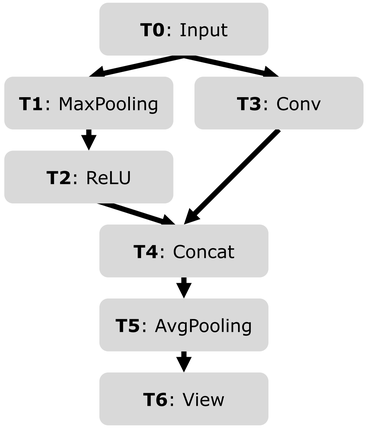
SOL: Transparent Neural Network Acceleration Platform

NEC Laboratories Europe – Nicolas Weber (**nicolas.weber@neclab.eu**)

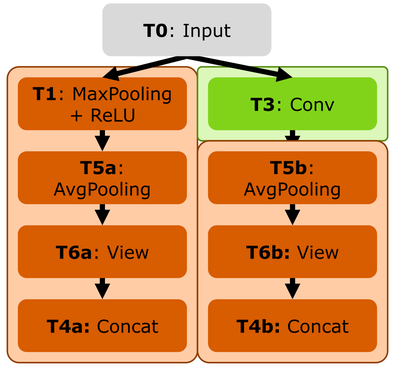
NN Frameworks

```
foreach(batch, channel, y, x):
  T1[…] = -inf
  foreach(ky, kx):
    T1[…] = max(T1[…], T0[…])

foreach(batch, channel, y, x):
  T2[…] = max(T1[…], 0.0f);

foreach(batch, channel, y, x):
  T3[…] = 0.0f
  foreach(inChannel, ky, kx):
    T3[…] += T0[…] * T3_weight[…]
  T3[…] += T3_bias[…]

foreach(batch):
  memcpy(T4[…], T2[…], …)
  memcpy(T4[…], T3[…], …)

foreach(batch, channel, y, x):
  T5[…] = 0.0f
  foreach(ky, kx):
    T5[…] += T4[…]
  T5[…] /= |ky| * |kx|
```

Neural Network



NN Structure Optimization



Generic Optimized Code

```
# optimized nested loops
foreach(batch, channel):
  foreach(y, x):
    # merged MaxPooling + ReLU
    T1[…] = 0.0f
    foreach(ky, kx):
      T1[…] = max(T1[…], T0[…])

  T5a = 0.0f
  foreach(ky, kx):
    T5a += T1[…]
  # offsetted access for concat data
  T4[… + offset] = T5a / (|ky| * |kx|)

foreach(batch, channel, y, x):
  T3[…] = 0.0f
  foreach(inChannel, ky, kx):
    T3[…] += T0[…] * T3_weight[…]
  T3[…] += T3_bias[…]

# optimized nested loops
foreach(batch, channel):
    T5b = 0.0f
    foreach(ky, kx):
      T5b += T3[…]
    T4[… + offset] = T5b / (|ky| * |kx|)
```

Device Specific Optimized Code

```
__global__ void kernel(…) {
  int warp    = threadIdx.x / 32; // warp level parallelism
  int warpIdx = threadIdx.x % 32;
  int batch   = blockIdx.x;
  int channel = blockIdx.y * 4 + warp;

  __shared__ float T1[4][…]; // shared memory
  // MaxPooling + ReLU
  for(int tx = warpIdx; tx < (|y| * |x|); tx += 32) {
    int x = tx % |x|, y = ty / |x|;
    T1[warp][…] = 0.0f;
    for(int ky = …)
      for(int kx = …)
        T1[warp][…] = max(T1[warp][…], T0[…]);
  }

  // AvgPooling + Concat
  int T5a = 0.0f;
  for(int tx = warpIdx; tx < (|ky| * |kx|); tx += 32)
    T5a += T1[warp][tx];
  T5a += __shfl_down_sync(…); // warp level reduction
  if(warpIdx == 0)
    T4[… + offset] = T5a / (|ky| * |kx|);
}
```

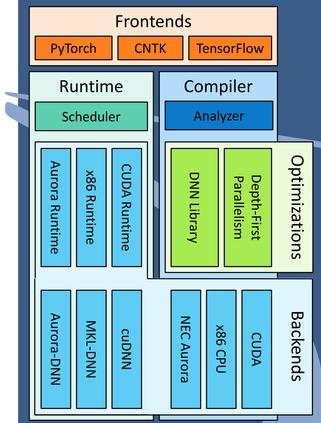
Sol Usage

Using Sol only requires two additional lines of code!

```
# import packages
import torch
from torchvision import models
import sol.pytorch as sol

# init network
size = [0, 3, 224, 224]
myNN = models.__dict__["resnet18"]()

# optimize network using Sol
optNN = sol.optimize(myNN, size)

# run network
X = torch.rand(128, *size[1:])
Y = optNN(X)
```

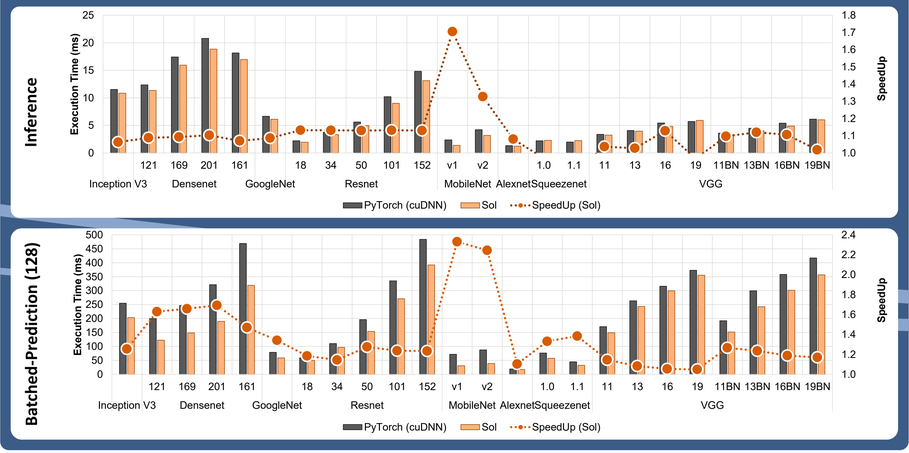
default NN Framework approach → **Sol Neural Network Optimization**

Plugin Architecture

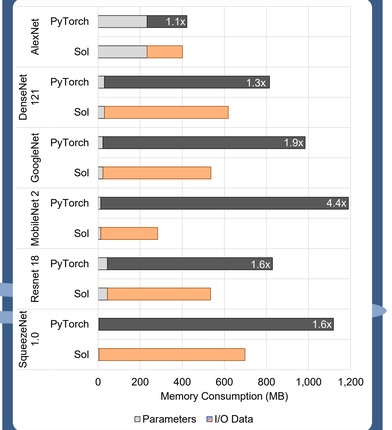


CPU: 2x Intel Xeon E5-2397 v4

GPU: NVIDIA GTX 1080 TI



Memory Consumption



sysml.neclab.eu

Orchestrating a brighter world

NEC