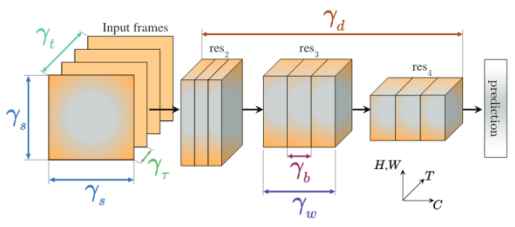
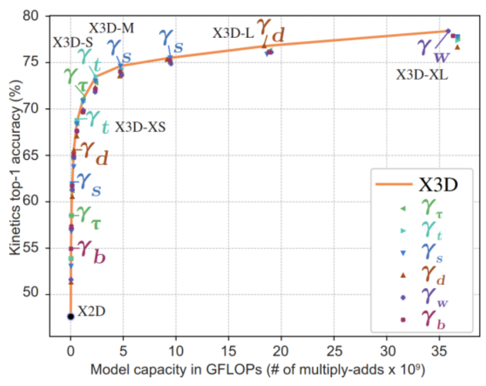
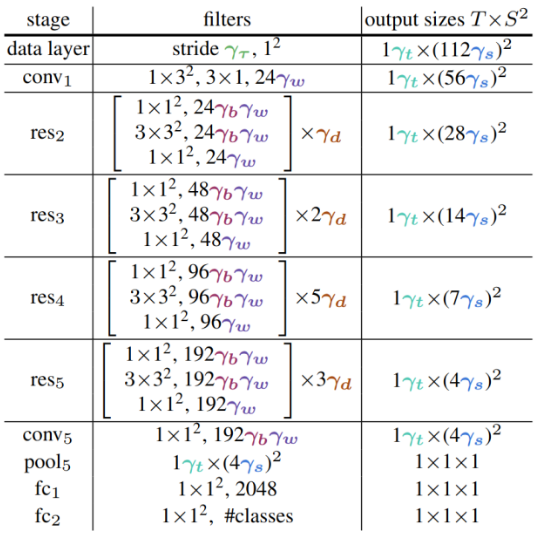
X3D: Expanding Architectures for Efficient Video Recognition

Christoph Feichtenhofer

Starting from a very lightweight model referred to as X2D, which takes as input a single frame with a spatial resolution of 112 x 112, with a conv1 followed by 4 resnet stages with lightweight channels of size [24, 48, 98, 192], and one of each stage contains a minimal number of resnet bottlenecks, ie [1, 2, 5, 3]. Then at each expansion step, one of the 6 possible axes is doubled, the axis that gave the best result is maintained while the rest is reinitialized into their values at the previous step. This is then repeated until a computational budget is met. If the last doubling step exceeds the budget, the selected axis is then reduced by a factor smaller than 2 until the budget is met.







stage	filters	output sizes $T\times S^2$
data layer	stride γ_τ, 1^2	$1\gamma_t\times(112\gamma_s)^2$
conv$_1$	1×3^2, 3×1, $24\gamma_w$	$1\gamma_t\times(56\gamma_s)^2$
res$_2$	$\begin{bmatrix} 1\times1^2, 24\gamma_b\gamma_w \\ 3\times3^2, 24\gamma_b\gamma_w \\ 1\times1^2, 24\gamma_w \end{bmatrix}\times\gamma_d$	$1\gamma_t\times(28\gamma_s)^2$
res$_3$	$\begin{bmatrix} 1\times1^2, 48\gamma_b\gamma_w \\ 3\times3^2, 48\gamma_b\gamma_w \\ 1\times1^2, 48\gamma_w \end{bmatrix}\times2\gamma_d$	$1\gamma_t\times(14\gamma_s)^2$
res$_4$	$\begin{bmatrix} 1\times1^2, 96\gamma_b\gamma_w \\ 3\times3^2, 96\gamma_b\gamma_w \\ 1\times1^2, 96\gamma_w \end{bmatrix}\times5\gamma_d$	$1\gamma_t\times(7\gamma_s)^2$
res$_5$	$\begin{bmatrix} 1\times1^2, 192\gamma_b\gamma_w \\ 3\times3^2, 192\gamma_b\gamma_w \\ 1\times1^2, 192\gamma_w \end{bmatrix}\times3\gamma_d$	$1\gamma_t\times(4\gamma_s)^2$
conv$_5$	1×1^2, $192\gamma_b\gamma_w$	$1\gamma_t\times(4\gamma_s)^2$
pool$_5$	$1\gamma_t\times(4\gamma_s)^2$	$1\times1\times1$
fc$_1$	1×1^2, 2048	$1\times1\times1$
fc$_2$	1×1^2, #classes	$1\times1\times1$

The axes that can be tested at each expansion step are:

- Fast: increasing the frame rate by having a smaller stride when sampling from a given clip/video.
- Temporal: increasing the duration of the sampled clip resulting in larger sampled input frames.
- Spatial: increasing the spatial dimension of the input frames.
- Depth: increasing the number of resnet blocks per stage.
- Width: increasing the number of channels across the model.
- Bottleneck: increasing the channels of the center convolution of the resnet bottleneck block.

By doing such an iterative expansion, starting from the above detailed X2D and choosing the axis with the best top1 accuracy at each step, we have the following results: Bottleneck -> Fast -> Spatial -> Depth -> Temporal -> Fast -> Temporal -> Spatial -> Spatial -> Depth -> Width.