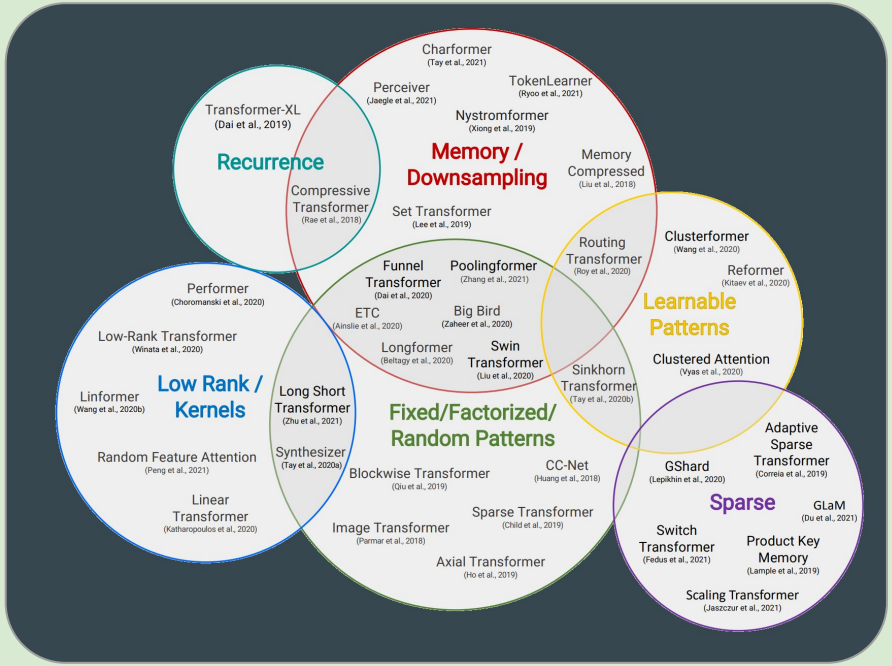


IMPERIAL

Future Work

- Alternative transformer designs [4] to reduce complexity
- Post-training quantization advancements
- Quantization-aware training revaluation

  → Block floats (e.g. MX datatype)



[4] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. Sep 14, 2020. URL https://arxiv.org/abs/2009.06732.

Conclusions

- Transformer neural network architecture exploration in software and hardware

  → **Novel TNN optimizations (general & problem-specific)**

- Accurate, low-latency model optimized for FPGAs

  → **Nanosecond latency, outperforms state-of-the-art**

- Post-training quantization algorithm

  → **Customizable and easy-to-use. Significantly reduced required bit-widths**

Thank you!

Original paper (FPT)

Project repo on GitHub

Filip Wojcicki
filip.wojcicki18@imperial.ac.uk

IMPERIAL