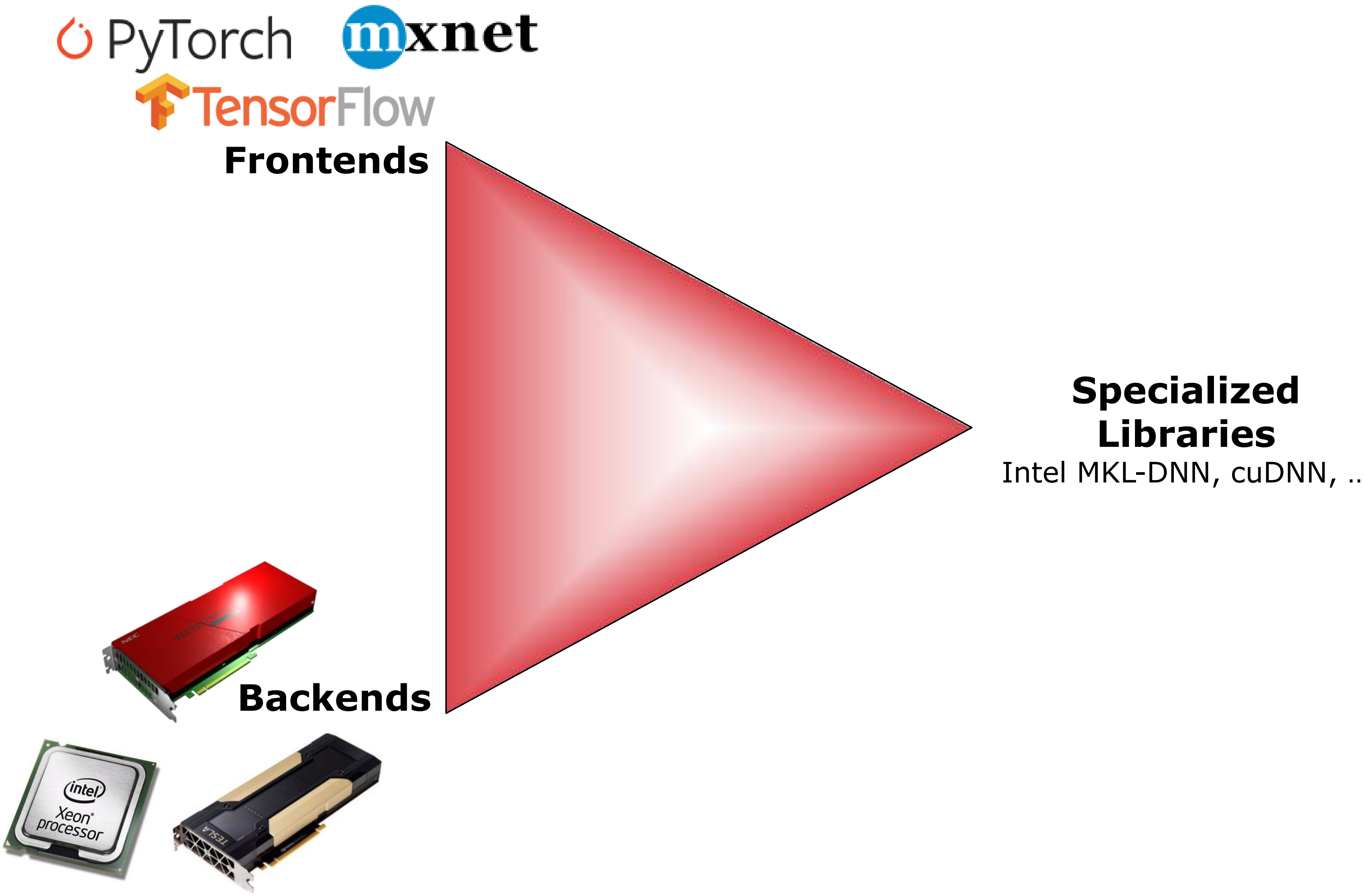
Orchestrating a brighter world NEC

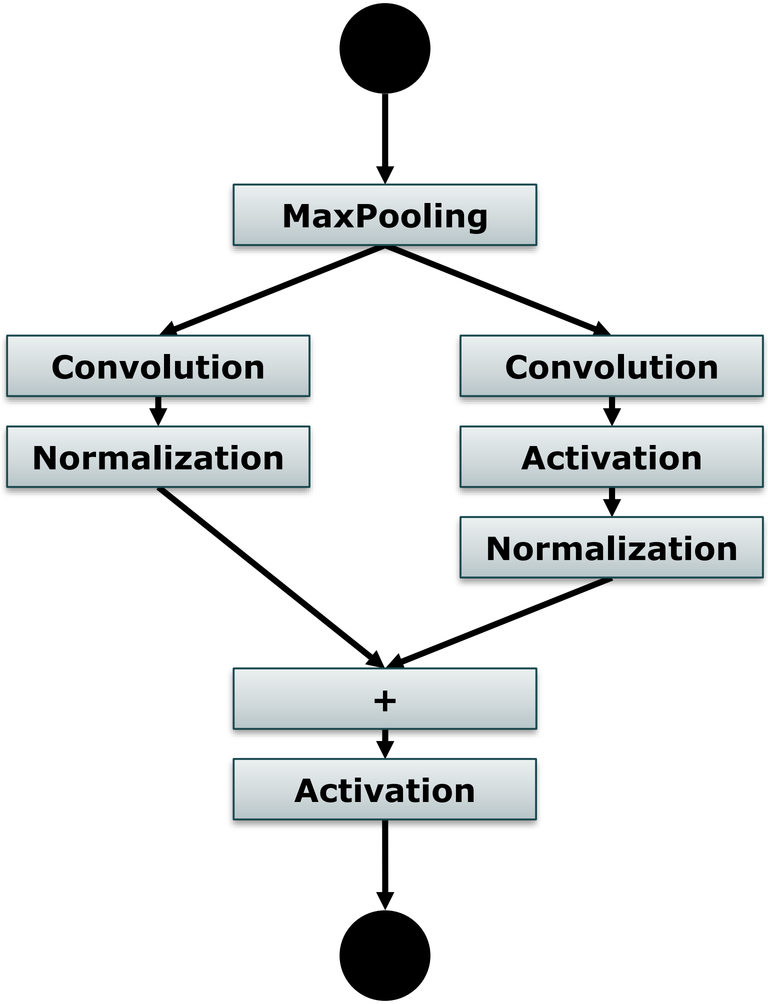
Sol: Transparent Neural Network Acceleration Platform

Nicolas Weber
nicolas.weber@neclab.eu
NEC Laboratories Europe

AI Frameworks

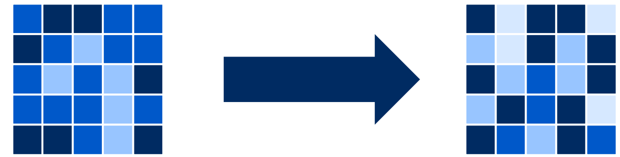


Neural Network Basics

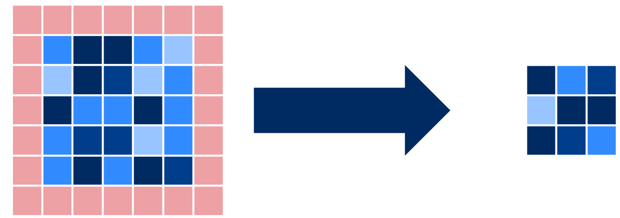


Neural Network Layer Types

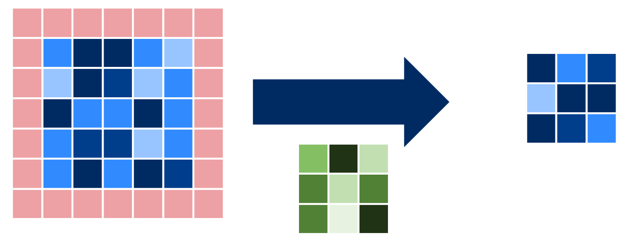
- **Element-wise (e.g. activation):**
  - ReLU, Sigmoid, BatchNorm*, …



- **I/O Memory Bound (e.g. pooling):**
  - Avg-, Max-Pooling, Subsampling, …

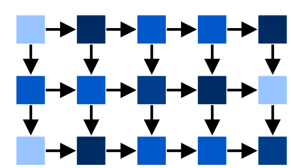


- **Computational Bound (e.g. conv):**

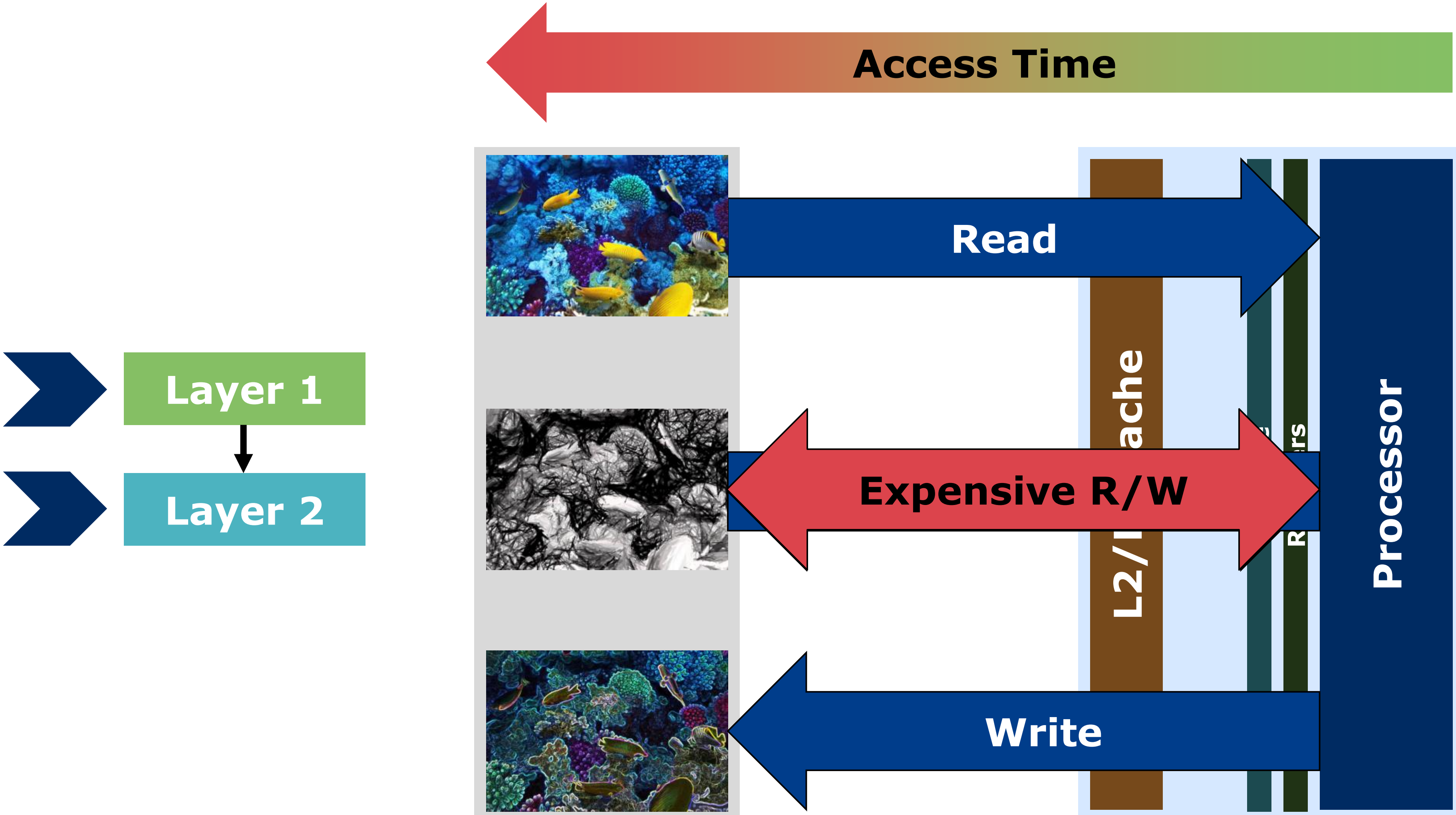


- **Parameter Memory Bound (e.g. dense):**

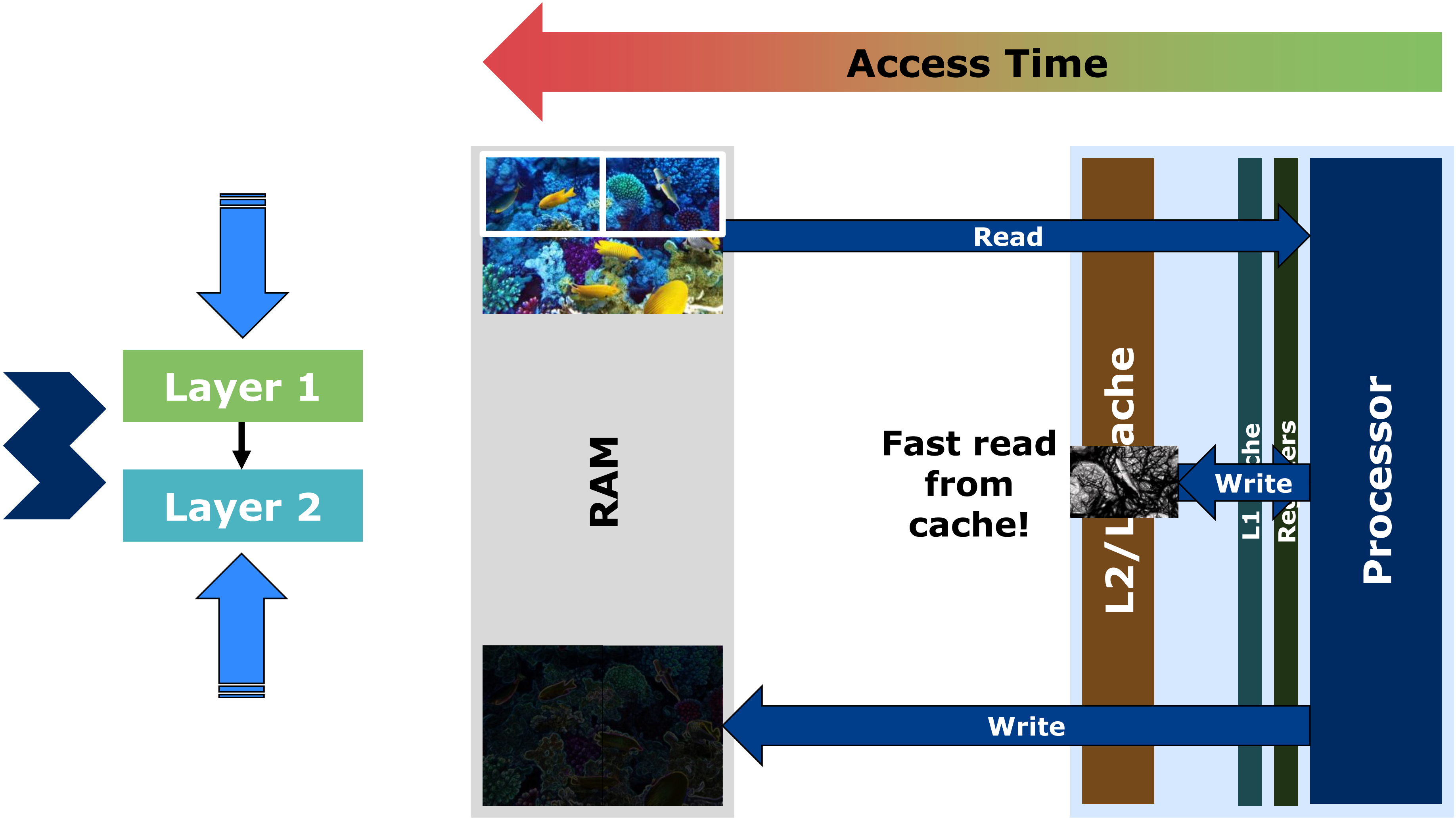
- **Execution order constrained (e.g. RNN):**



Traditional Neural Network Processing



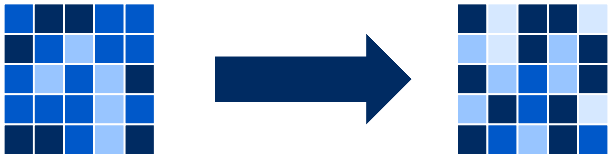
Depth-First Parallelism



Neural Network Layer Types

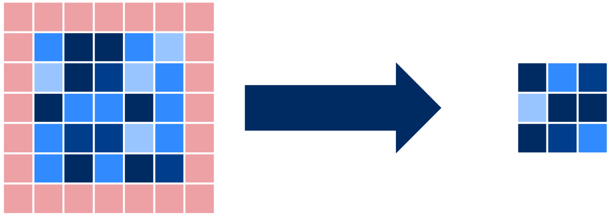
Element-wise (e.g. activation):

- ReLU, Sigmoid, BatchNorm*, …

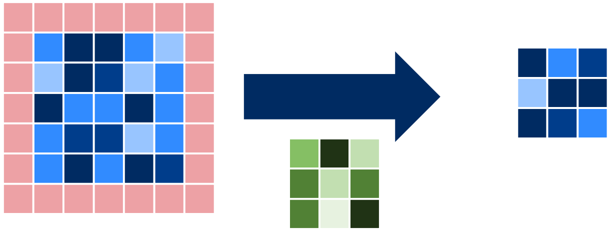


I/O Memory Bound (e.g. pooling):

- Avg-, Max-Pooling, Subsampling, …

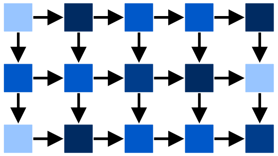


Computational Bound (e.g. conv):



Parameter Memory Bound (e.g. dense):

Execution order constrained (e.g. RNN):



Code Generation

```
__global__ void F64486B08(...) {
  const int O0idx = blockIdx.x;
  const int O0 = O0idx / 256;
  const int O1 = O0idx % 256;
  __shared__ float T64[169];
  for(int O2idx = threadIdx.x; O2Idx < 169; O2Idx += 128) {
    float T63 = 0.0f;
    for(int I1 = 0; I1 < 512; I1++)         // #1 Convolution: 1x1 Pooling
      T63 += T61[O0 * 86528 + I1 * 169 + O2idx] * P63_weight[O1 * 512 + I1];
    T63 = (T63 + P63_bias[O1]);              // #1 Convolution: Bias
    T64[O2Idx] = fmaxf(T63, 0.0f);           // #2 ReLU
  }
  T66[O1] = REDUCE_ADD(T64);                 // #3 AvgPooling: 13x13 Pooling
  T66[O1] = (T66[O1] / 169.0f);              // #3 AvgPooling: Normalization
}
```

CUDA blocks

shared memory

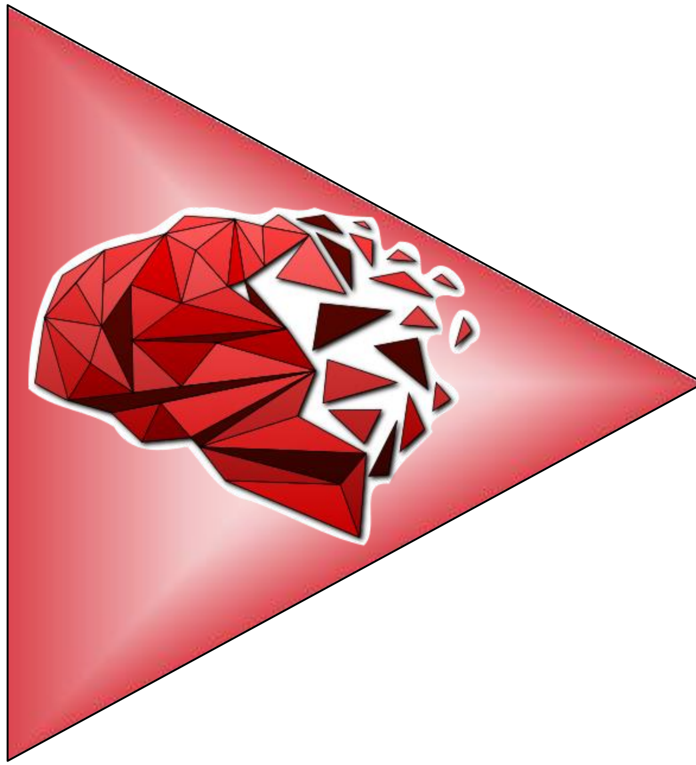
CUDA cores

Reduction

Sol Platform

PyTorch mxnet
TensorFlow

Frontends

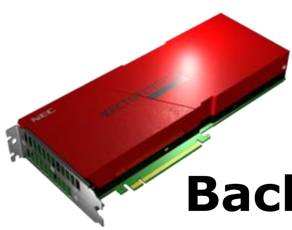


Optimizations

Graph Transformations

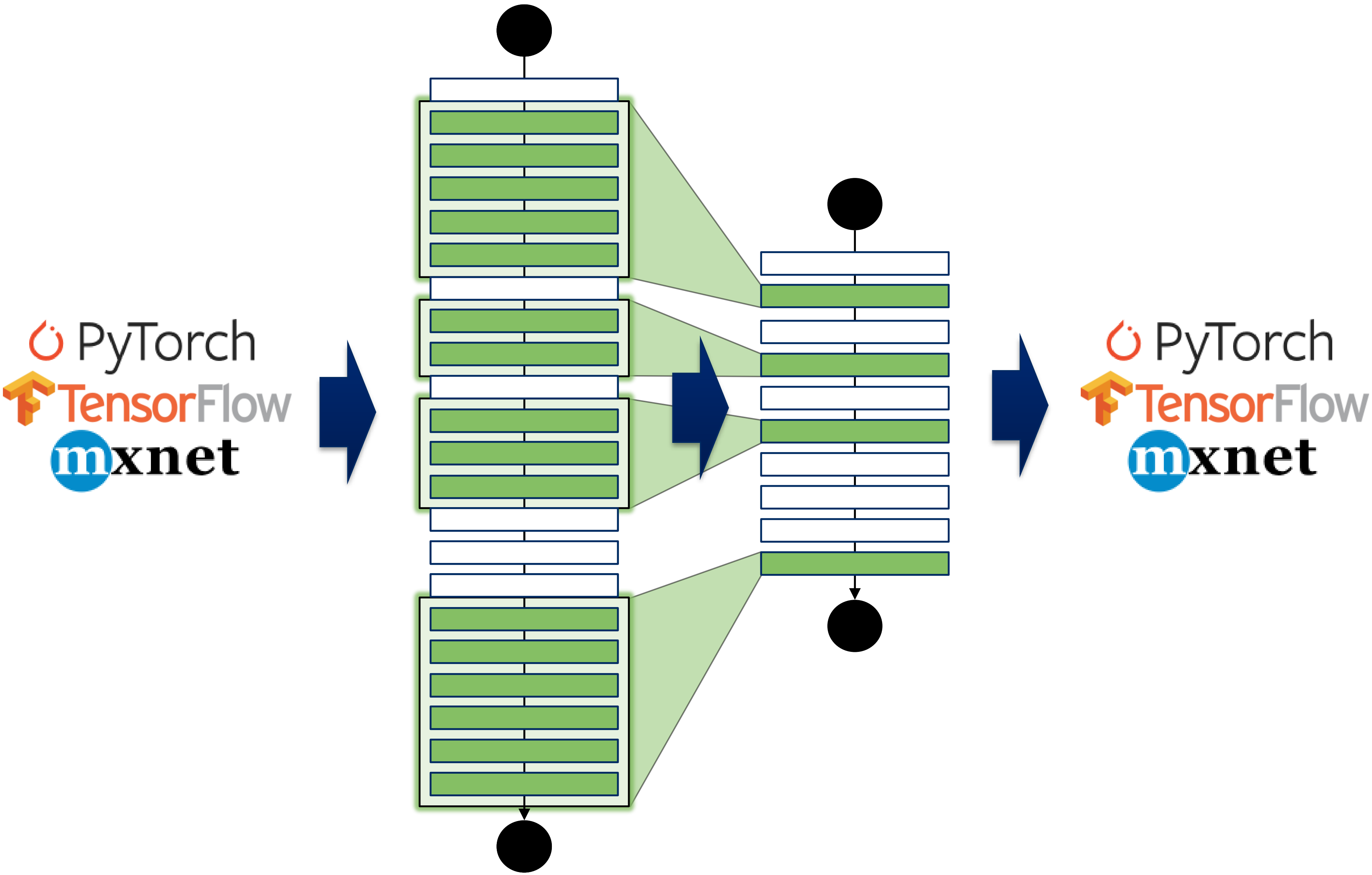
Depth-First-Parallelism
I/O bound + element-wise

Specialized Libraries
Computational + Parameter bound (e.g. Intel MKL-DNN, cuDNN,…)



Backends

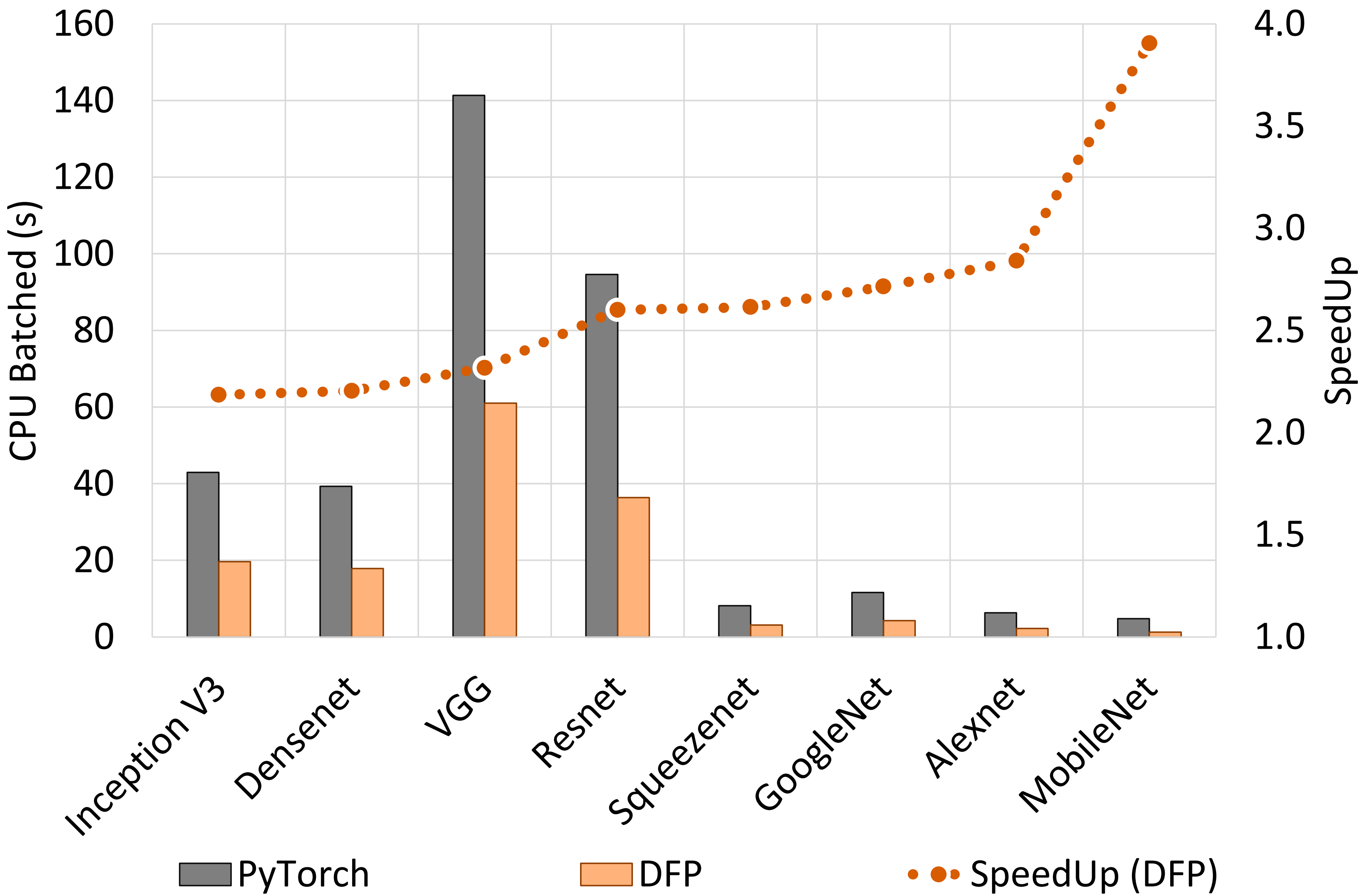
Network Analysis



Sol Usage (PyTorch)

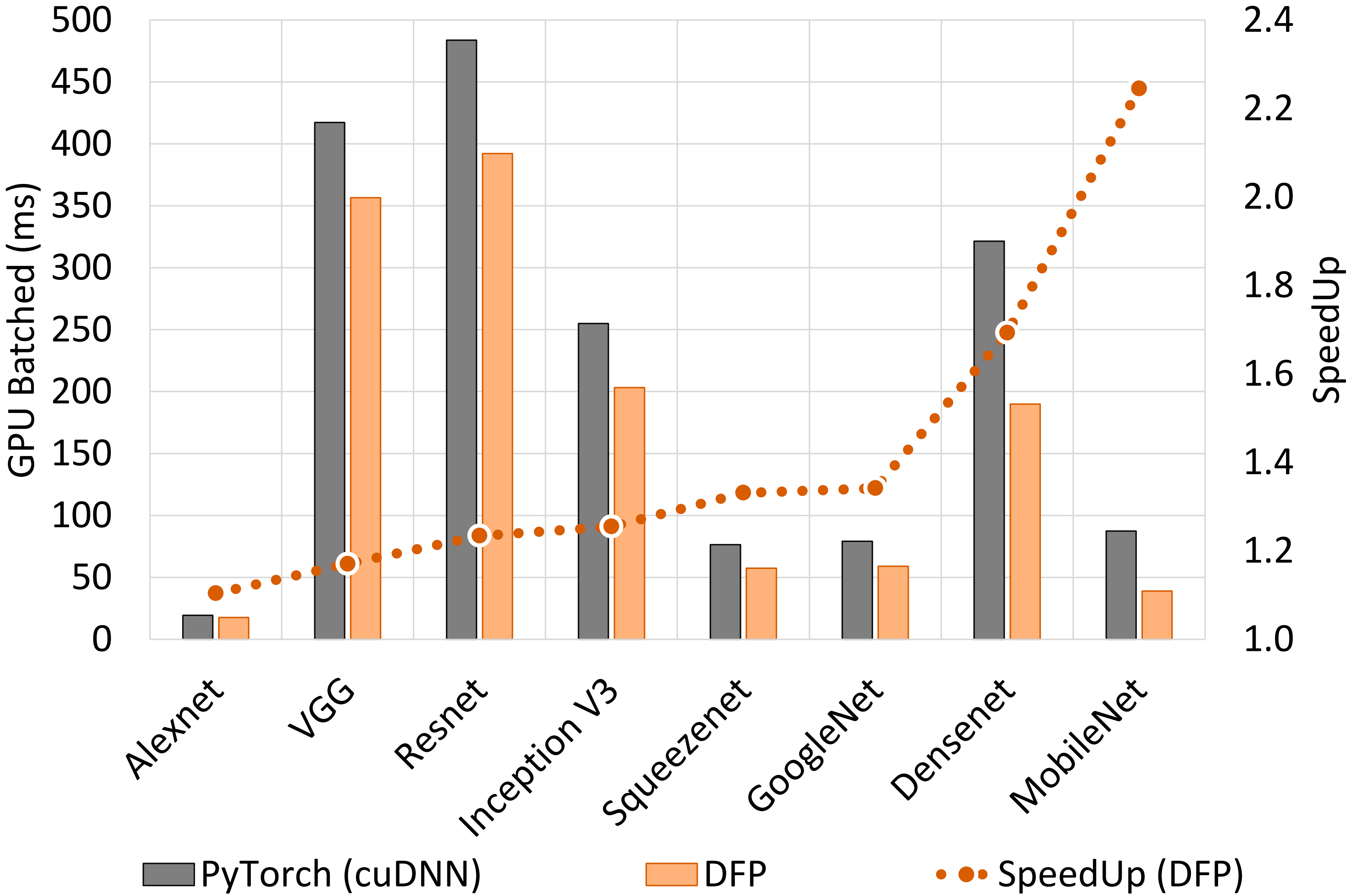
```
import torch
from torch.autograd import Variable
from torchvision import models
from sol.pytorch import optimize

model  = models.__dict__["…"]()
input  = torch.rand(32, 32, 224, 224)
model  = optimize(model, input.size())
output = model(input)
```

Performance: PyTorch v0.4.8, CPU, Batched-128



\Orchestrating a brighter world NEC

Performance: PyTorch v0.4.8, GPU, Batched-128



Lambda

Results summary

As of October 8, 2018, the NVIDIA RTX 2080 Ti is the best GPU for deep learning research on a single GPU system running TensorFlow. A typical single GPU system with this GPU will be:

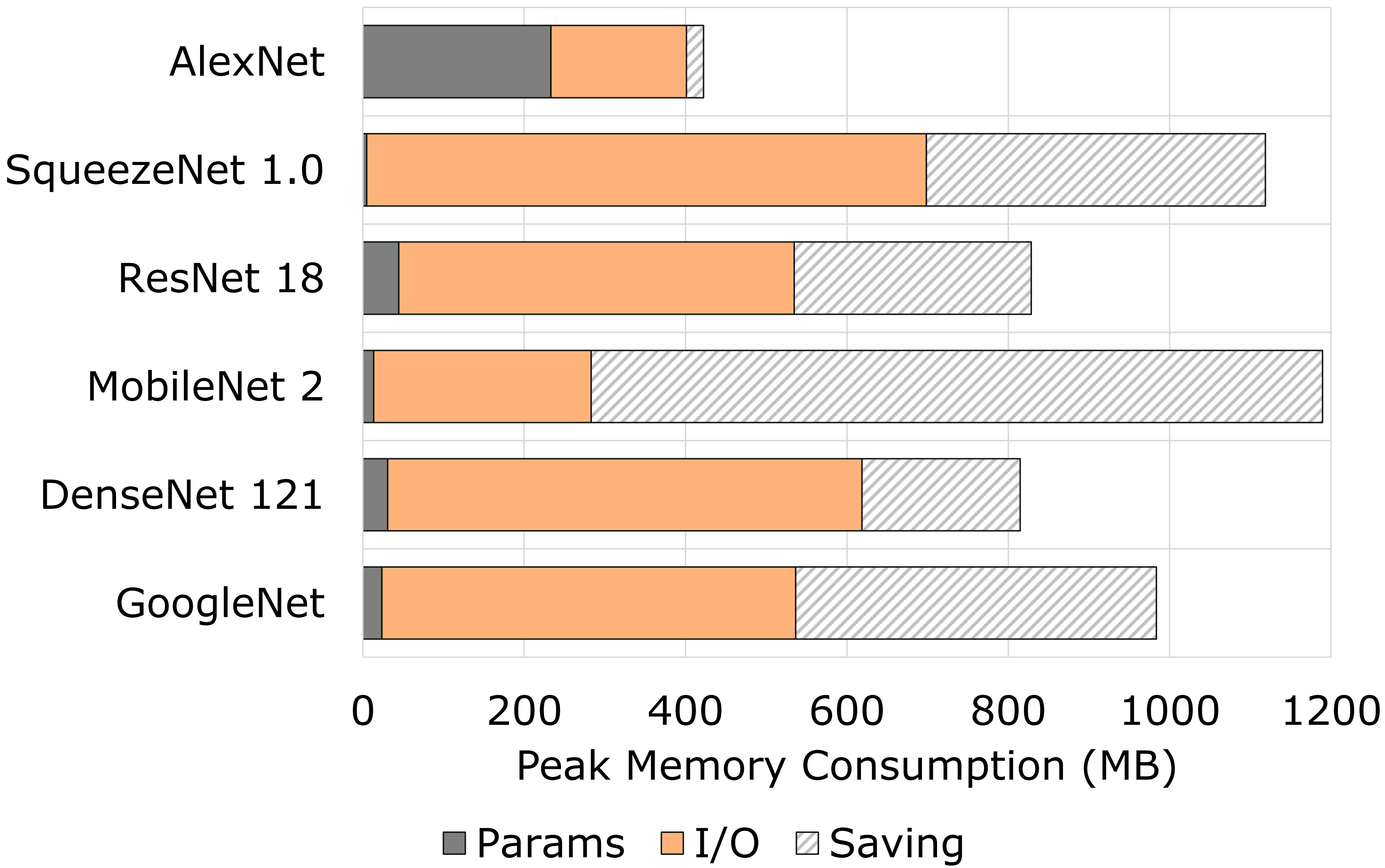
- **37% faster than the 1080 Ti** with FP32, 62% faster with FP16, and 25% more expensive.

Sol on average 36.2% faster on GTX 1080 Ti than PyTorch

- Performance of tomorrow already on todays hardware!

https://lambdalabs.com/blog/best-gpu-tensorflow-2080-ti-vs-v100-vs-titan-v-vs-1080-ti-benchmark/

Memory Saving (Batched Prediction-128)



Roadmap

Next Milestone (1st quarter 2019):

- **Frontends**: PyTorch, TensorFlow and CNTK
- **Backends**: X86, CUDA and NEC SX-Aurora TSUBASA
- Inference and Training support

Future Work

- Specialized optimizations for more NN layer types (e.g. RNNs)
- Inference deployment (e.g. Linux library, Unikernel, …)
- More device support (e.g. ARM-CPU/-GPU, AMD-GPU, …)
- Automatic tuning of internal parameters, memory formats, …

Sol: Transparent Neural Network Accelerator

Nicolas Weber

nicolas.weber@neclab.eu
NEC Laboratories Europe

SC Poster Reception, Ballroom C2/3/4
Tuesday, November 13th 5:15pm-7pm

BrainSlug: Transparent Acceleration of Deep Learning Through Depth-First Parallelism
Nicolas Weber, Florian Schmidt, Mathias Niepert and Felipe Huici
https://arxiv.org/abs/1804.08378