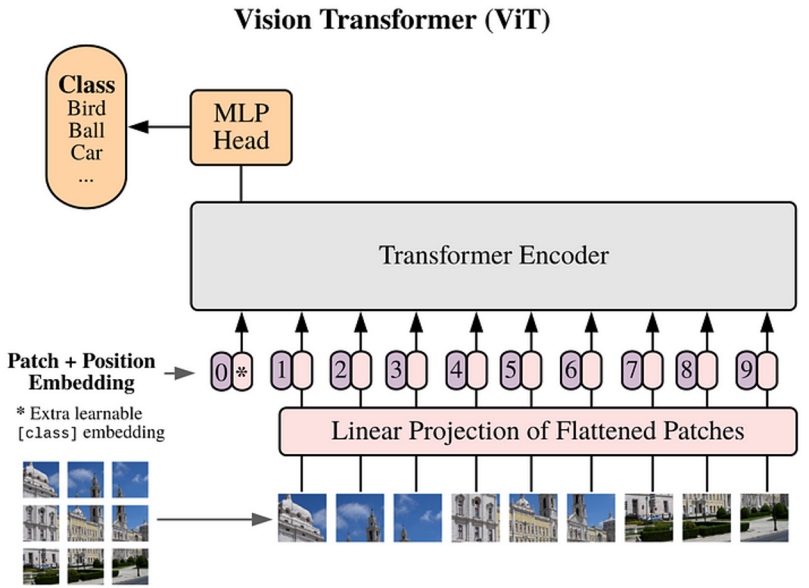
Efficient Vision Transformer Inference using CPU+FPGA

Group 15

Problem

- Model: Vision Transformers
- Vision transformers are computationally intensive
  - Long time
  - Memory bottlenecks & data intensive
- Limitation?
  - Slow if done serially
- Target Task?
  - Image recognition / CV via transformer
- Data?
  - MNIST



Model Architecture

Input Image: typically resized

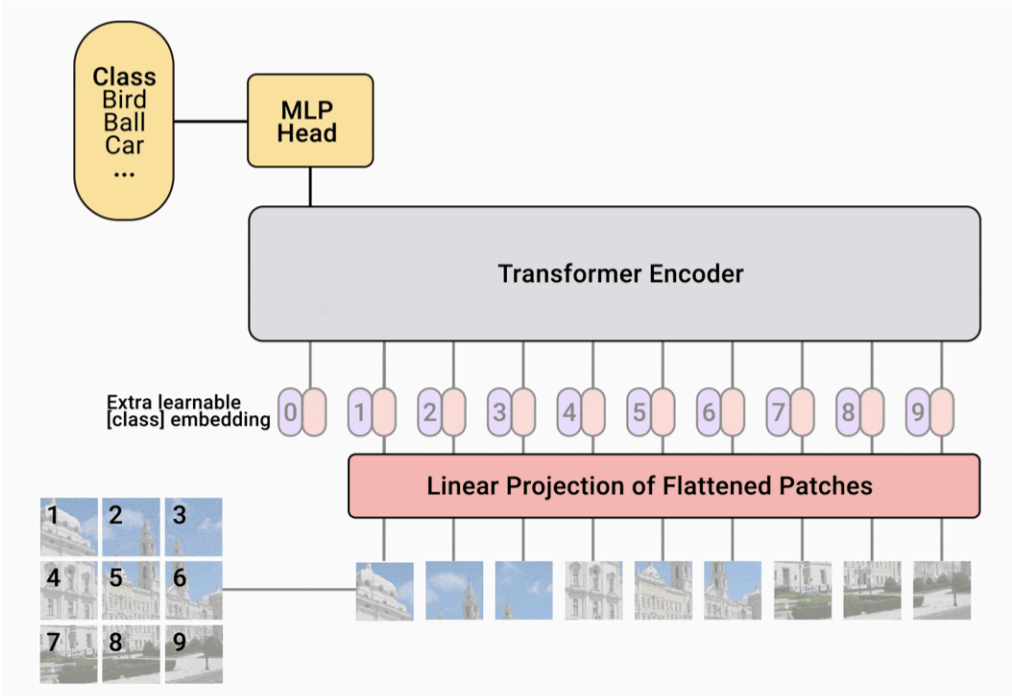
Patch Extraction: patches are usually non-overlapping and of a fixed size.

Embedding: Each patch embedded into a high-dimensional space using a linear projection.

Positional Encoding: positional encoding is added, encodes the location of the patch

Transformer Encoder: multiple layers of self-attention and feedforward neural networks

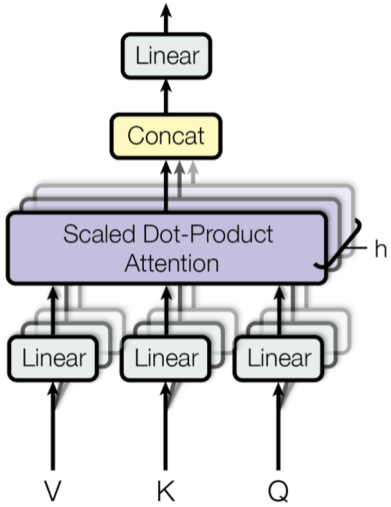
Classification Head: classifies image



Limitations

$$Attention(XW_q, XW_k, XW_v) = Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \in \mathbb{R}^{N \times d_v}$$

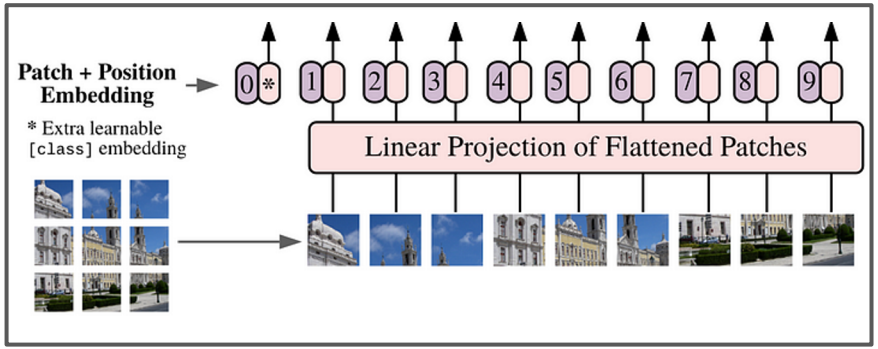
- ➔ Complexity: $\mathcal{O}(N^2 d)$, where N = number of patches, d = dimension of latent vectors each patches are projected to.
- ➔ Scale **quadratically** with the input length (image size).
- ➔ Requires more data to learn due to a higher complexity compared to CNN.[1]



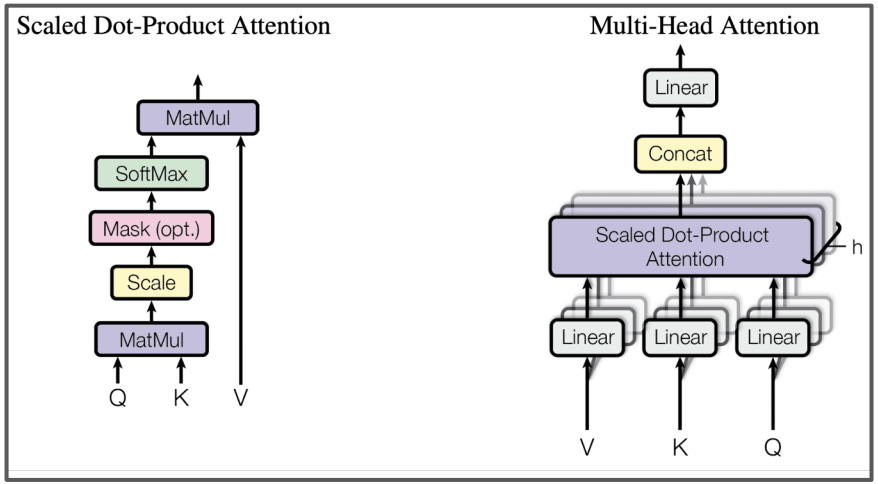
[1]: https://arxiv.org/pdf/2101.11986.pdf

Justification

- ➔ Model is highly parallelizable.
  1. Multi-head Attention module
  2. Patch embedding + Positional Encoding



- ➔ Model mainly consists of **fully connected layers** (matmul) and **attention modules** (scaled dot product). This allows us to specifically design custom kernel in FPGA to deal with these operations.

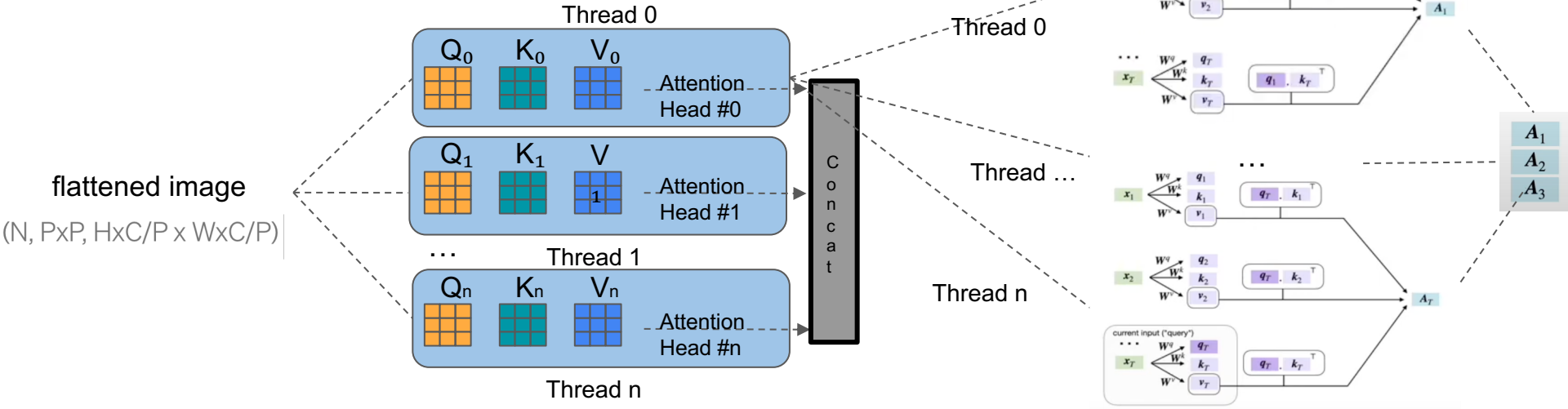


Code parallelization - shared memory parallelization

1. Patchify input & Position Encoding

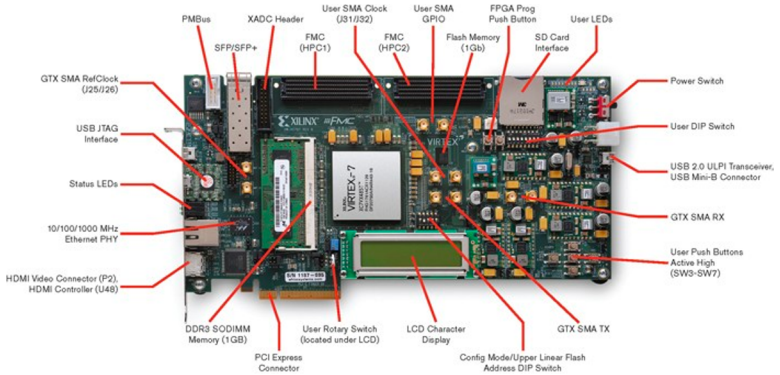
```
for idx, image in enumerate(images):
        for i in range(n_patches):
            for j in range(n_patches):
                patch = image[:, i * patch_size: (i + 1) * patch_size, j * patch_size: (j + 1) *
                patches[idx, i * n_patches + j] = patch.flatten()
```

2. Multi-head attention & Self-attention mechanism

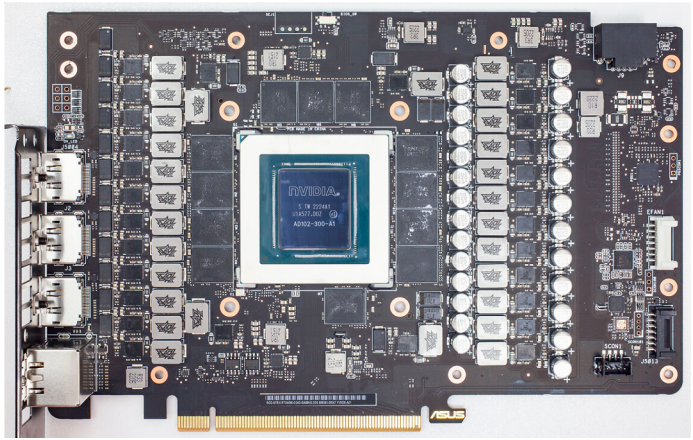


FPGA utilization

- FPGA
  - Hardware prototyping device
  - Host offloads computation intensive work to device (FPGA / GPU)
- matrix multiplication accelerators
  - Systolic Array
  - High computation intensity
  - Energy efficient



Xilinx VC707 FPGA



Nvidia RTX 4090 PCB

References

https://medium.com/mlearning-ai/vision-transformers-from-scratch-pytorch-a-step-by-step-guide-96c3313c2e0c

https://www.youtube.com/watch?v=0PjHri8tc1c

Thank you!