Paper: Continual 3D Convolutional Neural Networks for Real-time Processing of Videos

3D CNNs are **inefficient** at online video stream processing. **Continual 3D CNNs fix this.**

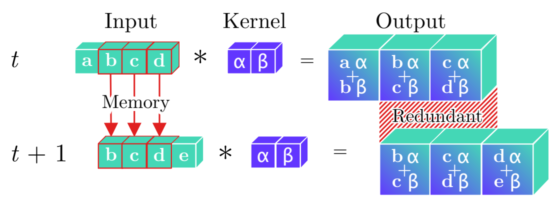
Lukas Hedegaard & Alexandros Iosifidis
Department of Electrical and Computer Engineering
Aarhus University, Denmark

{lhm, ai}@ece.au.dk

linkedin.com/in/lukashedegaard/

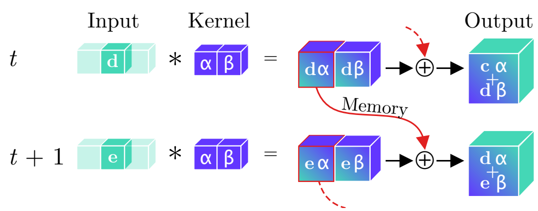
github.com/lukashedegaard/co3d
github.com/lukashedegaard/continual-inference

3D CNNs are ~~inefficient~~ but accurate at video stream processing



Conv: Per-frame online predictions yield overlapping computation

Continual **3D CNNs** have no redundancy



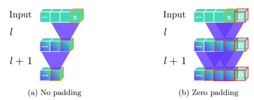
CoConv: Intra-convolutional features for a frame are *cached now* and *aggregated later*

How to make *pretrained* 3D CNNs ***efficient:***

1. Replace ~~Conv~~ with ***CoConv***
2. Remove ~~temporal padding~~
3. **Delay** residuals

```
def coconv3d(frame, prev_state = (mem, i)):
    frame = spatial_padding(frame)
    frame = temporal_padding(frame)
    feat = conv3d(frame, weights)
    output, rest_feat = feat[0], feat[1:]
    mem, i = prev_state or init_state(output)
    M = len(mem)
    for m in range(M):
        output += mem[(i + m) % M, M - m - 1]
    output += bias
    mem[i] = rest_feat
    i = (i + 1) % M
    return output, (mem, i)
```

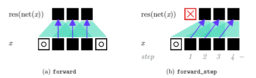
Listing 1.1: **Pseudo-code** for Continual Convolution. Ready-to-use modules are available in the Continual Inference library [15].



(a) No padding (b) Zero padding

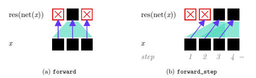
Fig. 4: **Issue with temporal padding**: The latest frame **x** is propagated through a CNN with (purple) temporal kernels of size 3 (a) without or (b) with zero padding. Highlighted cubes can be produced only in the latest frame, with yellow boarder indicating independence of padded zero and red boarders dependence. In the zero-padded case (b), the number of frame features dependent on **x** following a layer l increases with the number of padded zeros.

Hedegaard, L., Iosifidis, A.: Continual Inference: A Library for Efficient Online Inference with Deep Neural Networks in PyTorch (ECCVW 2022)



(a) forward (b) forward_step

Fig. 4: Residual connections ↑ over a module with receptive field of size ▲ and padding one ("equal padding") ⊡. ☒ are empty outputs.



(a) forward (b) forward_step

Fig. 5: Centered residual connections ↑ over a module with receptive field of size ▲ and no padding. ☒ are empty outputs.

Speed-up ∝ receptive field

12.1 – 15.3× FLOPs reduction achieved

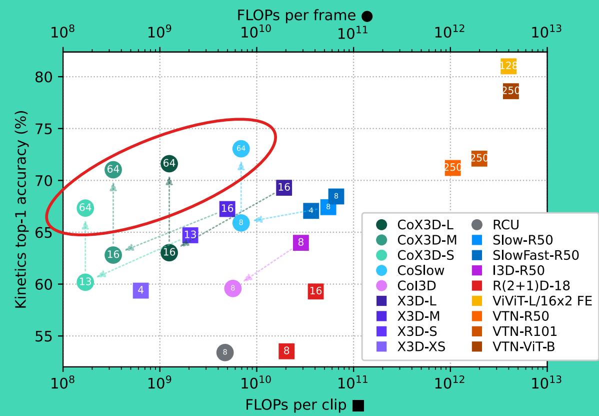
Need less **memory**

Internal state has overhead, but:
Intermediary *feature-maps* size is reduces

Boost **accuracy**

with extended temporal receptive fields

Reuse pretrained weights



AARHUS UNIVERSITY

open Deep Robotics

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449