Compressing Trained Neural Networks with Tensor Decompositions

Reservoir Labs

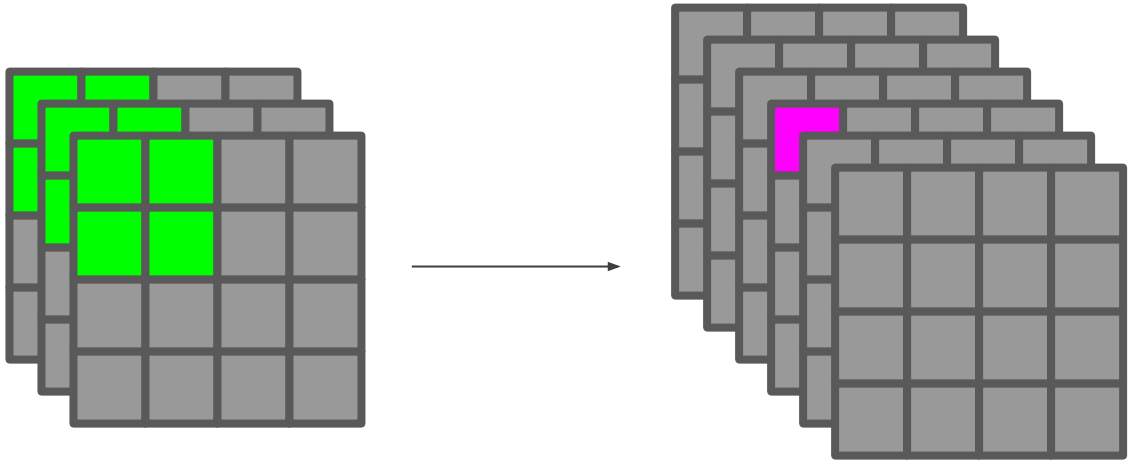
Dimitri Leggas
Muthu Baskaran
James Ezick
Brendan von Hofe

AI at the Edge

- Devices with low computational power, CPUs and low-end GPUs, need to run state-of-the-art models for real-time real-world applications
  - Camera auto-focus
  - Reducing background noise
  - Summarizing news
- Compressing models via tensor decomposition decreases number of parameters while maintaining high accuracy

Convolution

Input *U* has *S* channels



Output *V* has *T* channels

$$V(:,:,t) = \sum_{s=0}^{S-1} K(:,:,s,t) * U(:,:,s)$$

$$\therefore V(x,y,t) = \sum_{s=0}^{S-1} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} K(i,j,s,t) U(x+i, y+j, s)$$

CP Tensor Decomposition

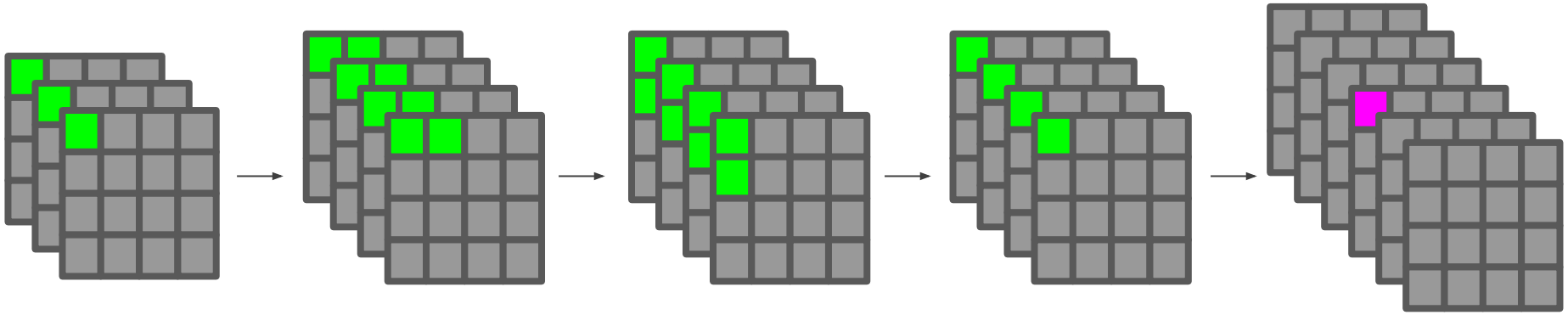
- Approximate 4D kernel tensor as a sum of outer products of appropriate vectors

$$K \approx \sum_{r=1}^{R} K_r^x \circ K_r^y \circ K_r^S \circ K_r^T$$

- Vectors for each dimension form factor matrices

$$K^x, K^y \in \mathbb{R}^{d \times R}, K^S \in \mathbb{R}^{S \times R}, K^T \in \mathbb{R}^{T \times R}$$

Decomposed Convolution



Convolve input along channels

Convolve along *y*-axis

Convolve along *x*-axis

Convolve along channels

Output

$$V(x,y,t) = \sum_{r=0}^{R-1} K^T(t,r) \left(\sum_{i=0}^{d-1} K^x(i,r) \left(\sum_{j=0}^{d-1} K^y(j,r) \left(\sum_{s=0}^{S-1} K^S(s,r) U(x+i, y+j, s) \right) \right) \right)$$

"Speeding-up Convolutional Neural Networks Using Fine-tuned Tensor Decompositions". Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. *International Conference on Learning Representations* 2015.

Layer Compression

- Consider a convolutional layer with a 3x3 kernels mapping 32 input channels to 64 output channels
  - Original layer has 3x3x32x64 = 18,432 parameters
  - Decomposing the layer with rank 16 results in 16x(3+3+32+64) = 1,632 parameters
  - New layer requires **<9%** the parameters and multiplications of the original

Method

- **Decompose:** Reduce number of parameters via CP decomposition and construct factorized layers
- **Fine-tuning:** Perform a few epochs of end-to-end training
- **Pruning:** Set low-magnitude weights to zero during a few more epochs of training

Key Result

- **Compressed ResNet-50:** Decompose each convolutional kernel with rank 32. The resulting model is 9% the size of the original.
  - The decomposed model loses 32.7% of original accuracy
  - After fine-tuning, the decomposed model loses only 1.8% of the original accuracy
  - After magnitude pruning, the decomposed model loses only 0.3% of the original accuracy

ENSIGN Jupyter Notebook

In [7]:
```
%%capture

conv_layers = []
for i, l in enumerate(resnet.layers):
    if type(l) == layers.Conv2D:
        conv_layers.append(i)

rank = 32
fact_resnet = factorized_cnn(resnet, rank, conv_layers,
                        directory='{}_fact_data_all'.format(model_dir))

fact_input = layers.Input(shape=(32, 32, 3,))
x = fact_resnet(fact_input)
for l in model.layers[1:]: # re-use classifier for decomposed ResNet
    x = l(x)
fact_model = models.Model(inputs=[fact_input], outputs=[x])
for l in fact_model.layers:
    l.trainable = True
```

Available at: https://reservoir-ensign.github.io/usecases/cnn.html