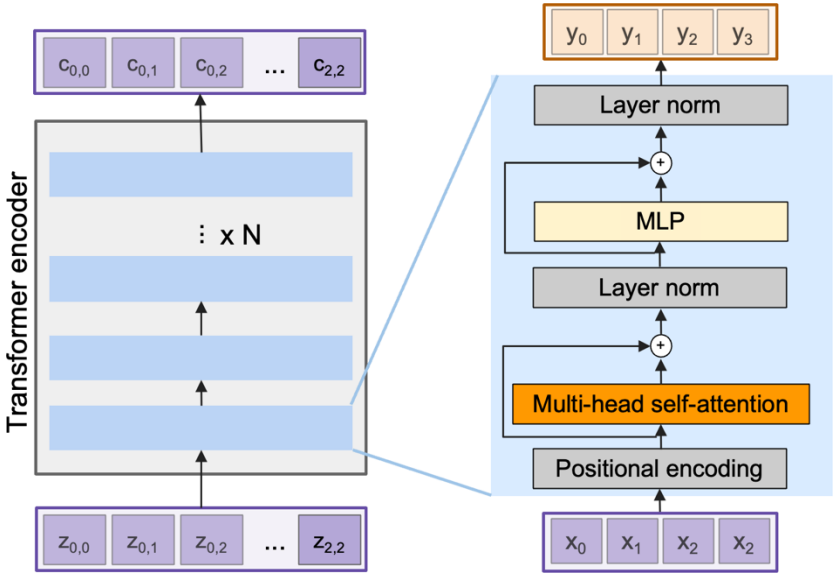
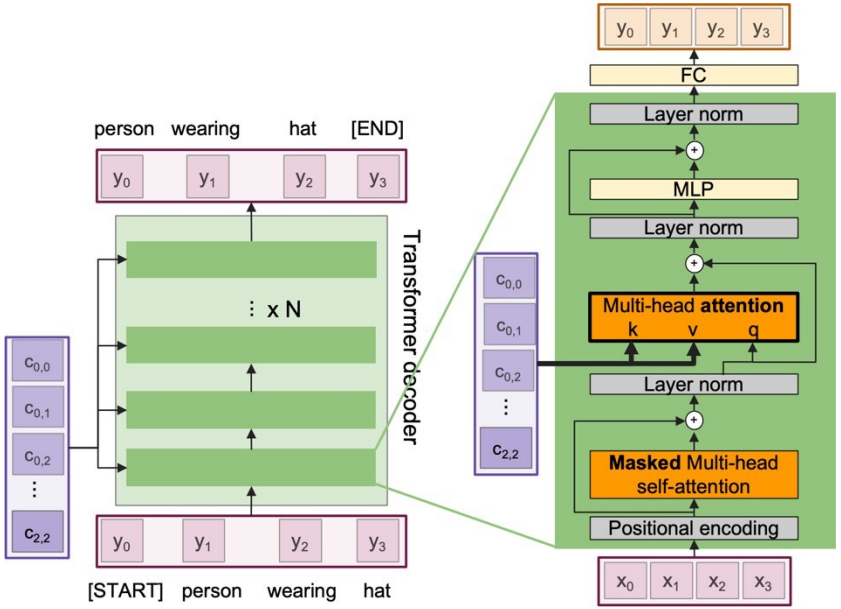
Lecture 9: Detection, Segmentation, Visualization, and Understanding

Last time: Transformer



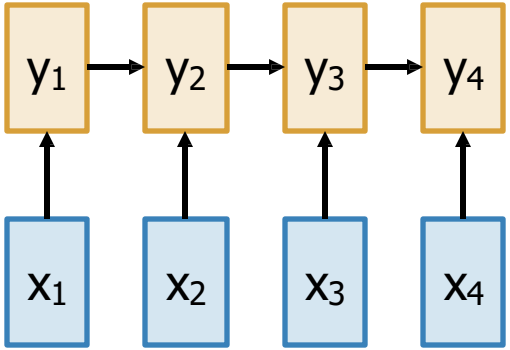
Encoder



Decoder

Three Ways of Processing Sequences

Recurrent Neural Network

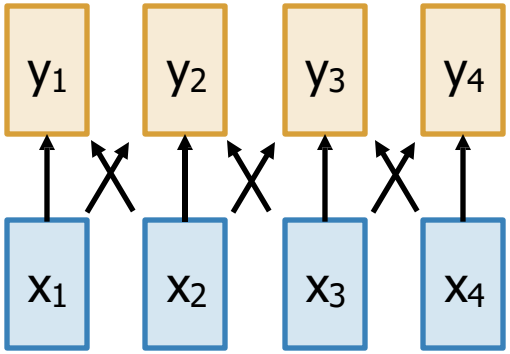


Works on 1D ordered sequences

(+) Theoretically good at long sequences: O(N) compute and memory for a sequence of length N

(-) Not parallelizable. Need to compute hidden states sequentially

Convolution

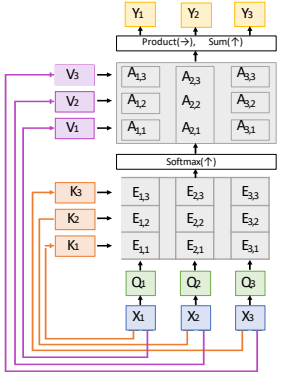


Works on N-dimensional grids

(-) Bad for long sequences: need to stack many layers to build up large receptive fields

(+) Parallelizable, outputs can be computed in parallel

Self-Attention



Works on sets of vectors

(+) Great for long sequences; each output depends directly on all inputs

(+) Highly parallel, it's just 4 matmuls

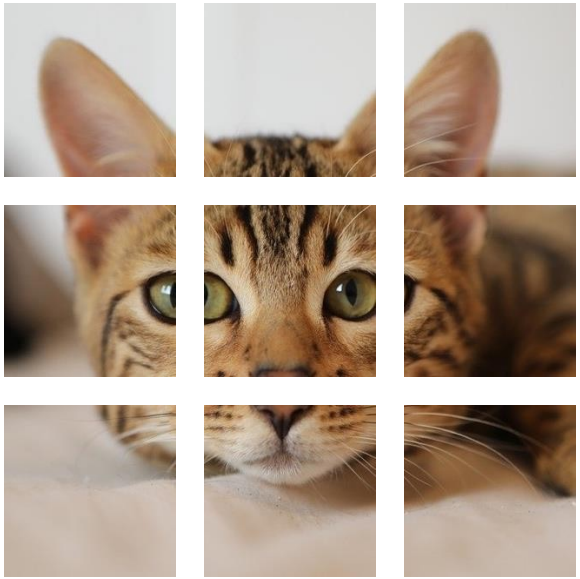
(-) Expensive: $O(N^2)$ compute, $O(N)$ memory for sequence of length N

Vision Transformers (ViT)

Input image:
e.g. 224x224x3

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

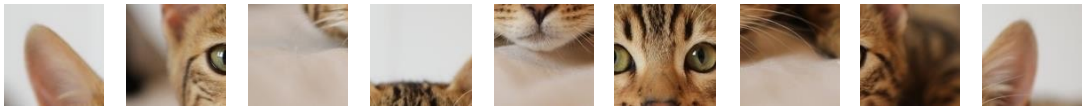
Vision Transformers (ViT)



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

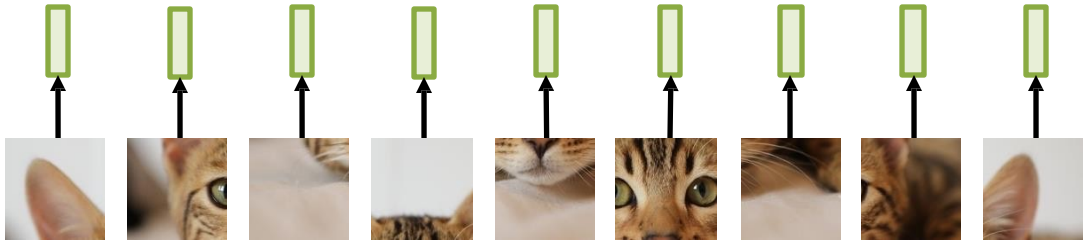
Vision Transformers (ViT)

N input patches, each of shape 3x16x16



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)



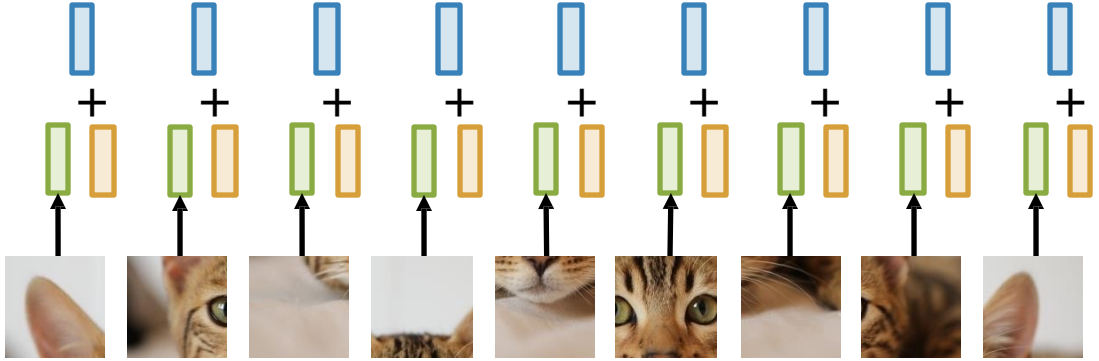
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)

Add positional embedding: learned D-dim vector per position

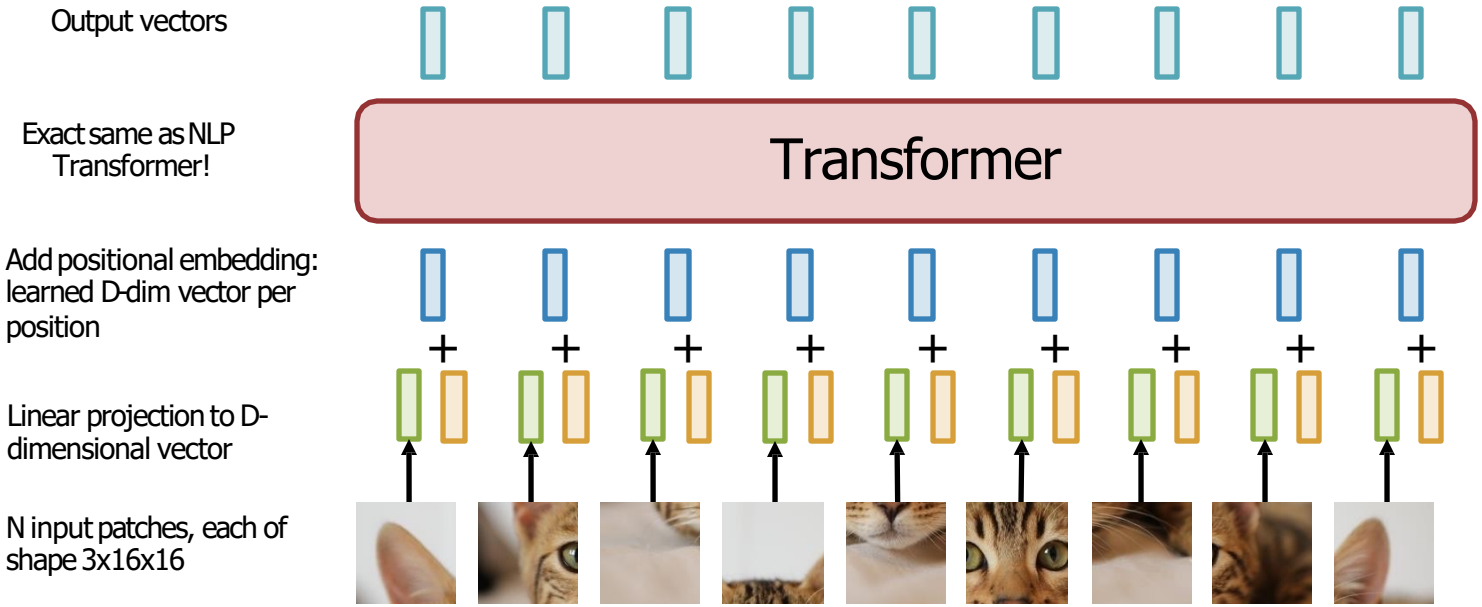
Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



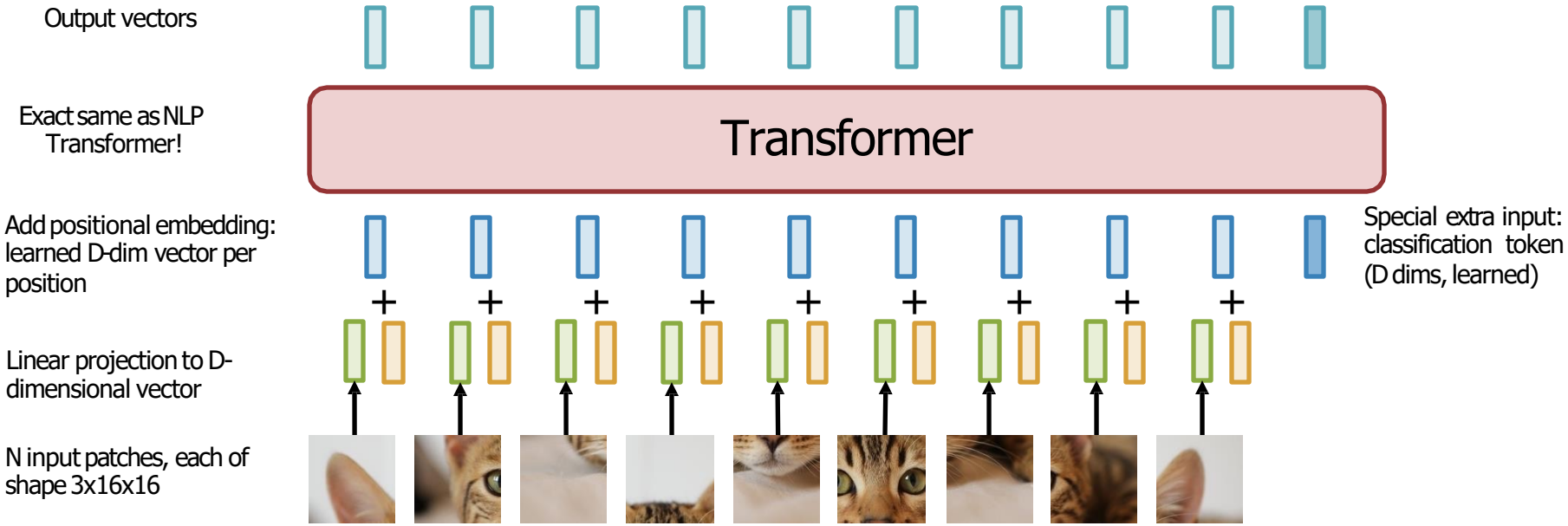
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)



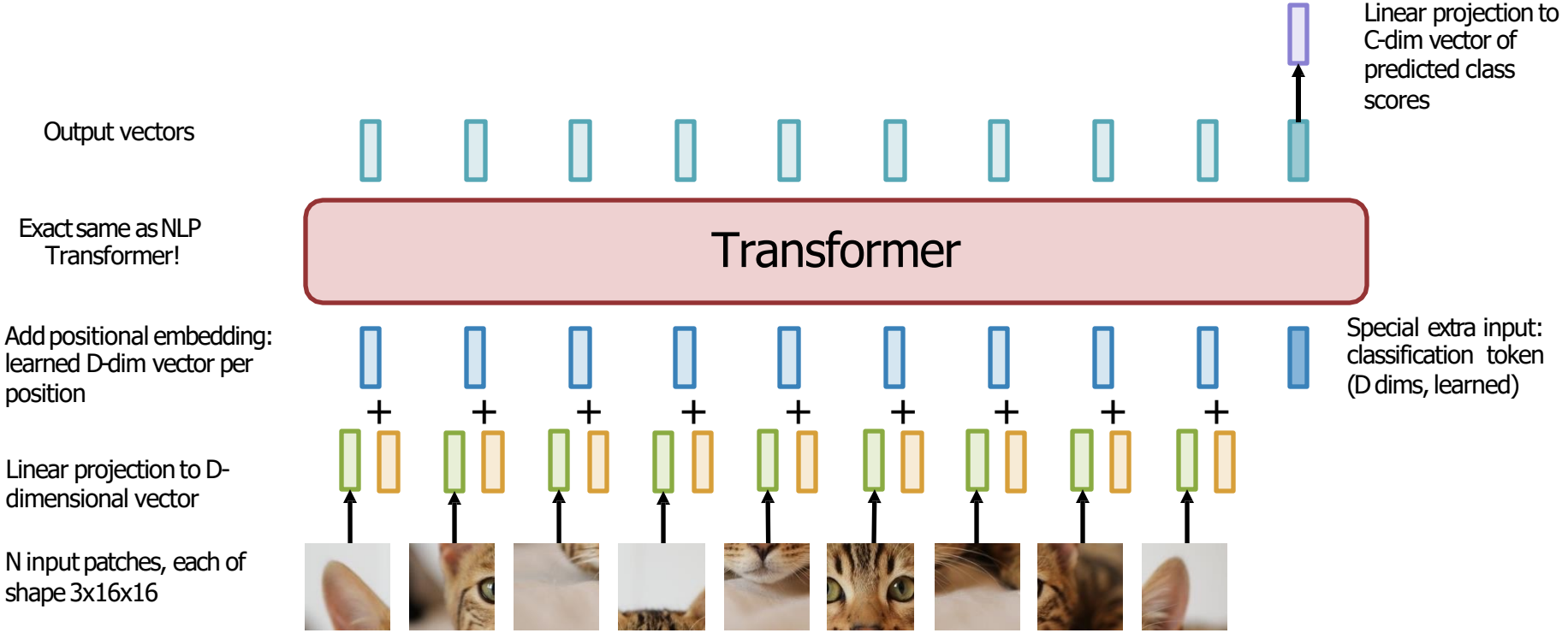
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)



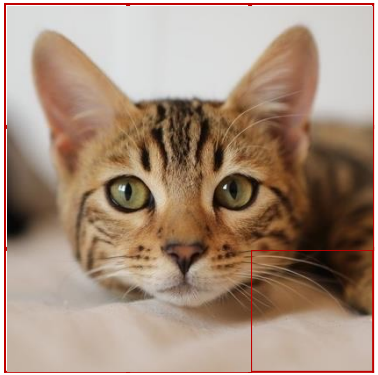
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT) – a similar approach (different classifier)



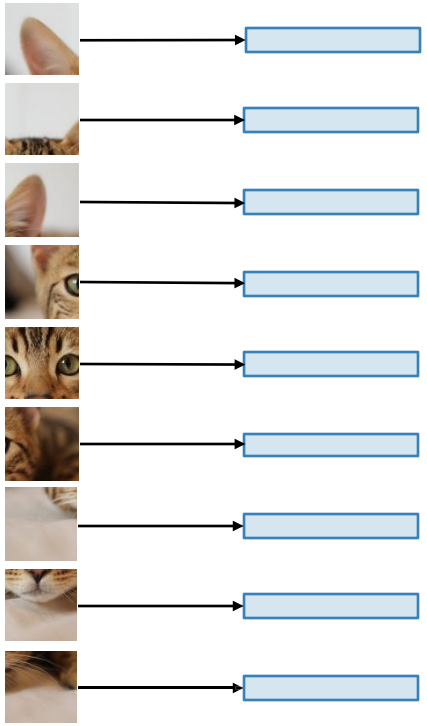
Input image:
e.g. 224x224x3

Break into patches
e.g. 16x16x3

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)

Input image:
e.g. 224x224x3



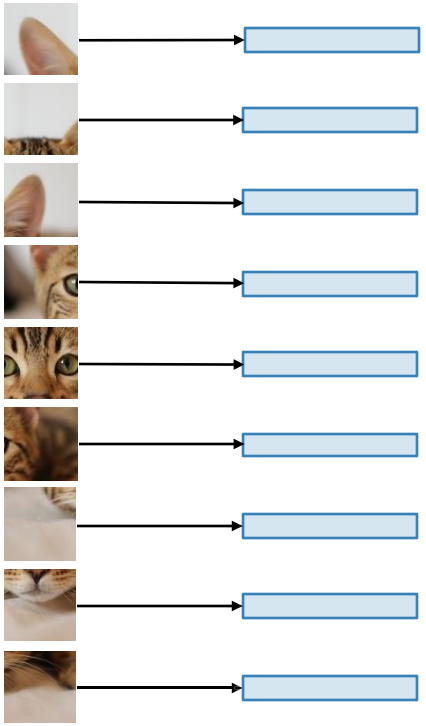
Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

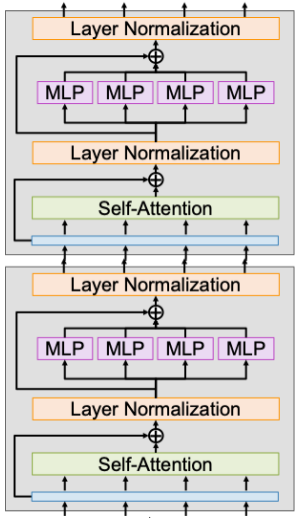
Vision Transformers (ViT)

Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D

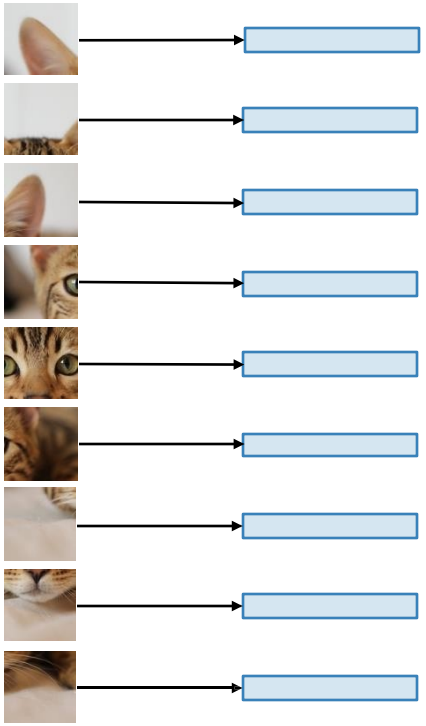


D-dim vector per patch are the input vectors to the Transformer

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

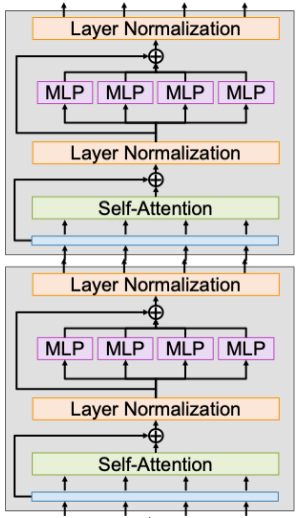
Vision Transformers (ViT)

Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



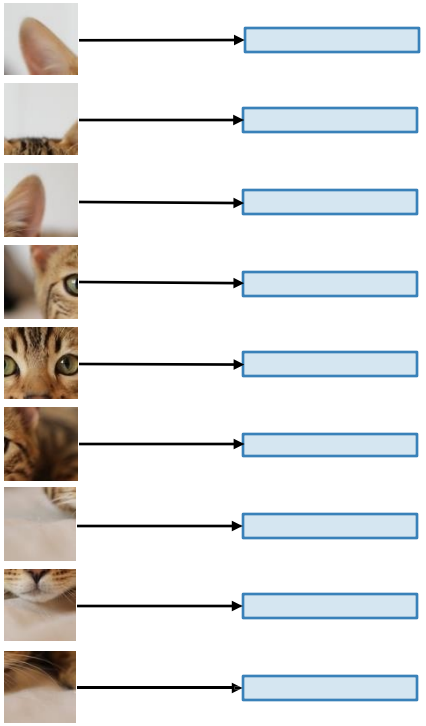
D-dim vector per patch are the input vectors to the Transformer

Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

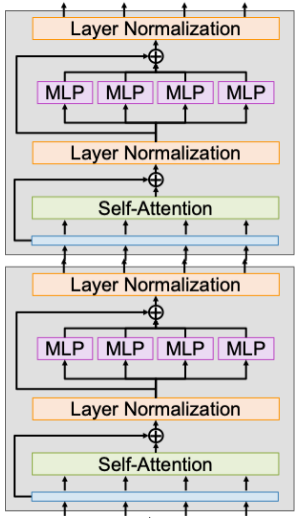
Vision Transformers (ViT)

Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



D-dim vector per patch are the input vectors to the Transformer

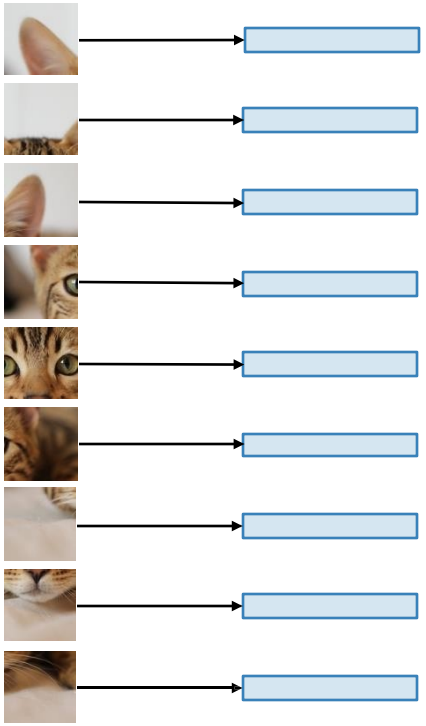
Don't use any masking; each image patch can look at all other image patches

Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

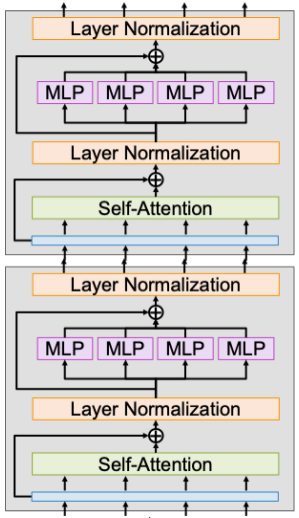
Vision Transformers (ViT)

Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



D-dim vector per patch are the input vectors to the Transformer

Transformer gives an output vector per patch

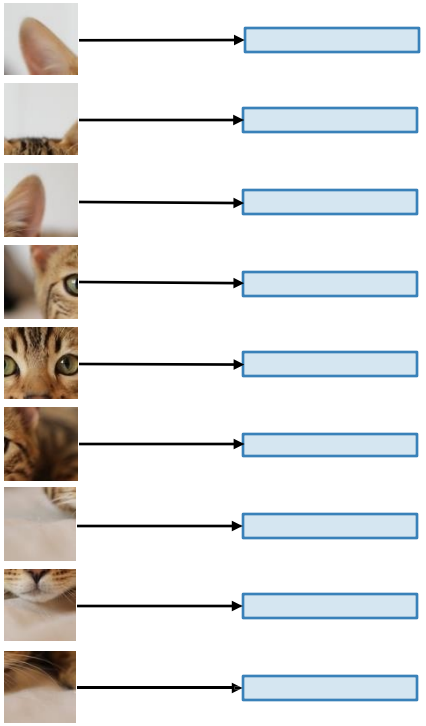
Don't use any masking; each image patch can look at all other image patches

Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)

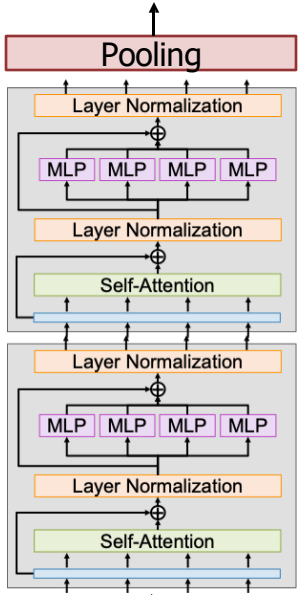
Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 =>D

D-dim vector per patch are the input vectors to the Transformer



Average pool NxD vectors to 1xD, apply a linear layer D=>Cto predict class scores

Transformer gives an output vector per patch

Don't use any masking; each image patch can look at all other image patches

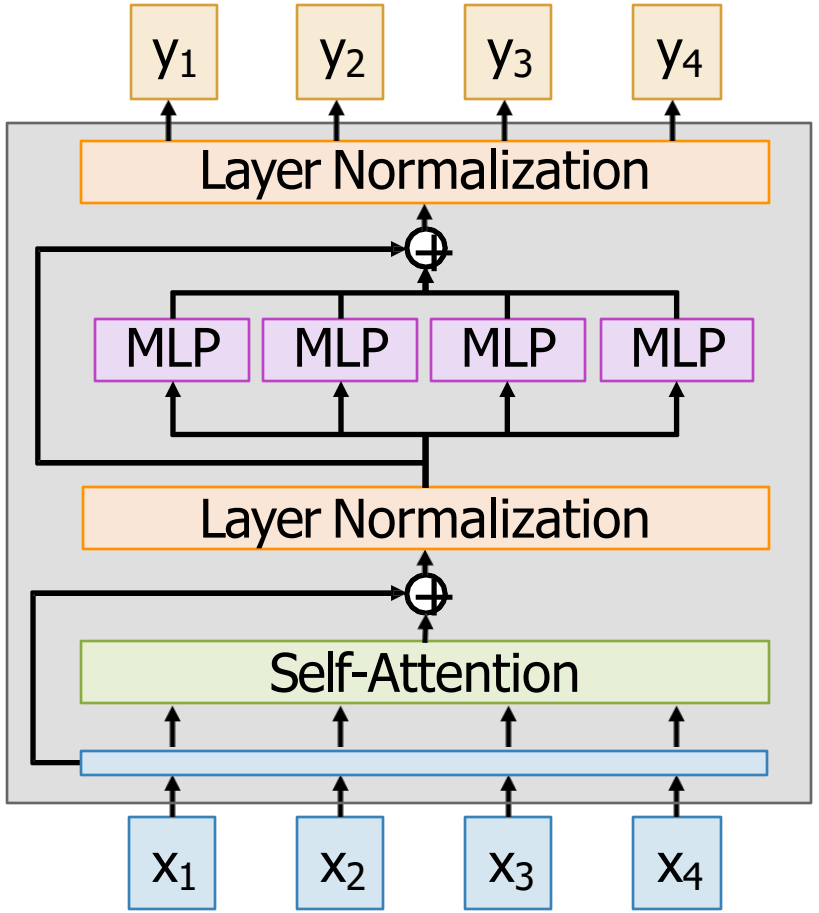
Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Tweaking Transformers

The Transformer architecture has not changed much since 2017.

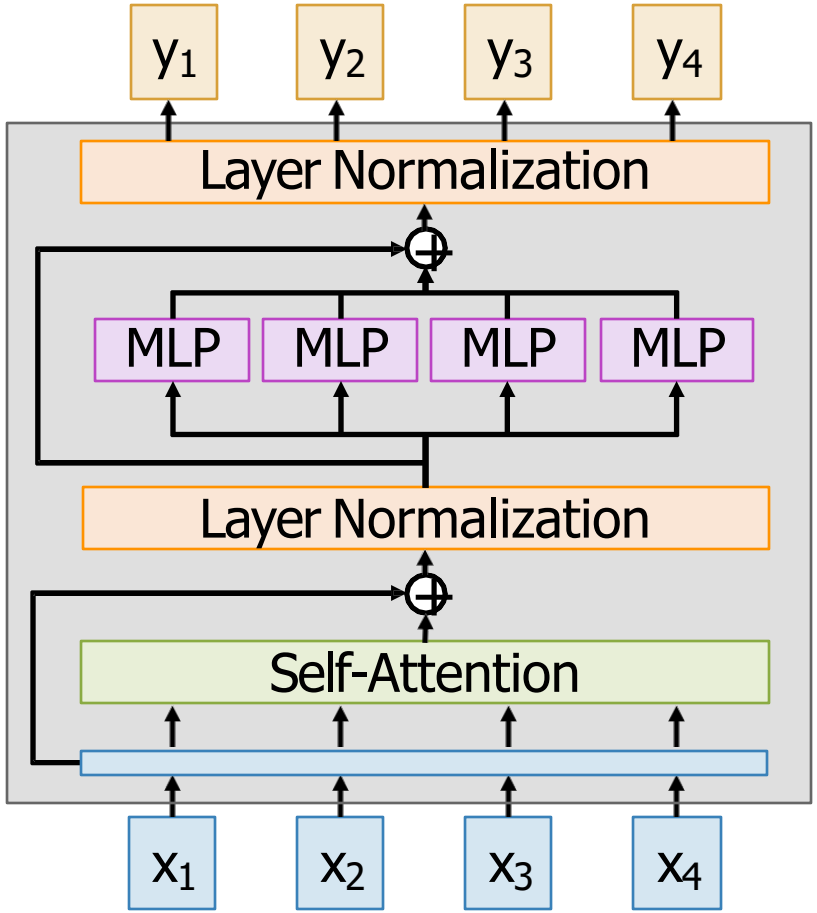
But a few changes have become common:



Pre-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function



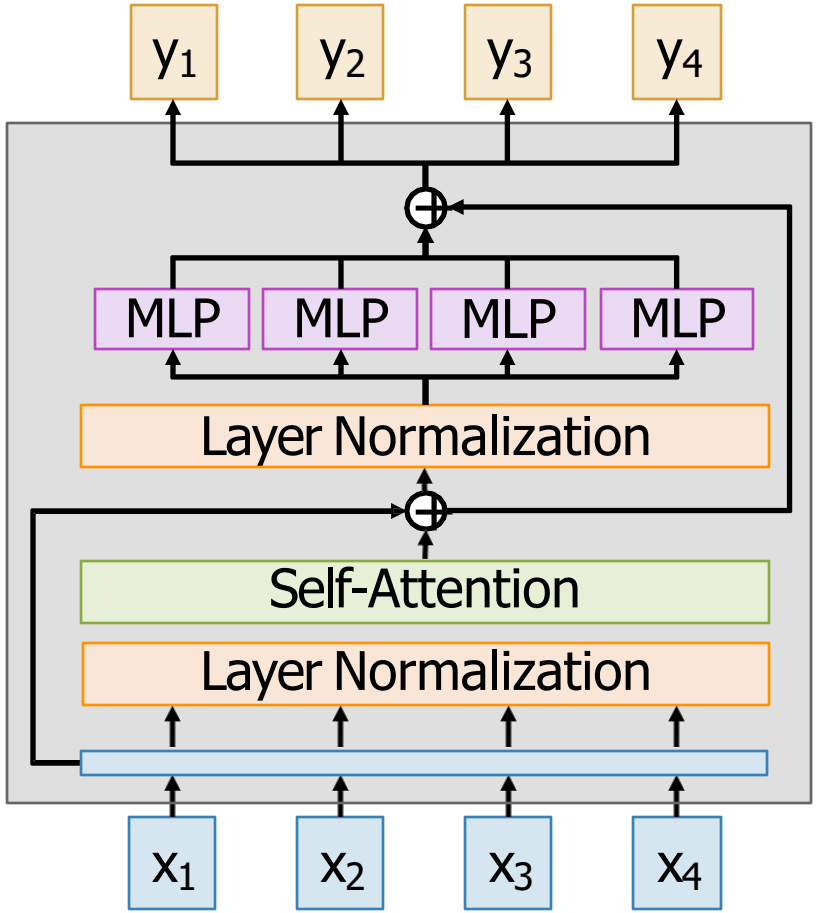
Baevski & Auli, "Adaptive Input Representations for Neural Language Modeling", arXiv 2018

Pre-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function

Solution: Move layer normalization before the Self-Attention and MLP, inside the residual connections. Training is more stable.



Baevski & Auli, "Adaptive Input Representations for Neural Language Modeling", arXiv 2018

RMSNorm

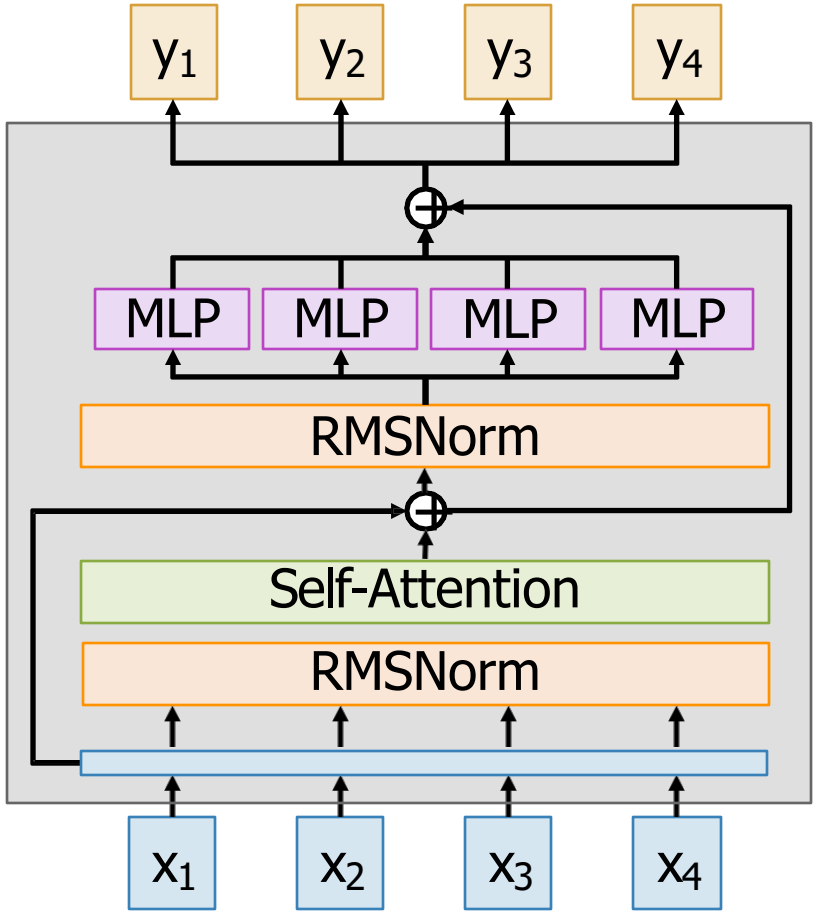
Replace Layer Normalization with Root-Mean-Square Normalization (RMSNorm)

Input: x [shape D]
Output: y [shape D]
Weight: γ [shape D]

$$y_i = \frac{x_i}{RMS(x)} * \gamma_i$$

$$RMS(x) = \sqrt{\varepsilon + \frac{1}{N} \sum_{i=1}^{N} x_i^2}$$

Training is a bit more stable

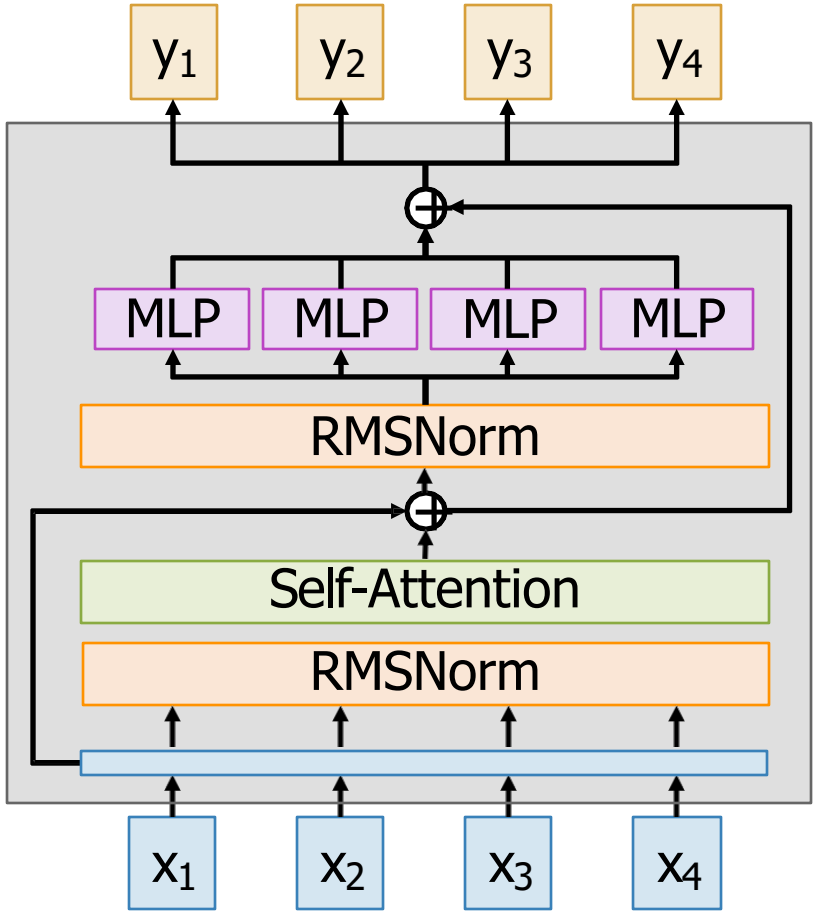


Zhang and Sennrich, "Root Mean Square Layer Normalization", NeurIPS 2019

SwiGLU MLP

Classic MLP:

Input: $X\,[N \times D]$
Weights: $W_1\,[D \times 4D]$
$W_2\,[4D \times D]$
Output: $Y = \sigma(XW_1)W_2\,[N \times D]$



Shazeer, "GLU Variants Improve Transformers", 2020

SwiGLU MLP

Classic MLP:

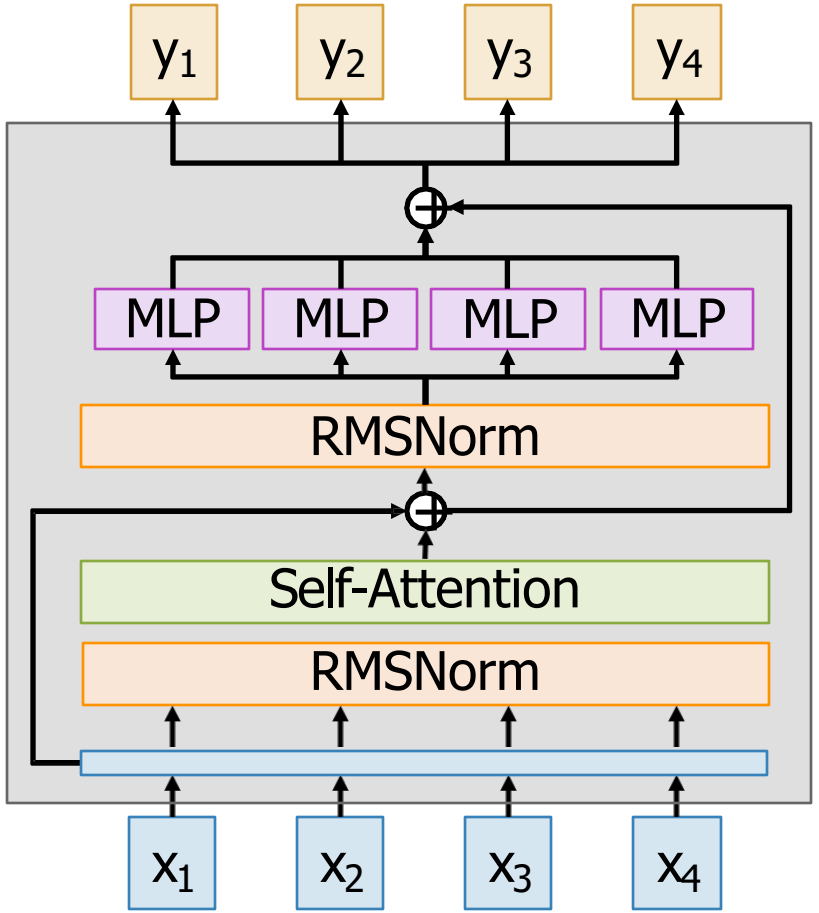
Input: X [N x D]
Weights: W_1 [D x 4D]
W_2 [4D x D]
Output: Y = σ(XW_1)W_2 [N x D]

SwiGLU MLP:

Input: X [N x D]
Weights: W_1 , W_2 [D x H]
W_3 [H x D]
Output:

$$Y = (\sigma(XW_1) \odot XW_2)W_3$$

Setting H =8D/3 keeps same total params

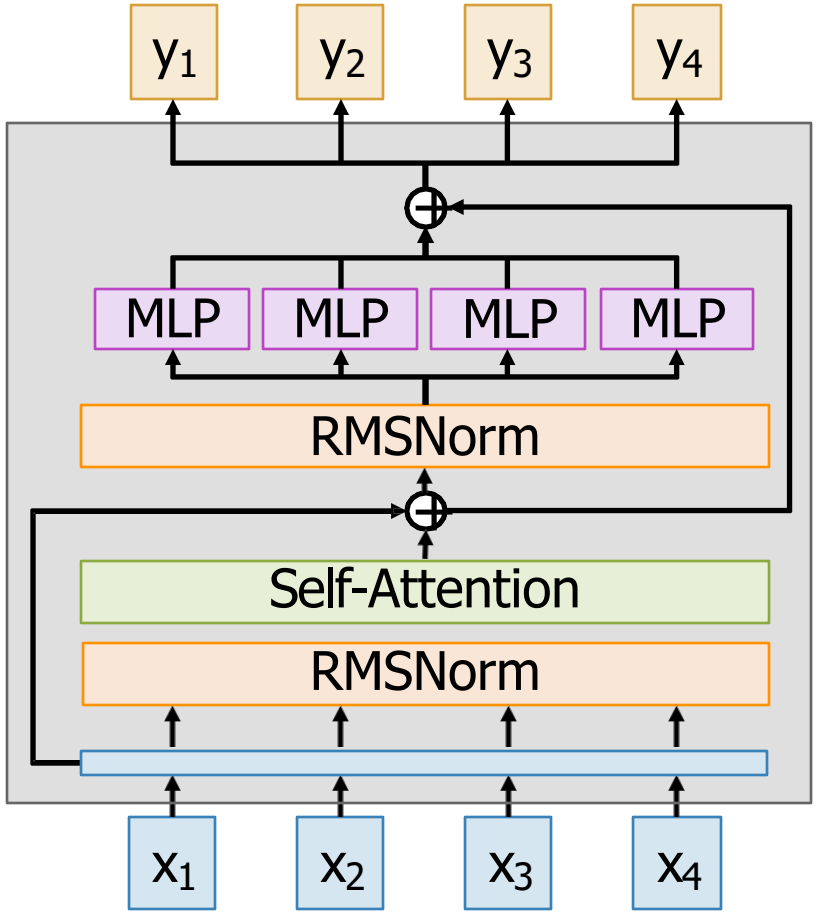


Shazeer, "GLU Variants Improve Transformers", 2020

Mixture of Experts (MoE)

Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]



Shazeer et al, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer", 2017

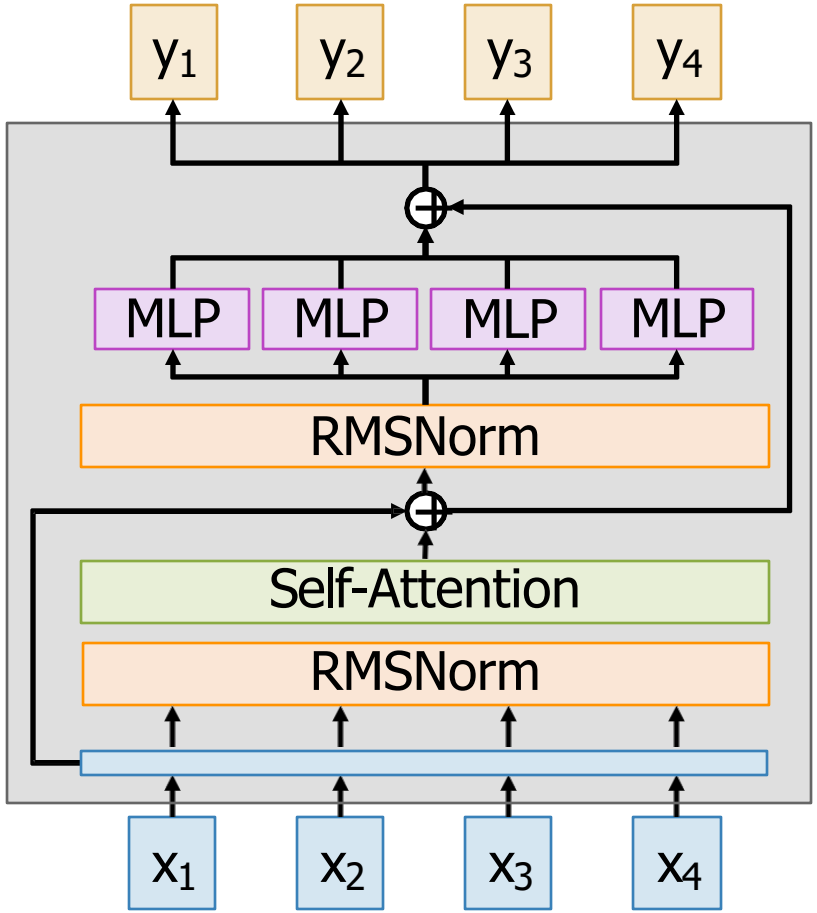
Mixture of Experts (MoE)

Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]

Each token gets routed to A < E of the experts. These are the active experts.

Increases params by E,
But only increases compute by A



Shazeer et al, "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer", 2017

Mixture of Experts (MoE)

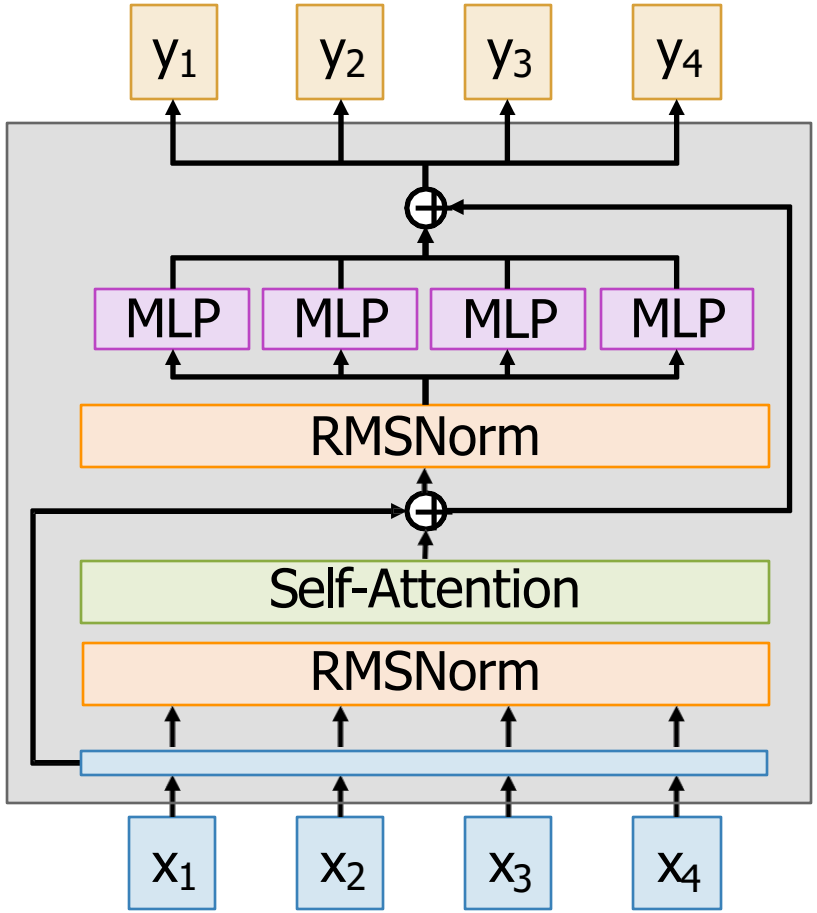
Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]

Each token gets routed to A < E of the experts. These are the active experts.

Increases params by E,
But only increases compute by A

All of the biggest LLMs today (e.g. GPT4o, GPT4.5, Claude 3.7, Gemini 2.5 Pro, etc) almost certainly use MoE and have >1T params; but they don't publish details anymore

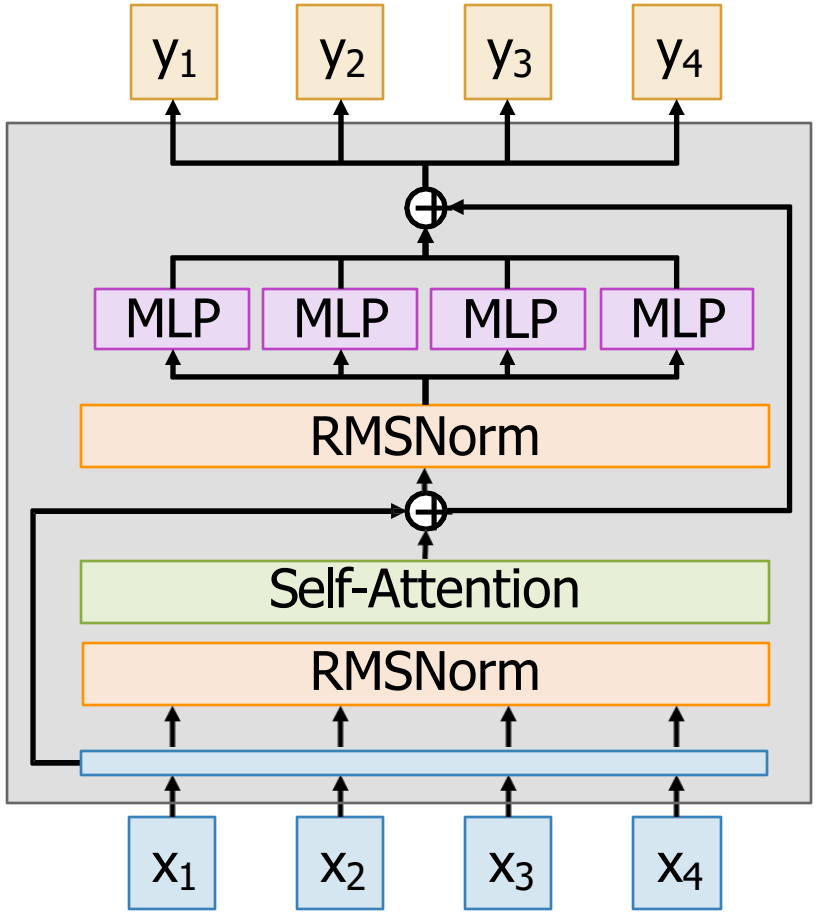


Tweaking Transformers

The Transformer architecture has not changed much since 2017.

But a few changes have become common:

- Pre-Norm: Move normalization inside residual
- RMSNorm: Different normalization layer
- SwiGLU: Different MLP architecture
- Mixture of Experts (MoE): Learn E different MLPs, use A <E of them per token. Massively increase params, modest increase to compute cost.



Today

- Transformers Recap

- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation

- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

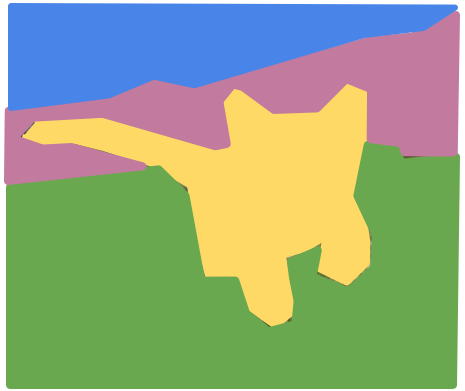
Computer Vision Tasks

Classification

CAT

No spatial extent

Semantic Segmentation



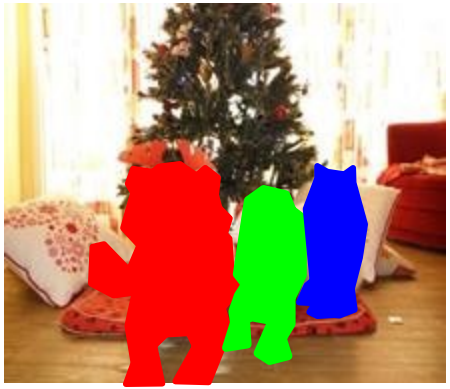
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Recall: Image Classification: A core task in Computer Vision

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

cat

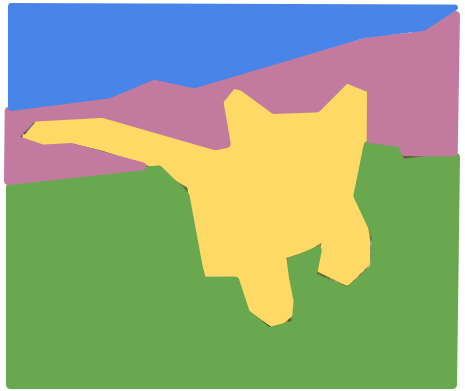
Semantic Segmentation

Classification

CAT

No spatial extent

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

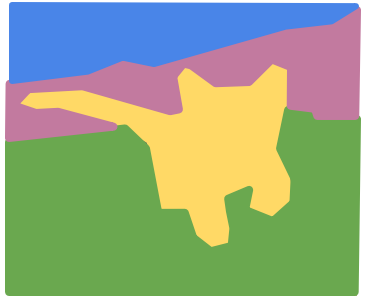
DOG, DOG, CAT

Multiple Object

Instance Segmentation

DOG, DOG, CAT

Semantic Segmentation: The Problem



GRASS, CAT, TREE,
SKY, …

Paired training data: for each training image, each pixel is labeled with a semantic category.

?

At test time, classify each pixel of a new image.

Semantic Segmentation Idea: Sliding Window

Full image

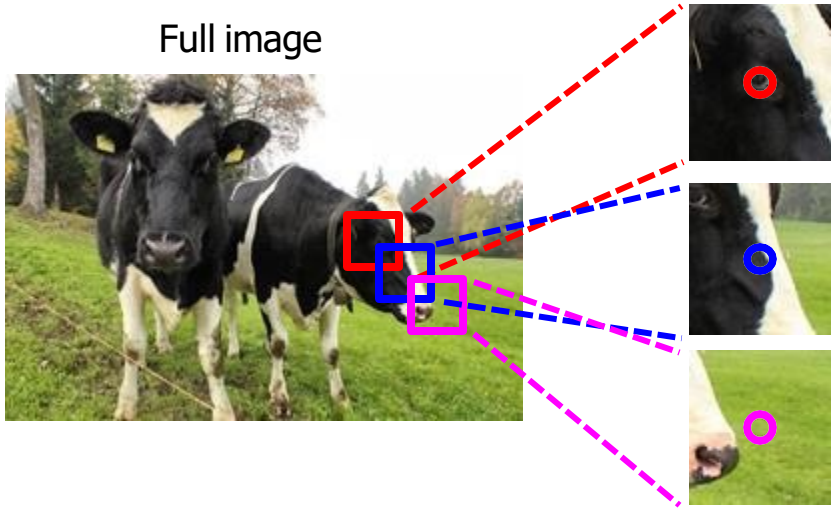
Semantic Segmentation Idea: Sliding Window

Full image

Impossible to classify without context

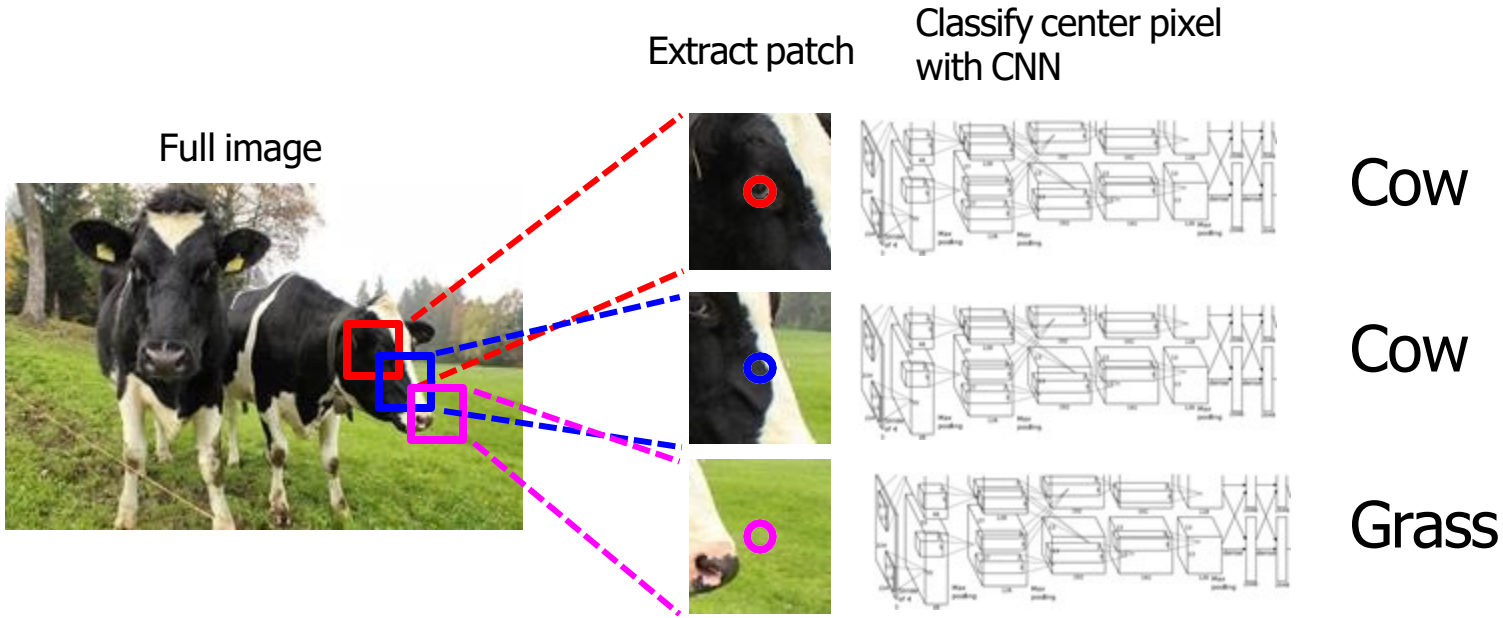
Q: how do we include context?

Semantic Segmentation Idea: Sliding Window



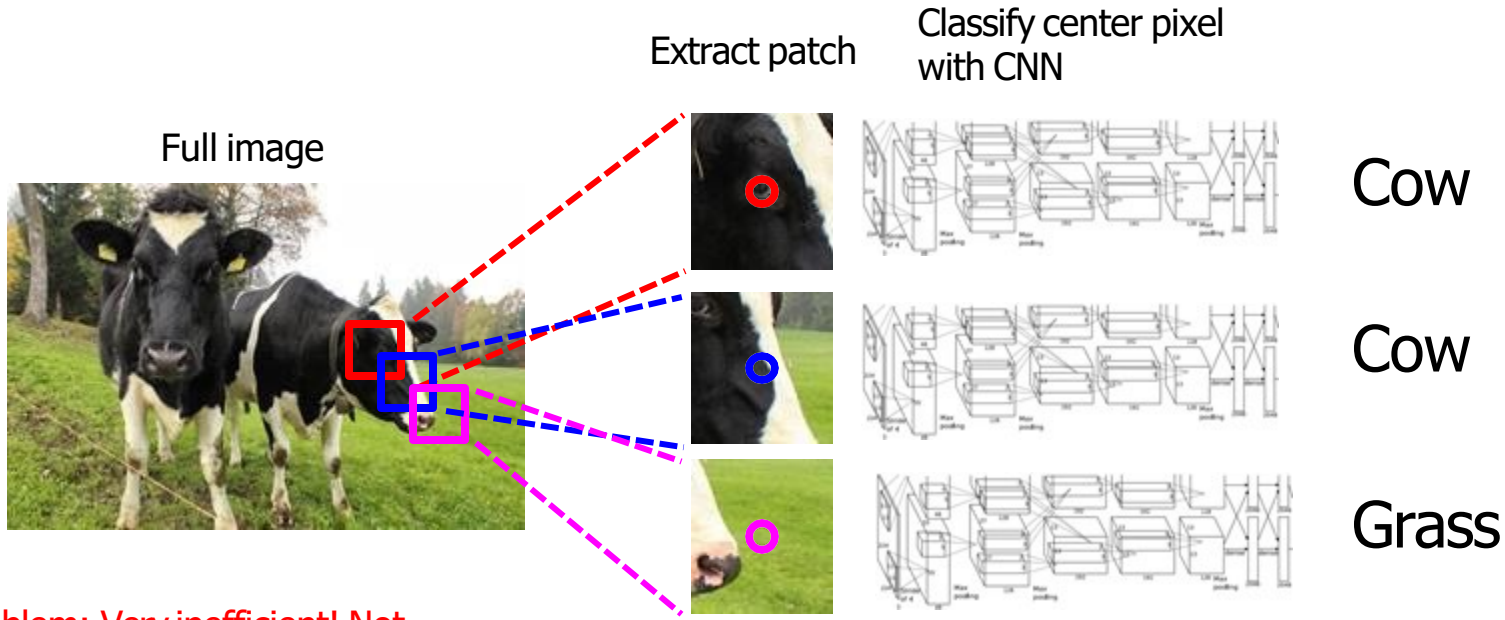
Q: how do we model this?

Semantic Segmentation Idea: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Sliding Window

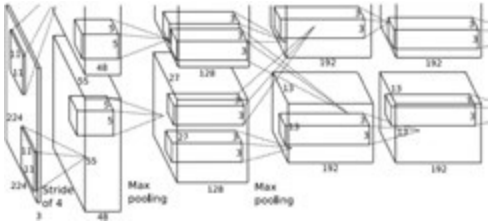


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Convolution

Full image

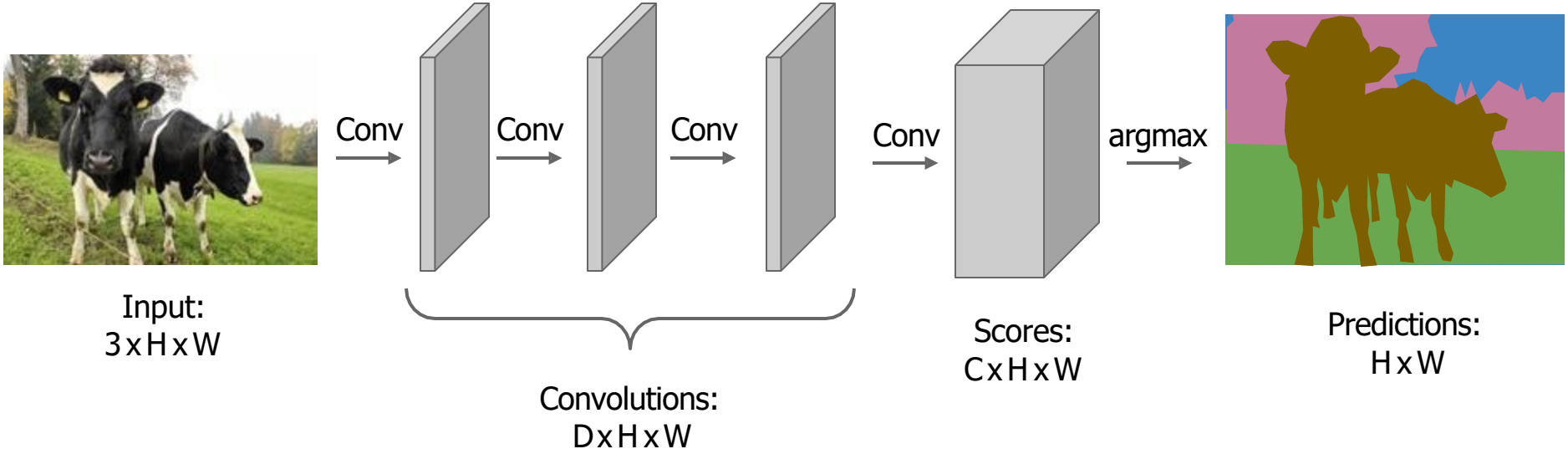


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!

Input:
3 x H x W

Conv

Conv

Conv

Convolutions:
D x H x W

Conv

Scores:
C x H x W

argmax

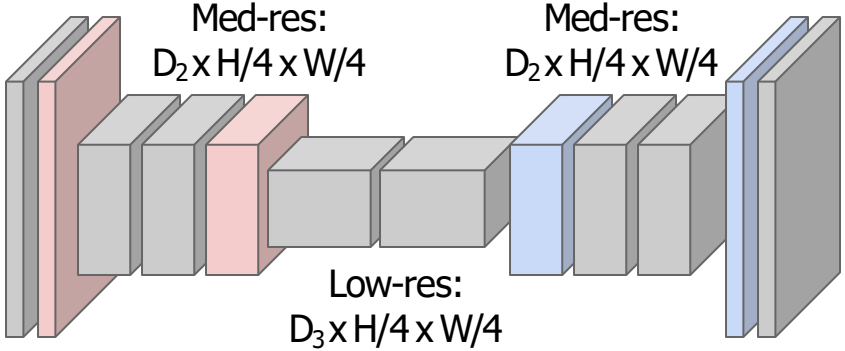
Predictions:
H x W

Problem: convolutions at original image resolution will be very expensive ...

Semantic Segmentation Idea: Fully Convolutional

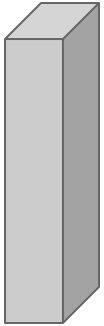
Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2



C x H x W

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

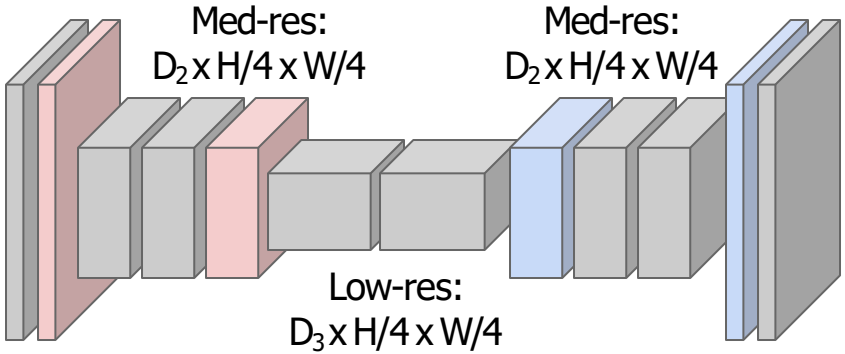
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

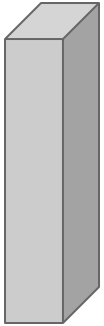
Upsampling:
???

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2

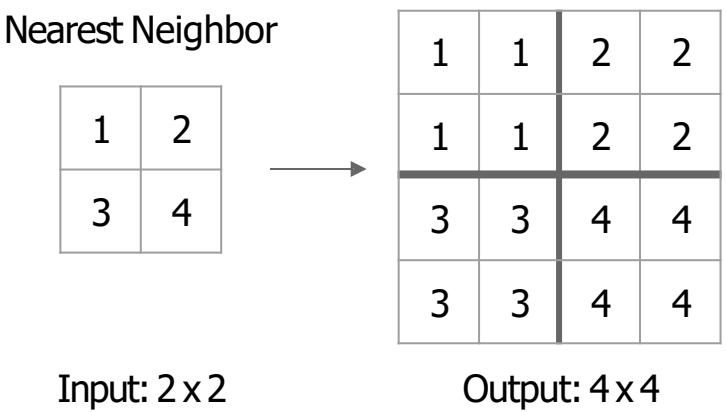


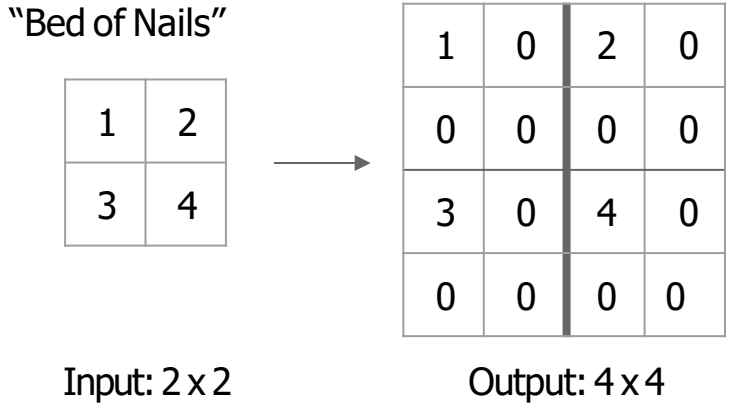
C x H x W

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network upsampling: “Unpooling”





In-Network upsampling: "Max Unpooling"

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

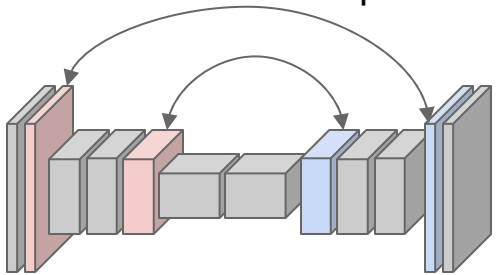
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

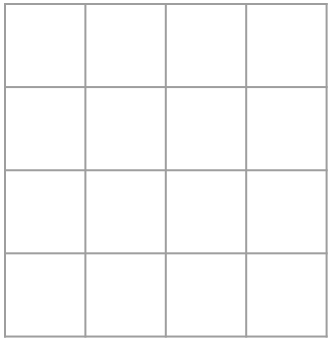
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

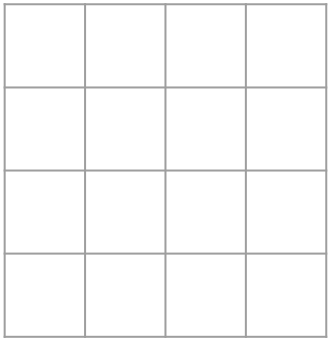


Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 1 pad 1



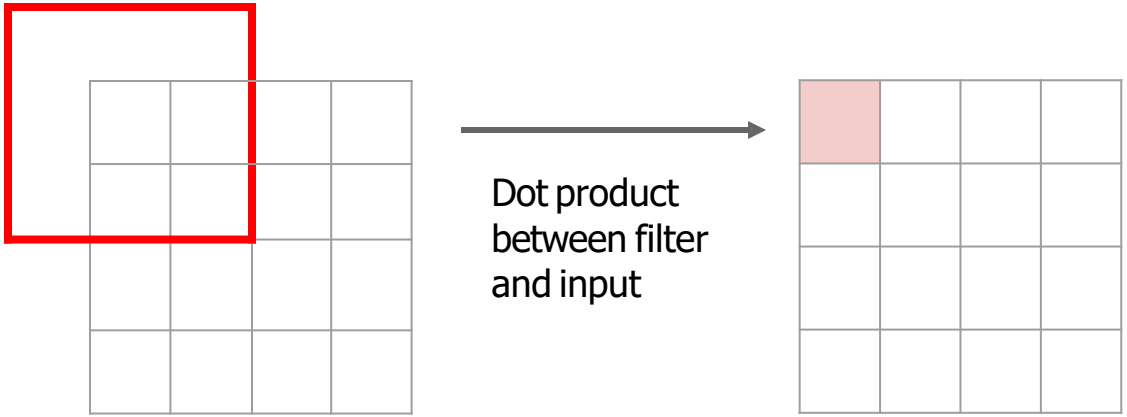
Input: 4 x 4



Output: 4 x 4

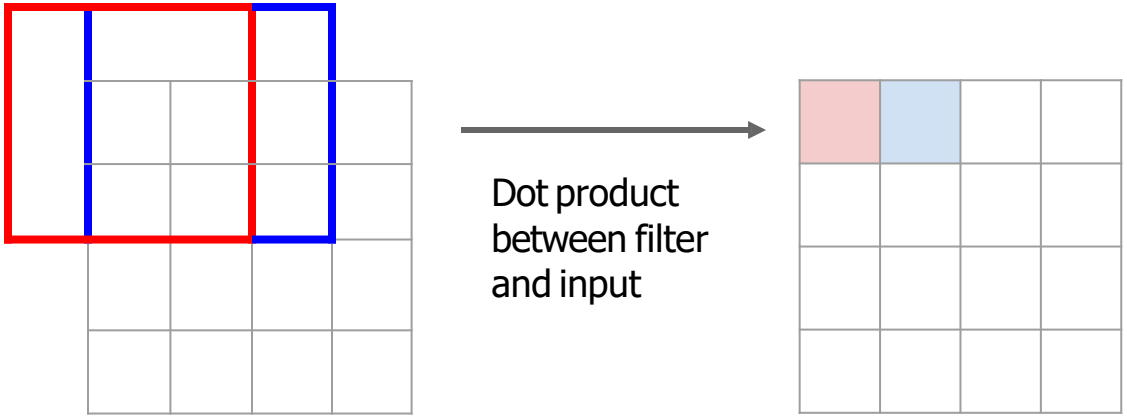
Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 1 pad 1

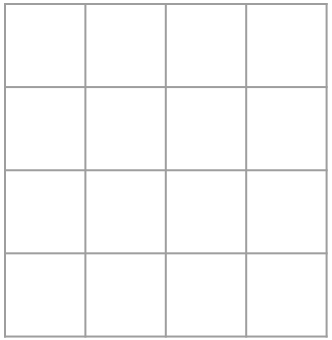


Input: 4 x 4

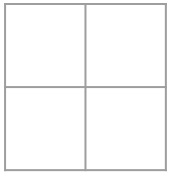
Output: 4 x 4

Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



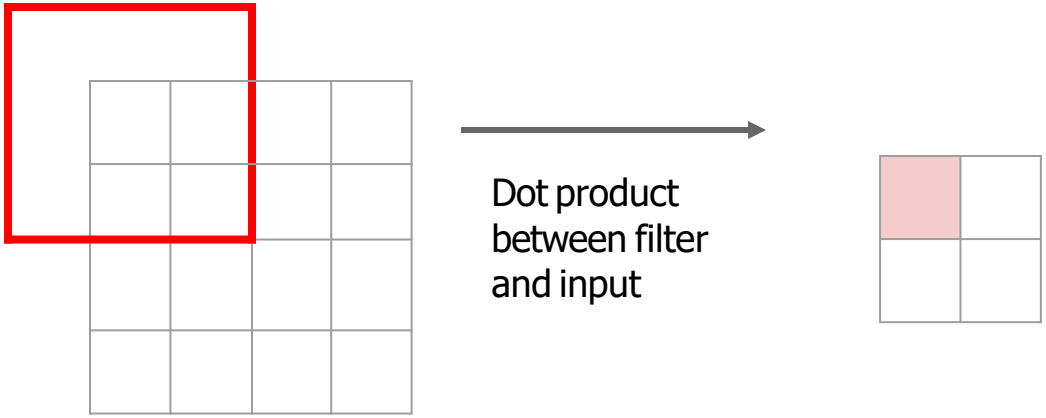
Input: 4 x 4



Output: 2 x 2

Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1

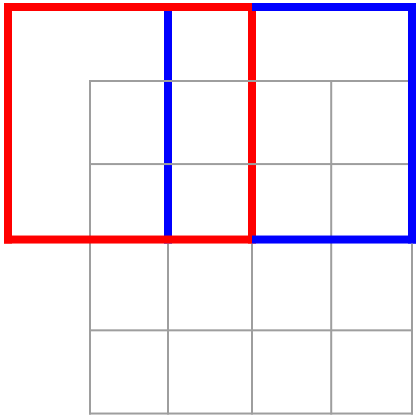


Input: 4 x 4

Output: 2 x 2

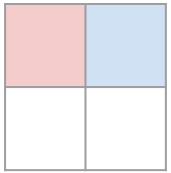
Learnable Upsampling

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

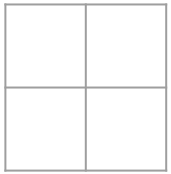
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

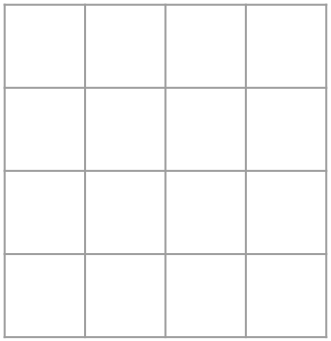
We can interpret strided convolution as "learnable downsampling".

Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



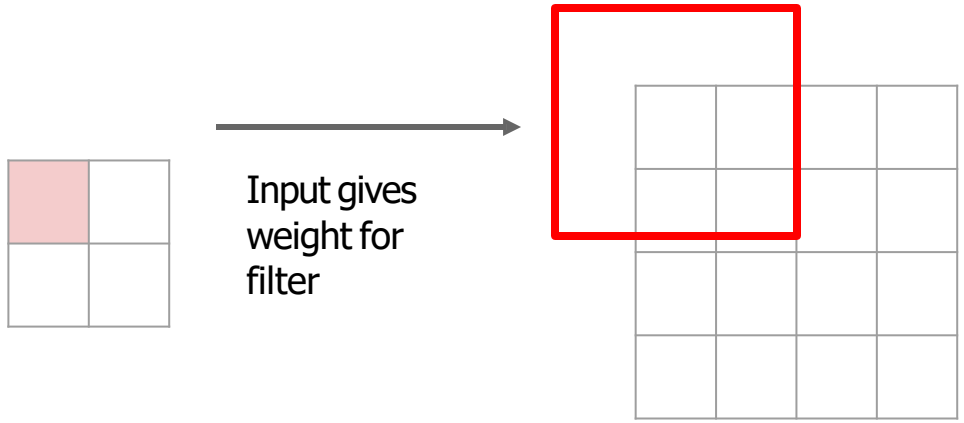
Input: 2 x 2



Output: 4 x 4

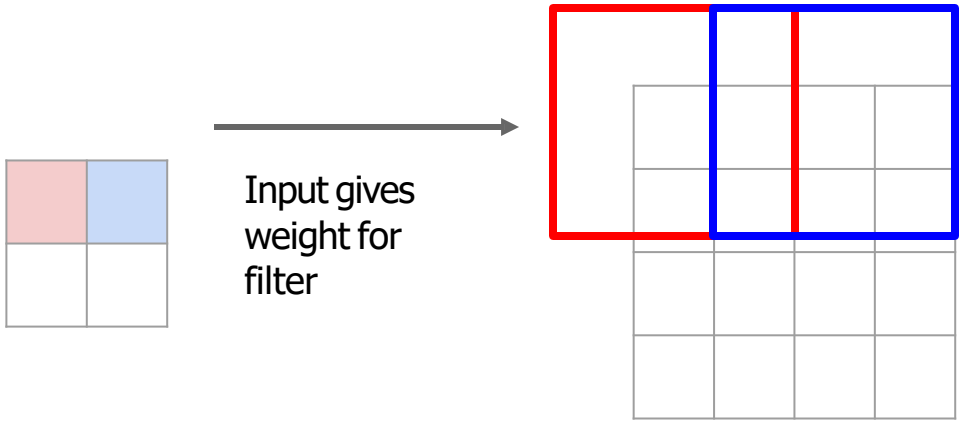
Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



Input gives weight for filter

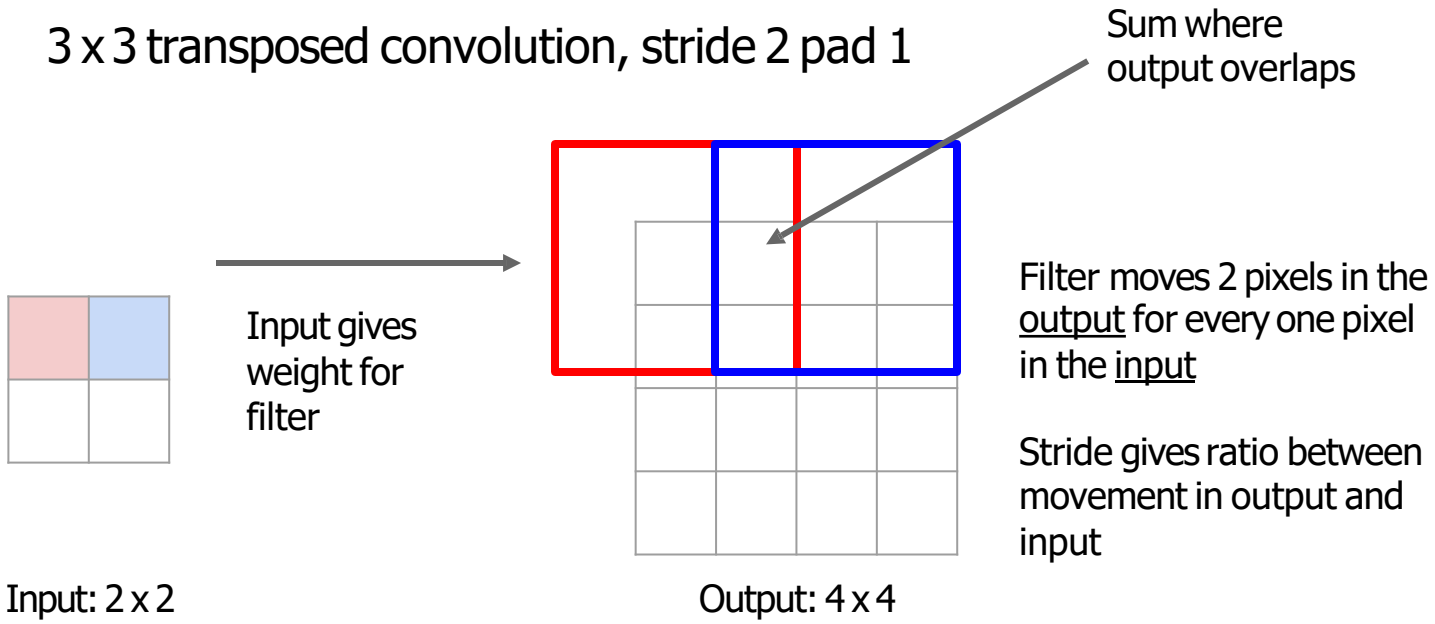
Input: 2 x 2

Output: 4 x 4

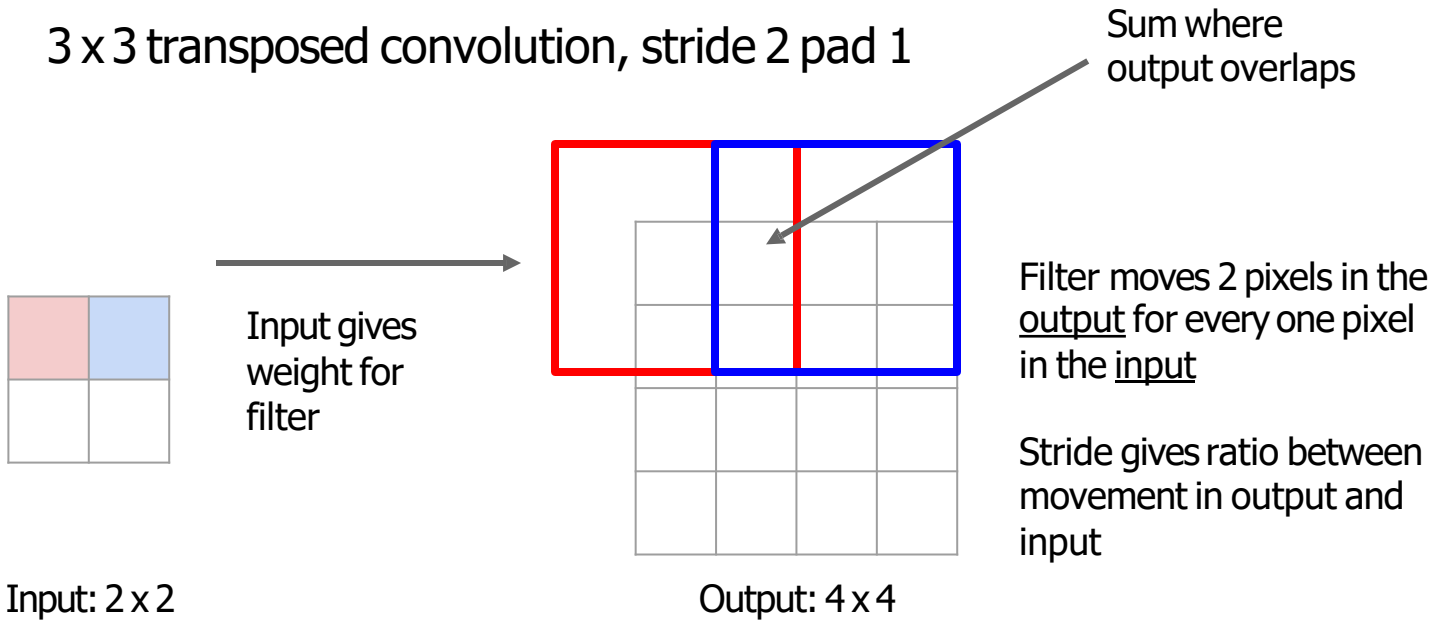
Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

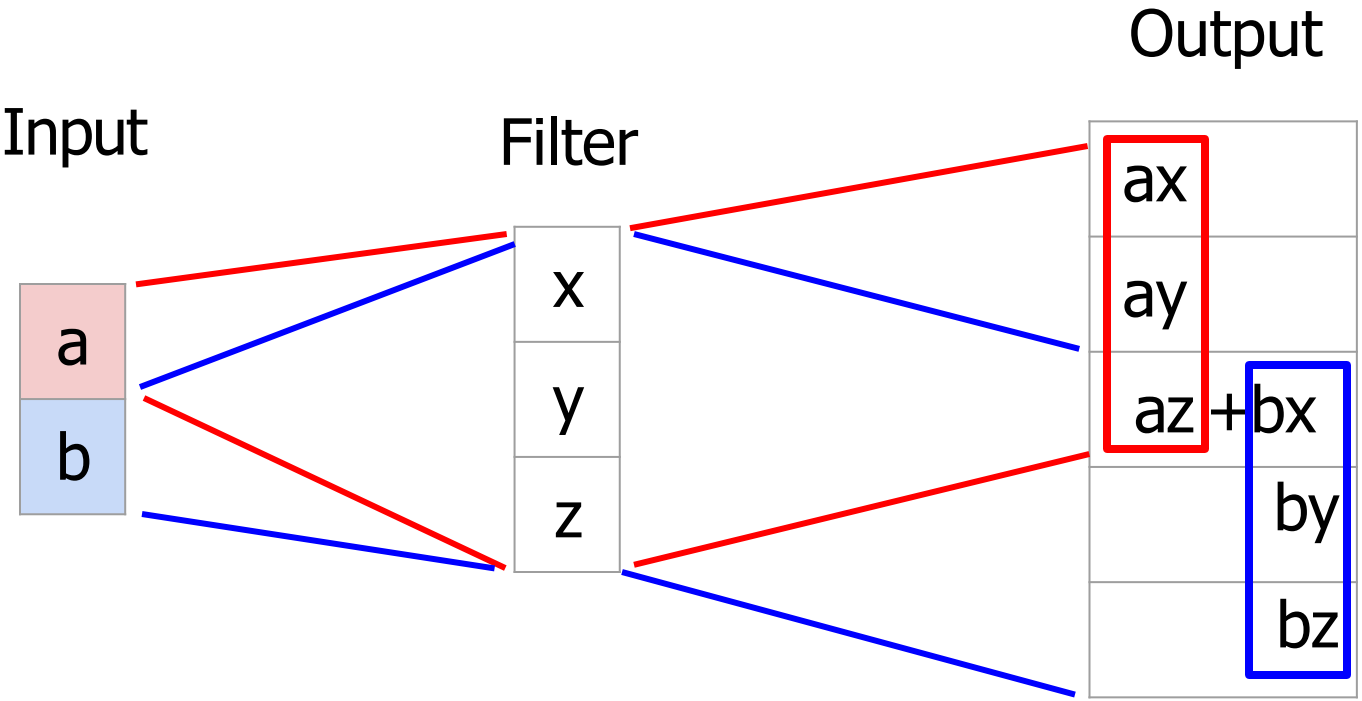
Learnable Upsampling: Transposed Convolution



Learnable Upsampling: Transposed Convolution



Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

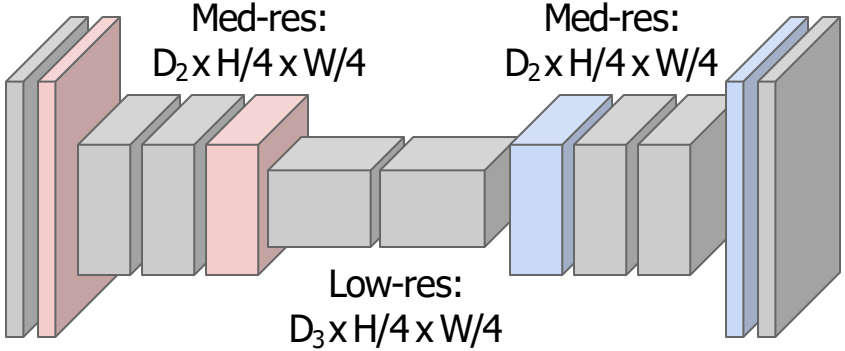
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

Upsampling:
Unpooling or strided transposed convolution

Input:
3 x H x W



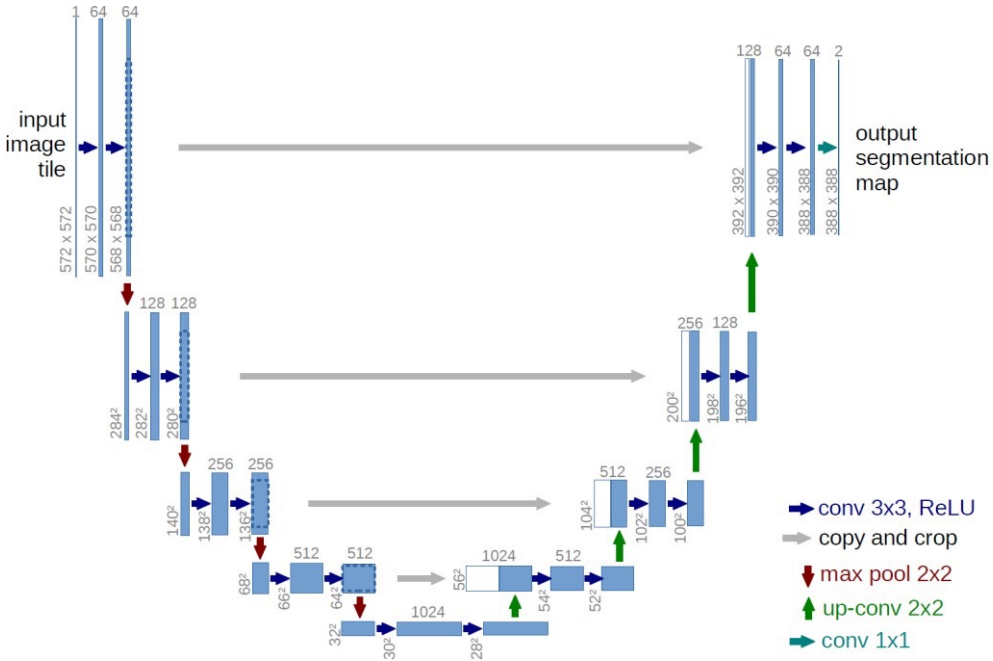
High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2

Predictions:
H x W

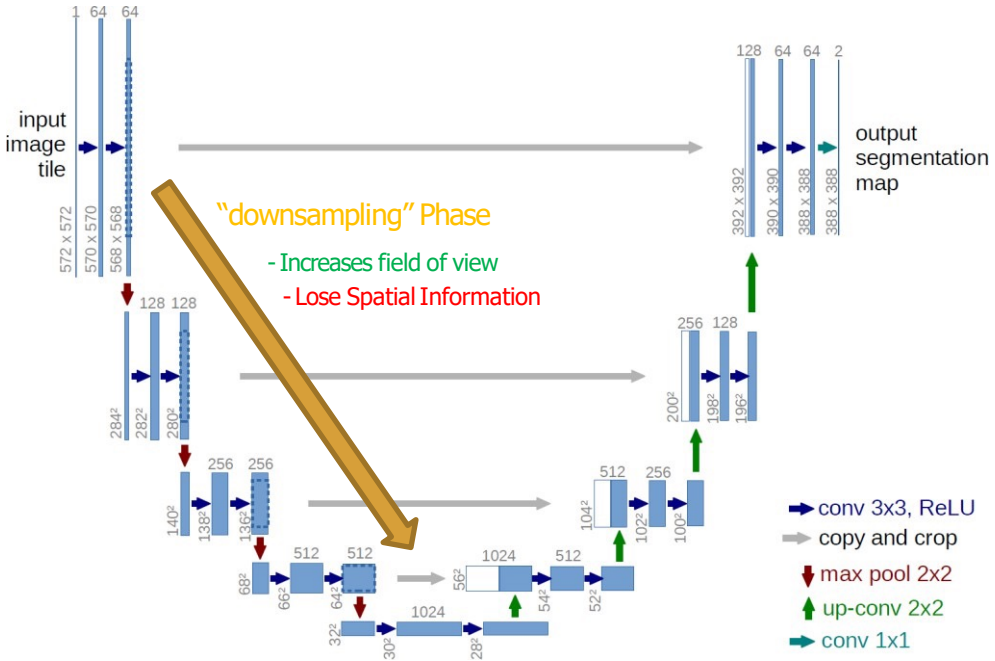
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

U-Net



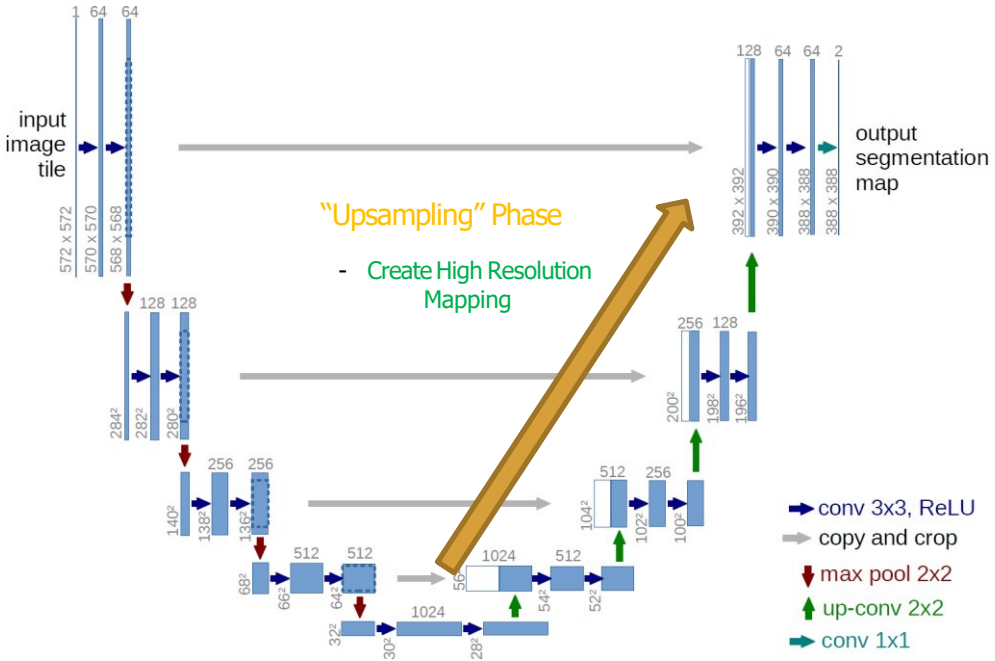
Ronneberger et al. (2015) U-net Architecture

U-Net



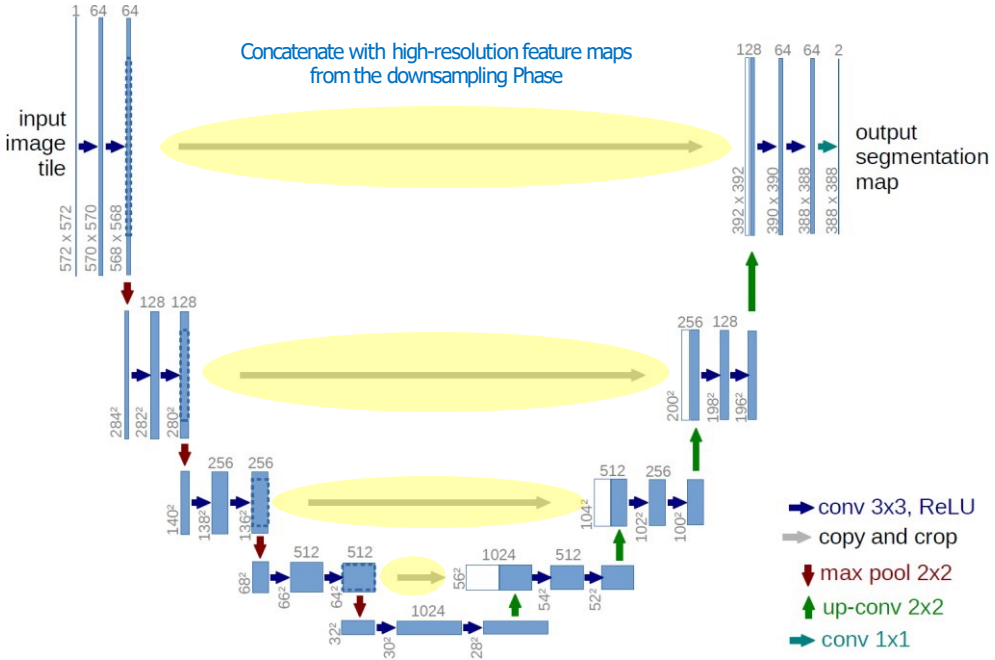
Ronneberger et al. (2015) U-net Architecture

U-Net



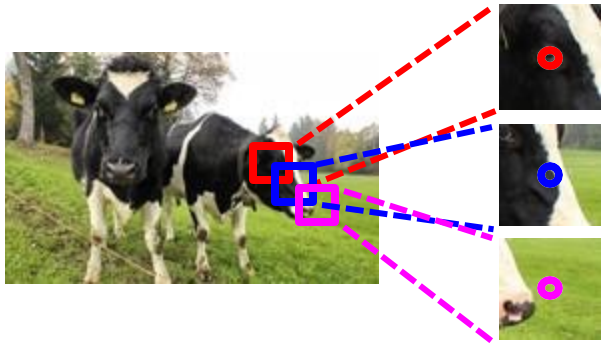
Ronneberger et al. (2015) U-net Architecture

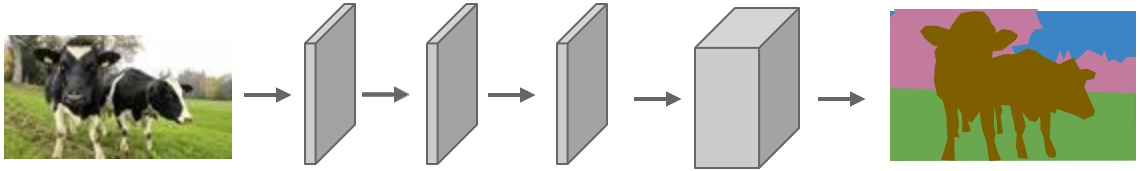
U-Net

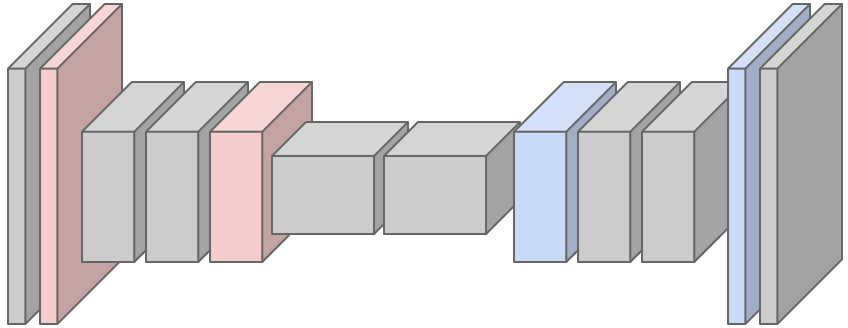


Ronneberger et al. (2015) U-net Architecture

Semantic Segmentation: Summary



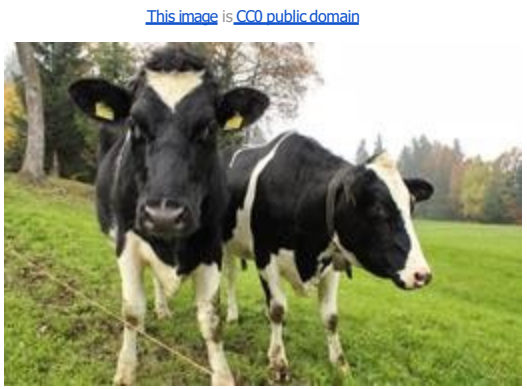


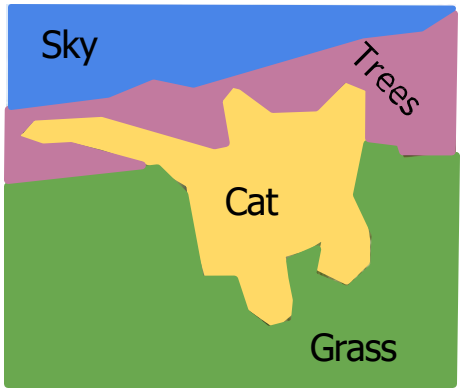


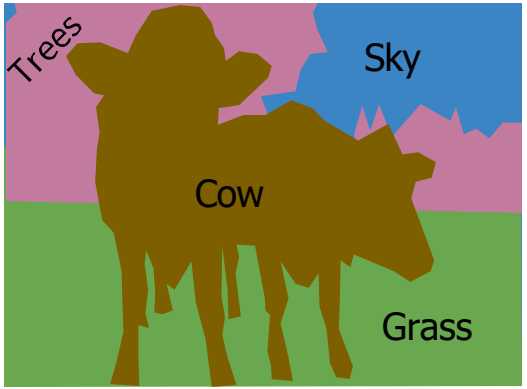
Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels







Object Detection

Classification

CAT

No spatial extent

Semantic Segmentation

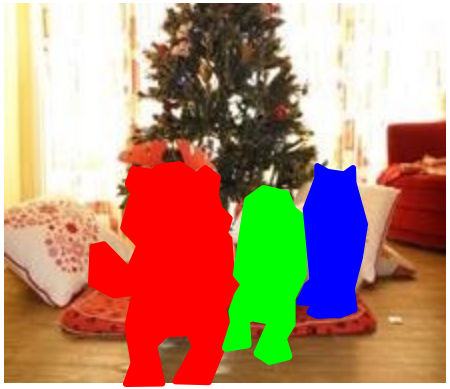
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Object Detection

Classification

CAT

No spatial extent

Semantic Segmentation

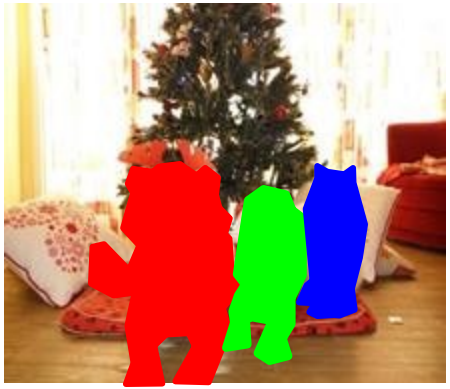
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation

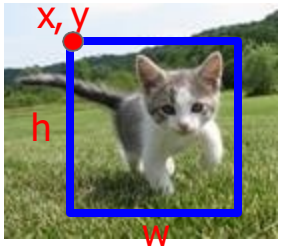


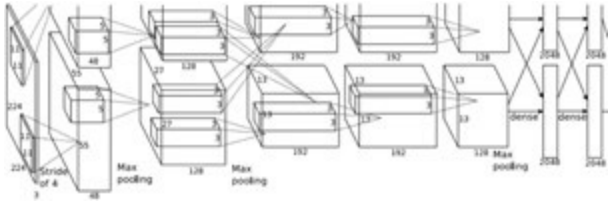
DOG, DOG, CAT

Multiple Object

Object Detection: Single Object

(Classification +Localization)





Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

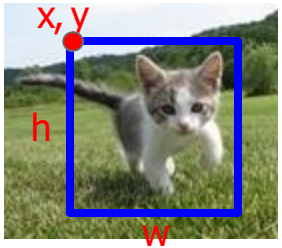
Vector: 4096

Fully Connected: 4096 to 4

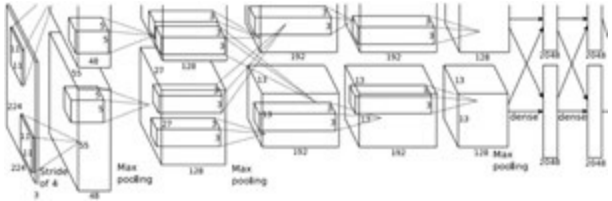
Box Coordinates
(x, y, w, h)

Object Detection: Single Object

(Classification +Localization)



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

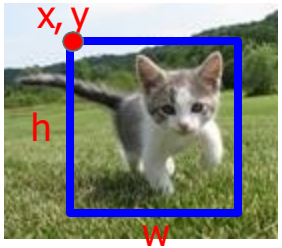
L2 Loss

Correct box: (x', y', w', h')

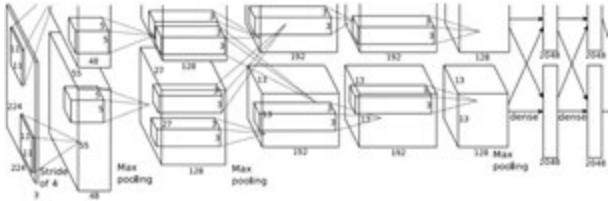
Treat localization as a regression problem!

Object Detection: Single Object

(Classification +Localization)



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

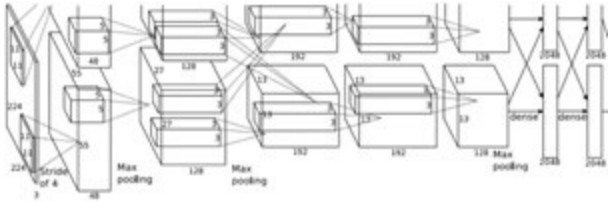
Correct box: (x', y', w', h')

Multitask Loss

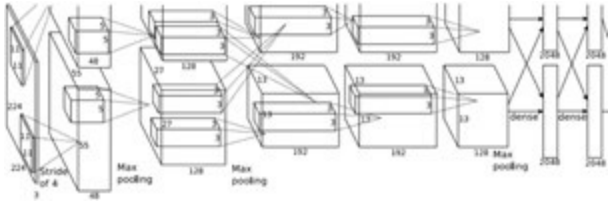
+ Loss

Treat localization as a regression problem!

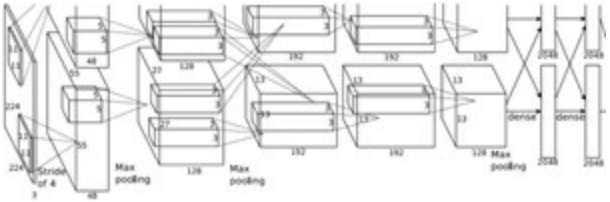
Object Detection: Multiple Objects



CAT: (x, y, w, h)



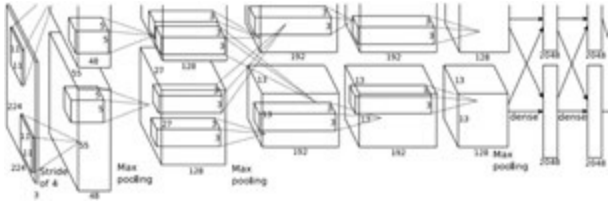
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

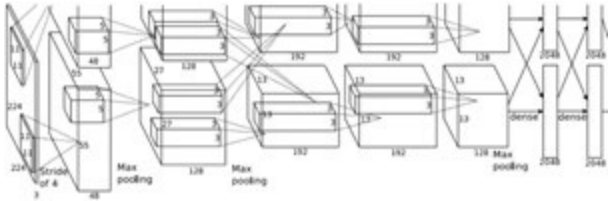
Object Detection: Multiple Objects

Each image needs a different number of outputs!



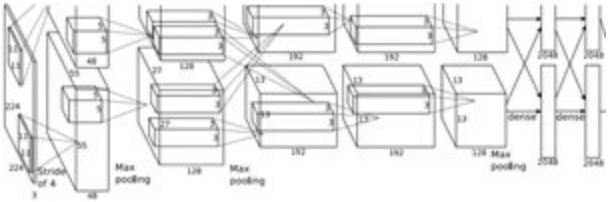
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers

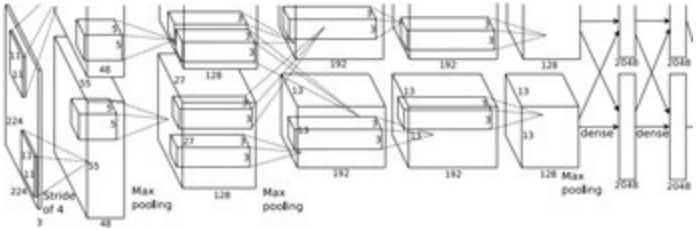


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

Object Detection: Multiple Objects

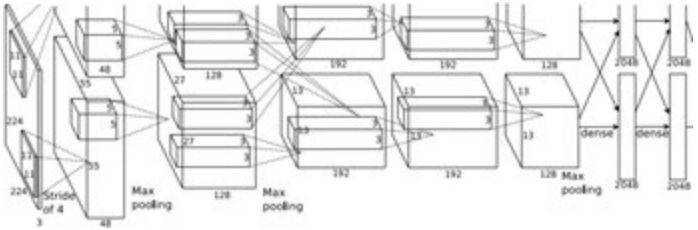
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

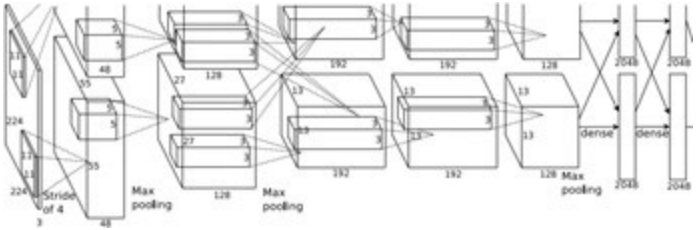
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

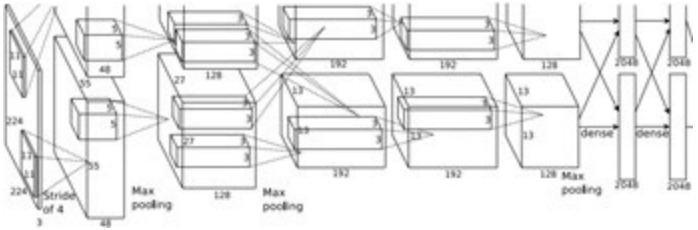
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

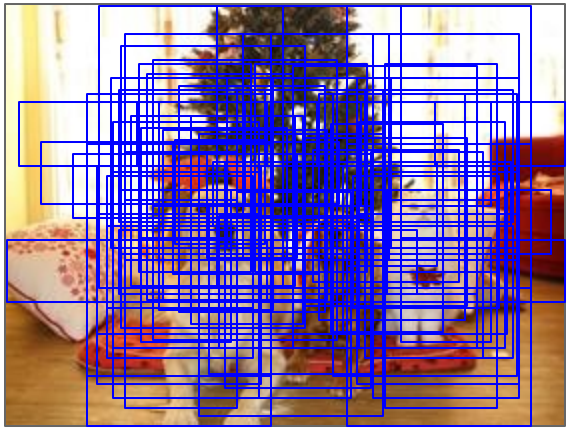
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



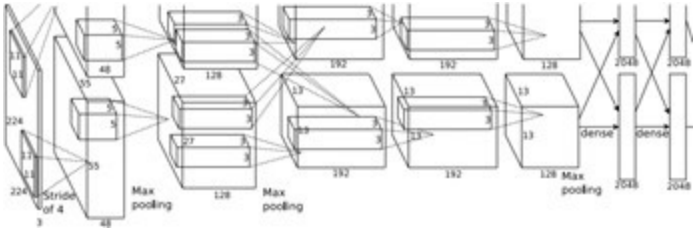
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Object Detection: Multiple Objects



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

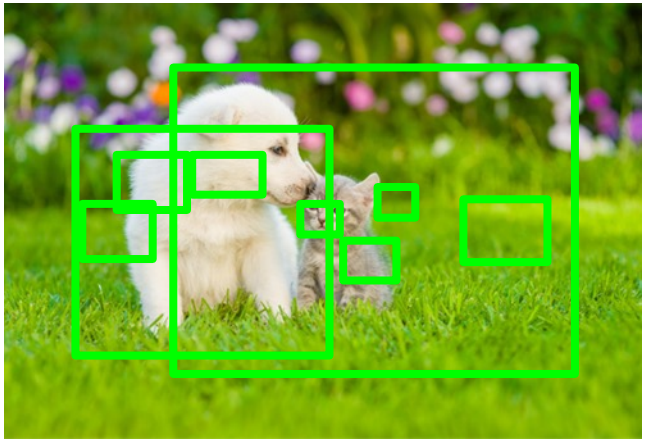


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals: Selective Search

- Find "blobby" image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

R-CNN

Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

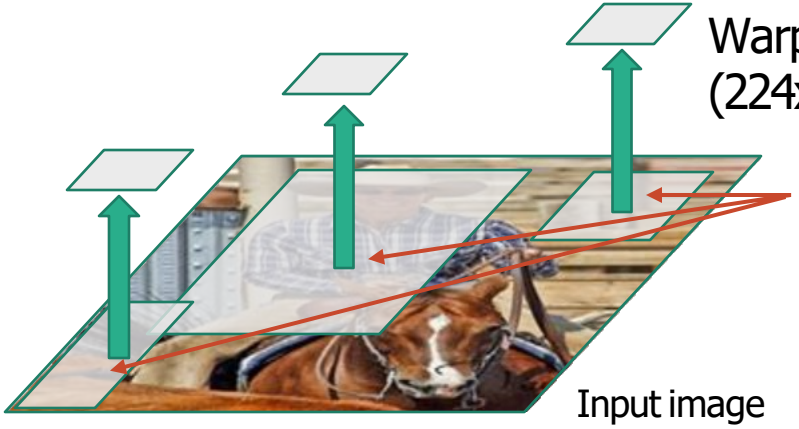
R-CNN

Input image

Regions of Interest (RoI) from a proposal method (~2k)

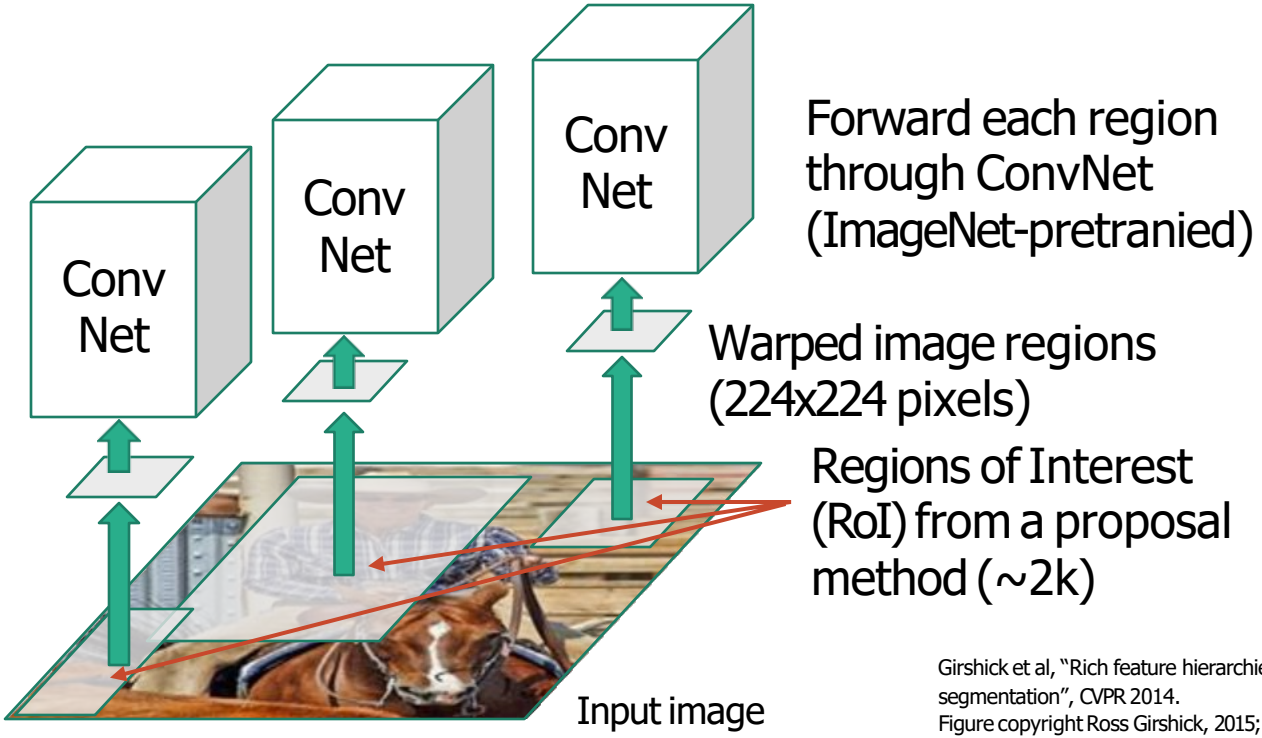
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



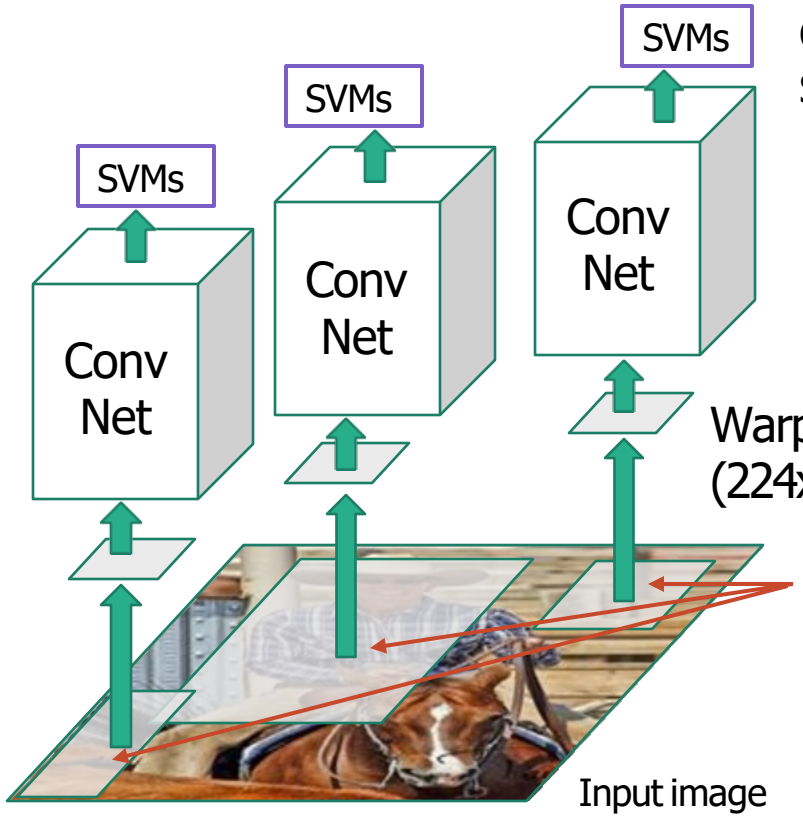
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

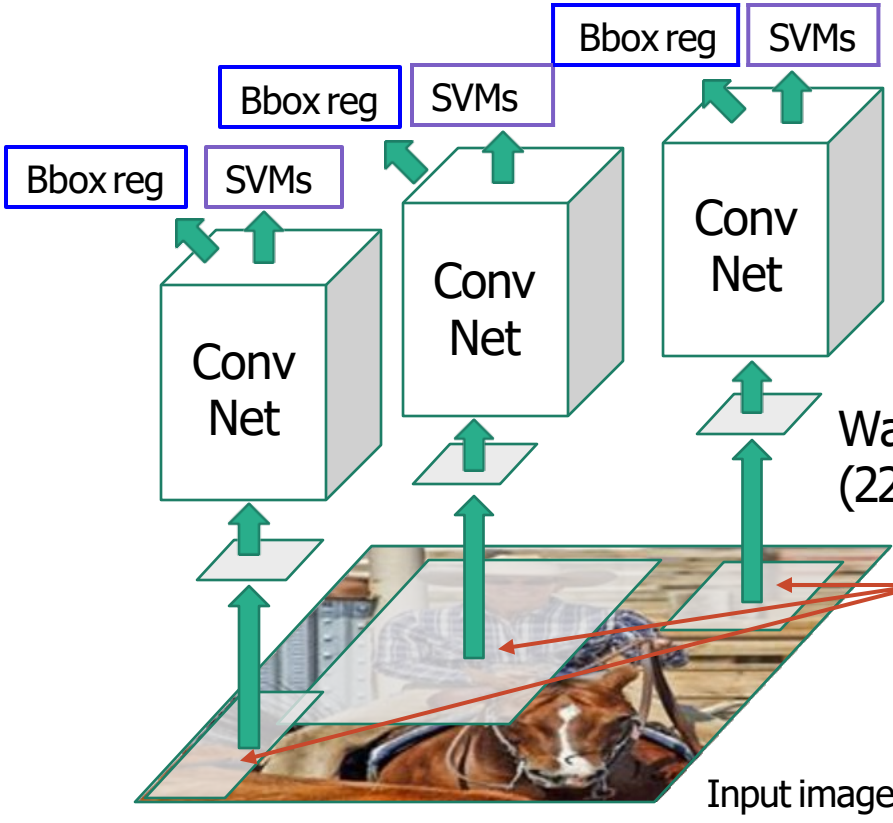
Regions of Interest (RoI) from a proposal method (~2k)

Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN

Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

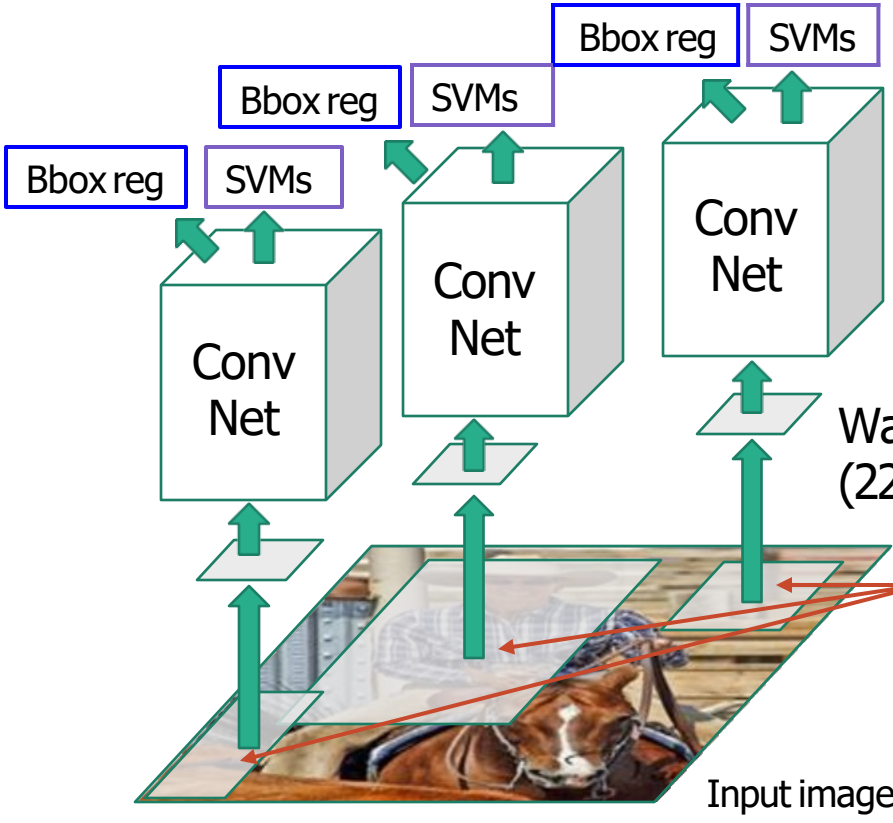
Regions of Interest (RoI) from a proposal method (~2k)

Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

R-CNN

Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

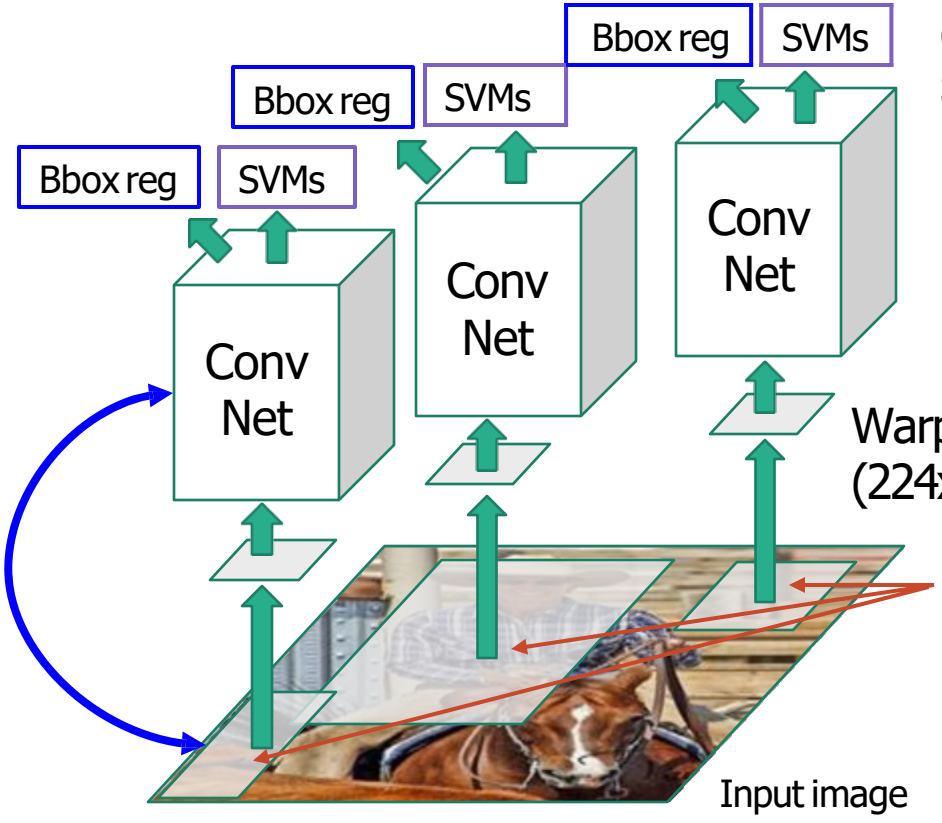
Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

"Slow" R-CNN

Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do ~2k independent forward passes for each image!

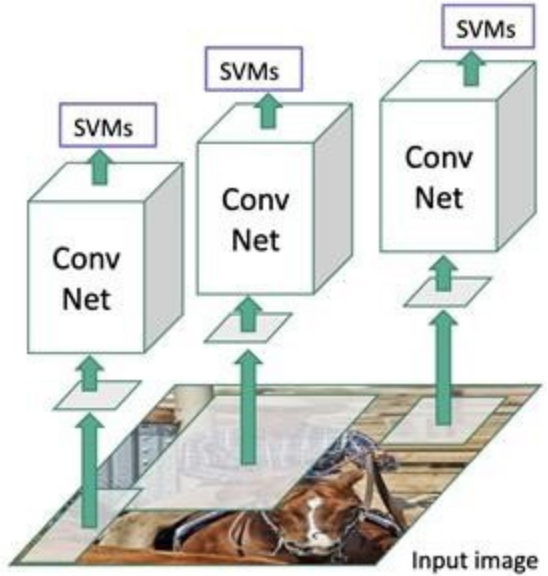
Idea: Pass the image through convnet before cropping! Crop the conv feature instead!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Fast R-CNN

Input image

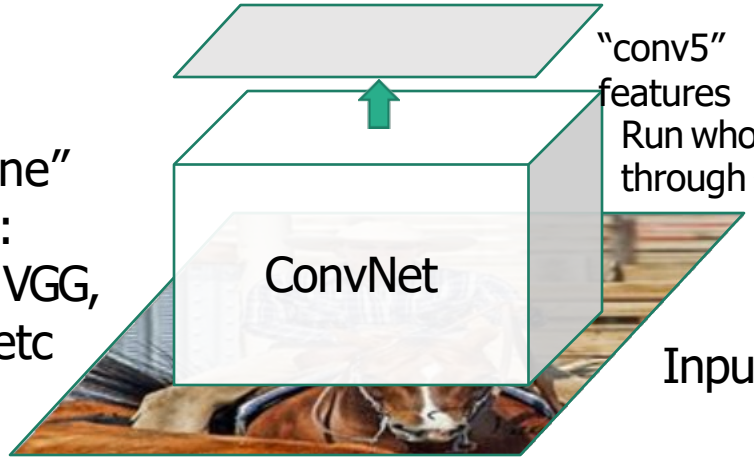
"Slow" R-CNN



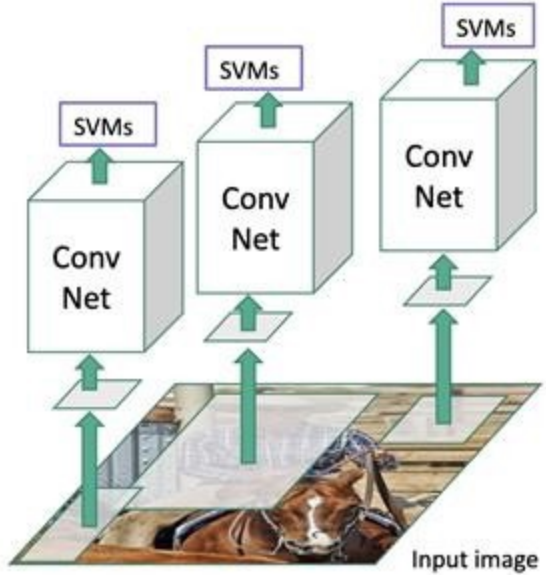
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

"Backbone" network: AlexNet, VGG, ResNet, etc



"Slow" R-CNN

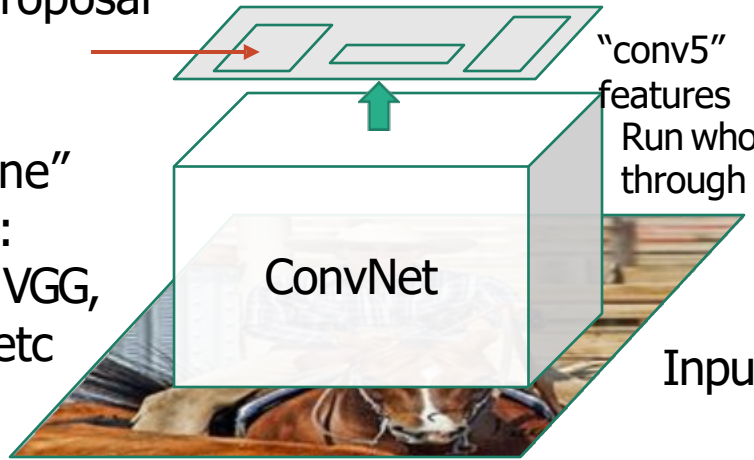


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

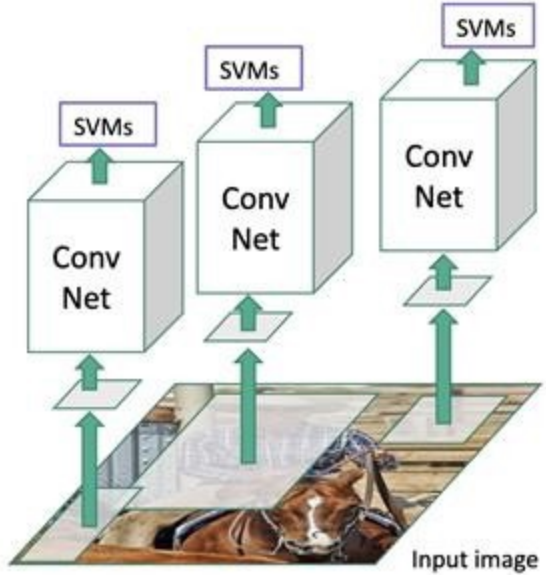
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



"Slow" R-CNN

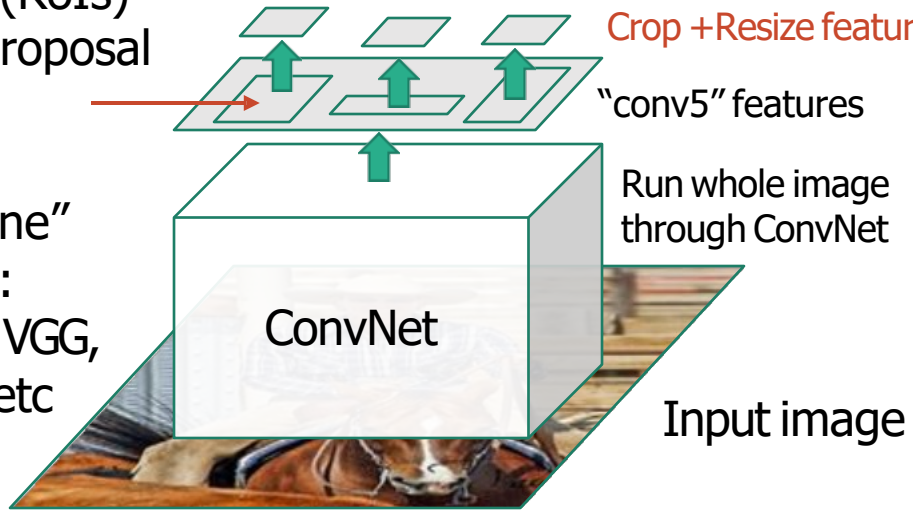


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

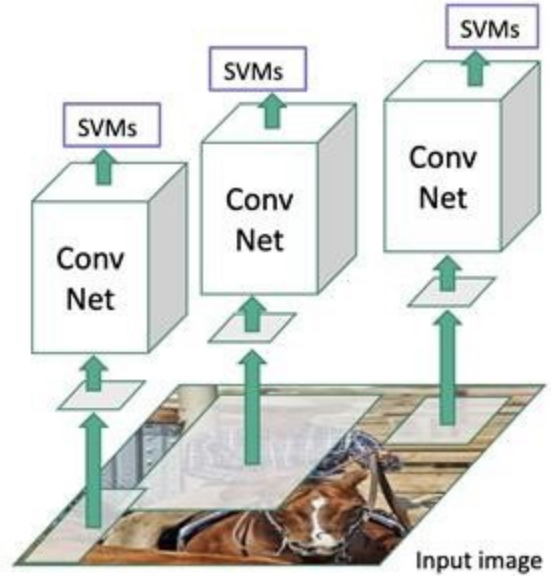
Fast R-CNN

Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

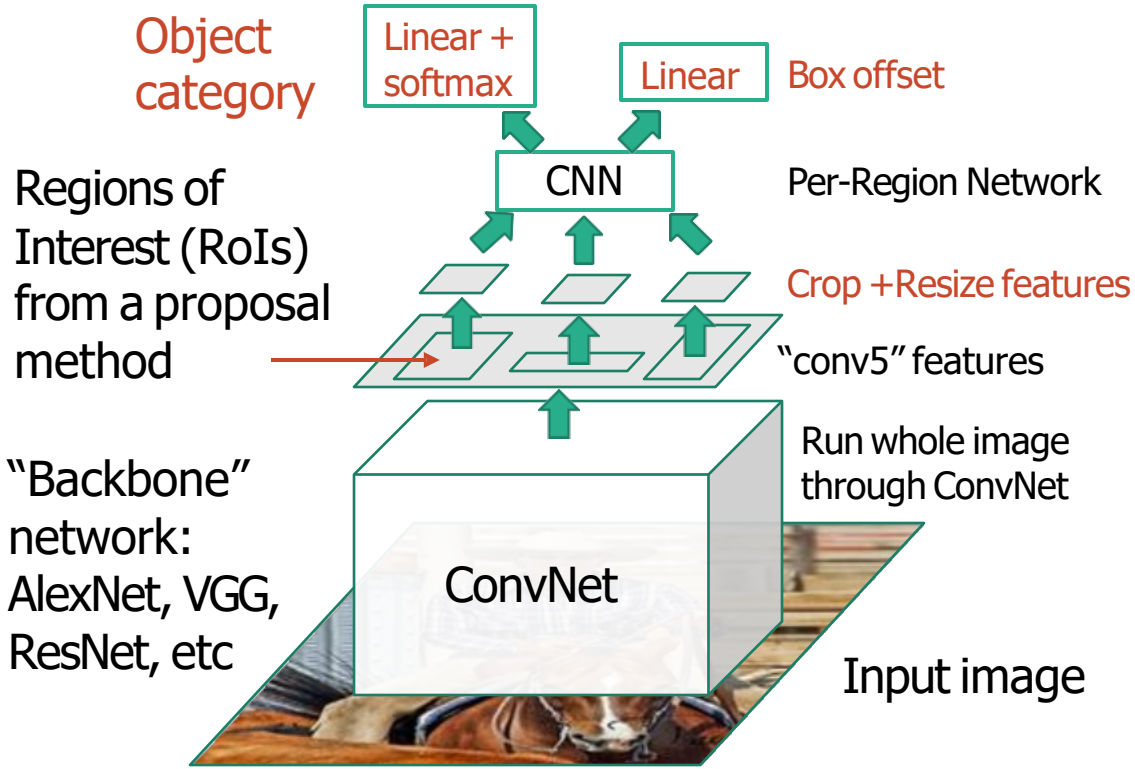


"Slow" R-CNN

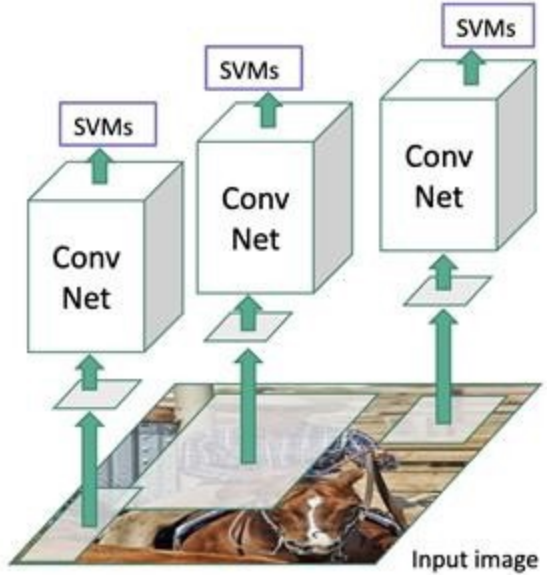


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN

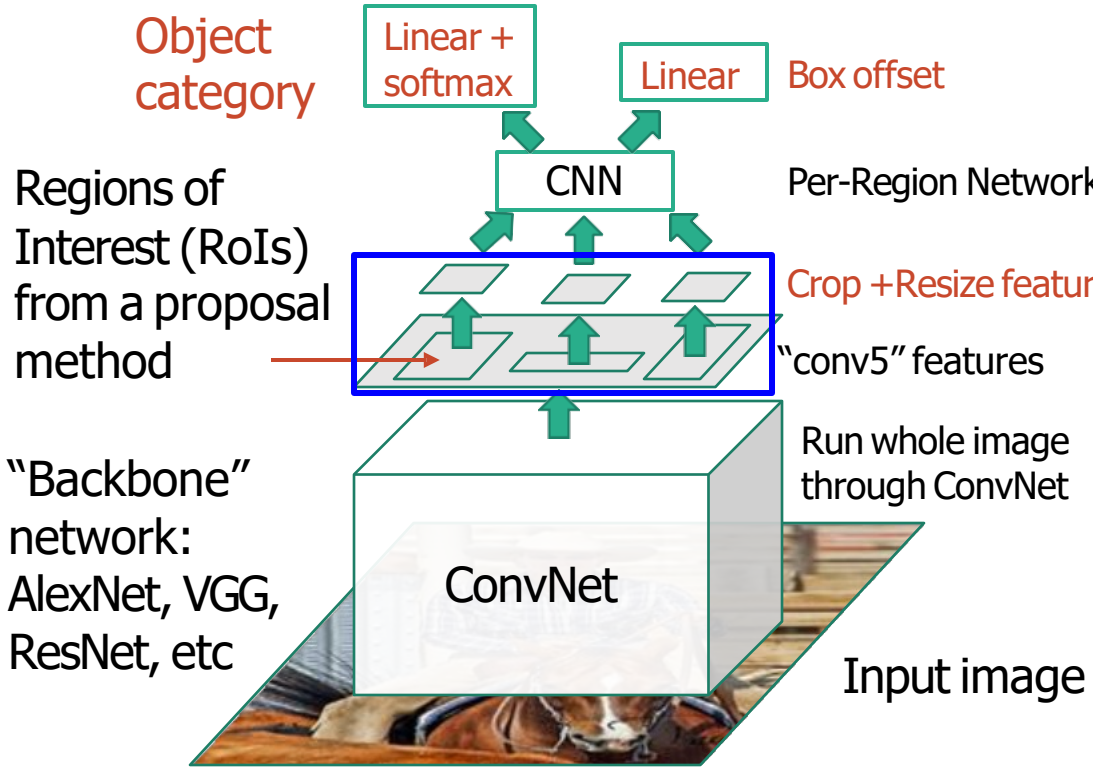


"Slow" R-CNN

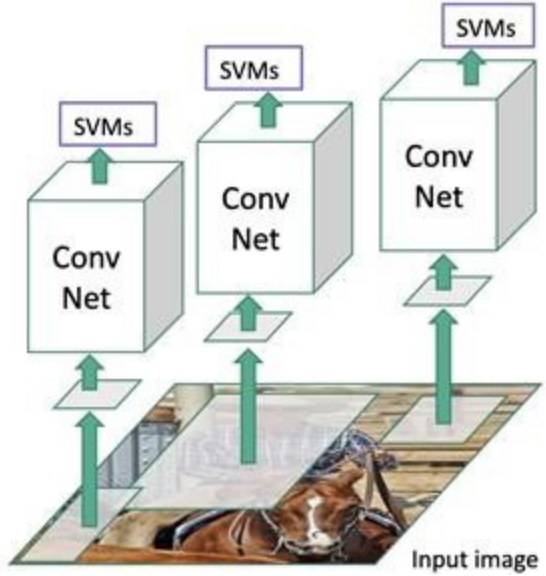


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



"Slow" R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

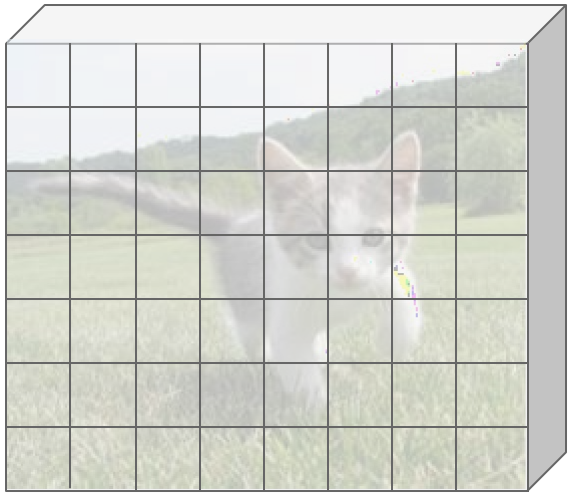


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

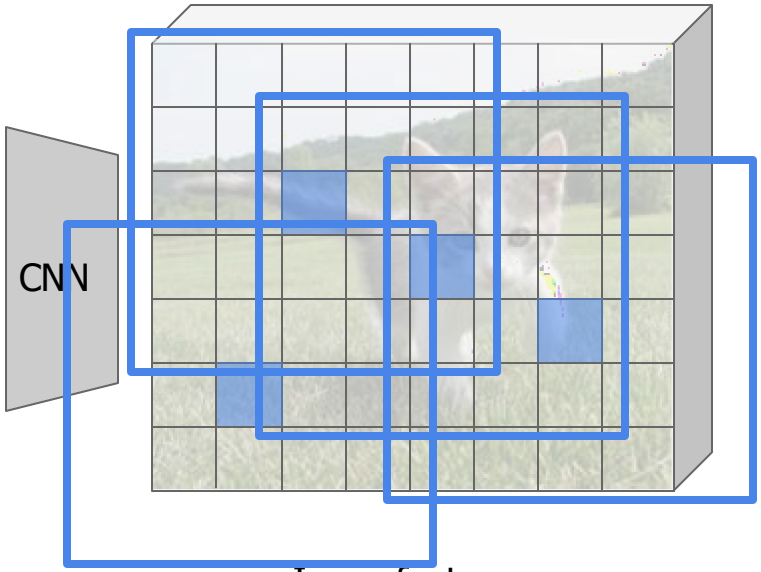


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

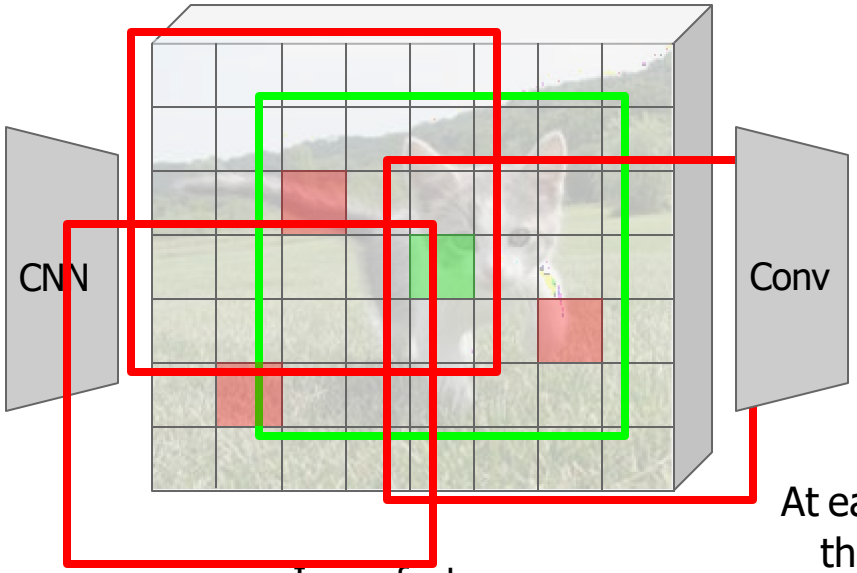


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map

Input Image
(e.g. 3 x 640 x 480)

CNN

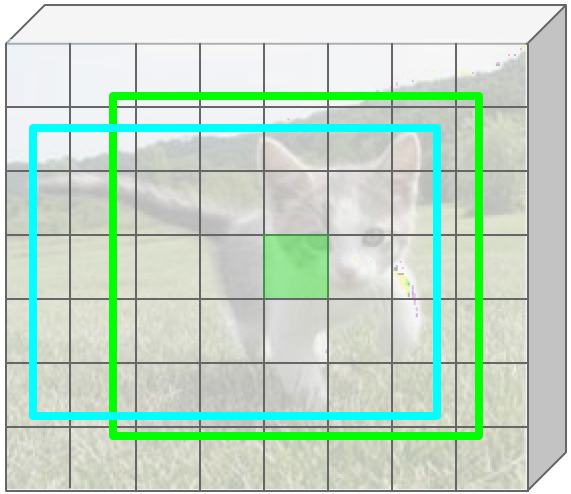


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

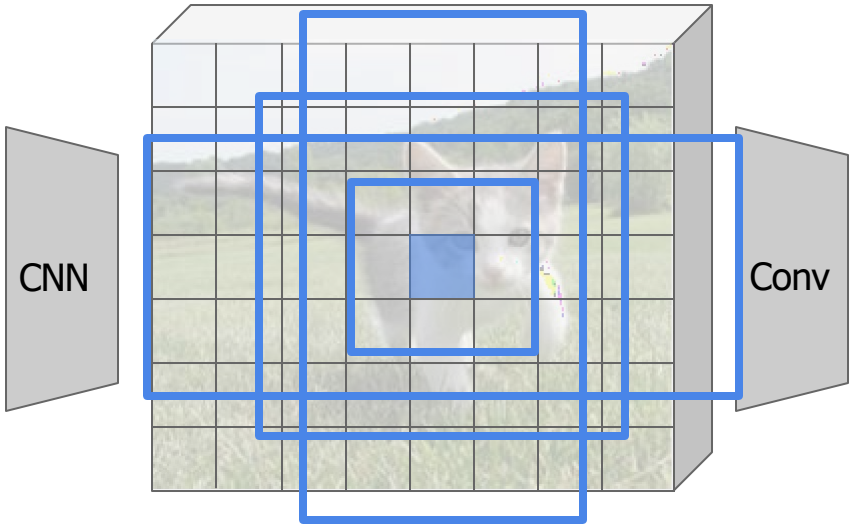


Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

Input Image
(e.g. 3 x 640 x 480)

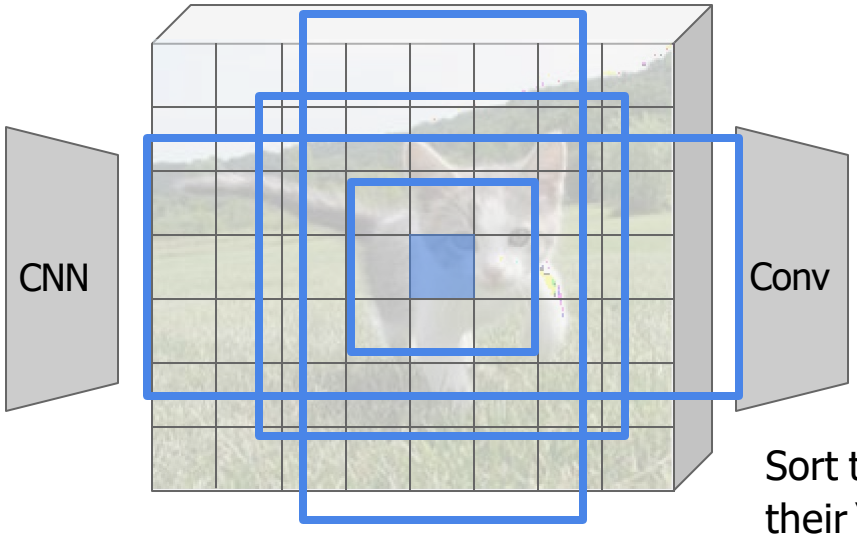
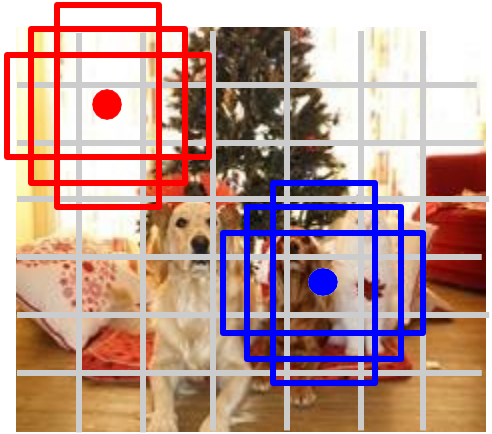


Image features
(e.g. 512 x 20 x 15)

Sort the K*20*15 boxes by their "objectness" score, take top ~300 as our proposals

Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of base boxes centered at each grid cell
Here B =3

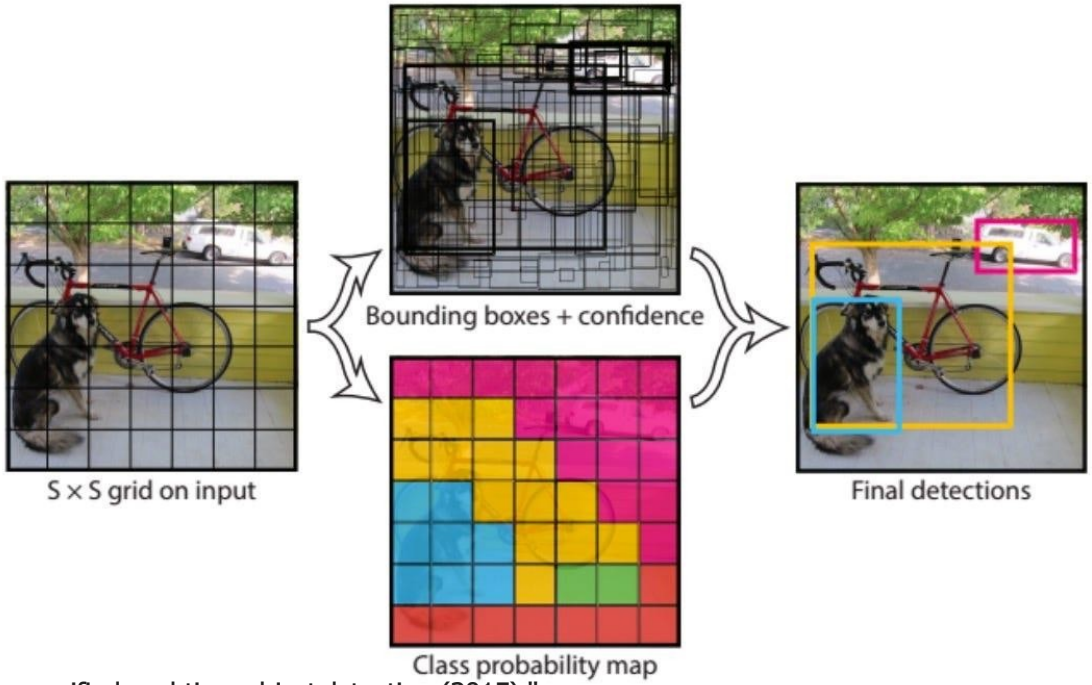
Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B +C)

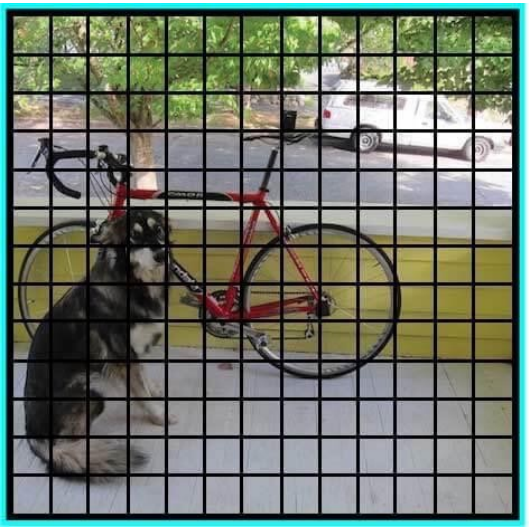
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

YOLO (You Only Look Once) real-time object detection



Redmon et al. "You only look once: unified, real-time object detection (2015)."

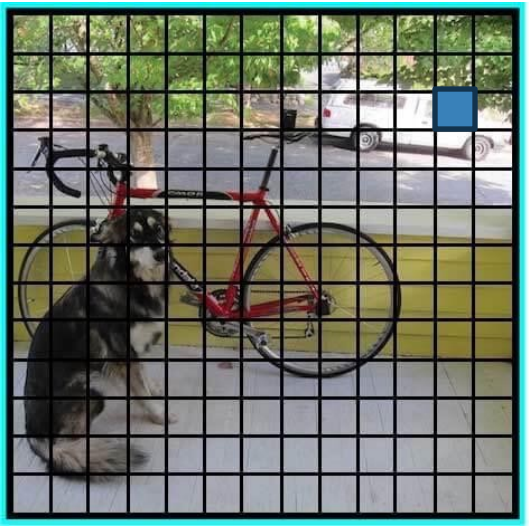
YOLO



SxS Grid

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



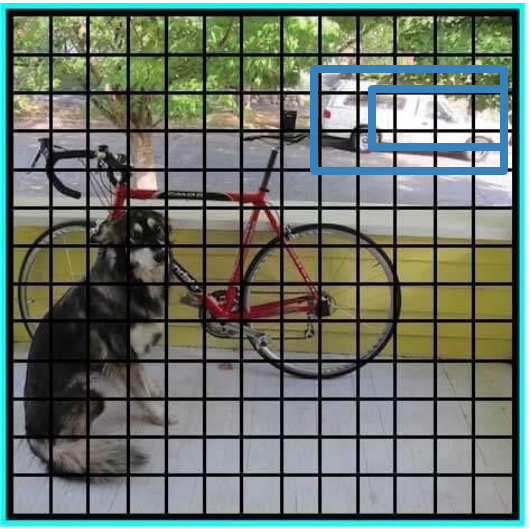
SxS Grid

For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

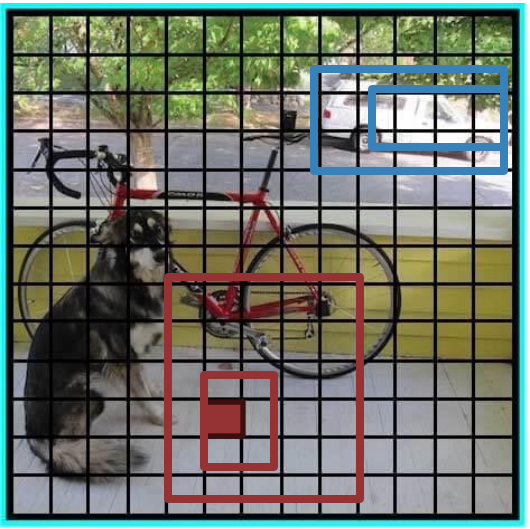
For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

B=2

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

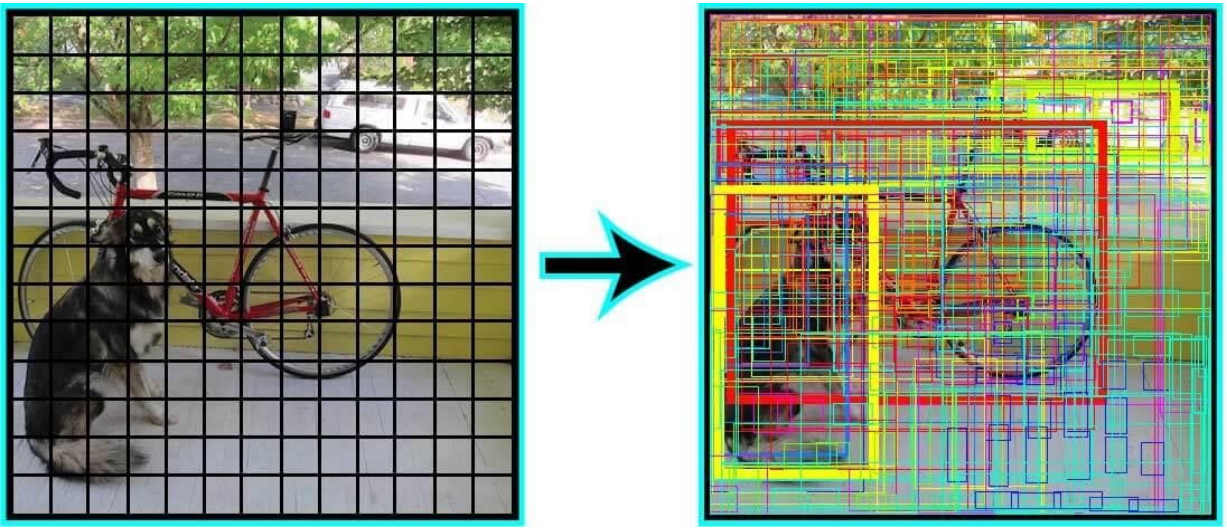
For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

B=2

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO

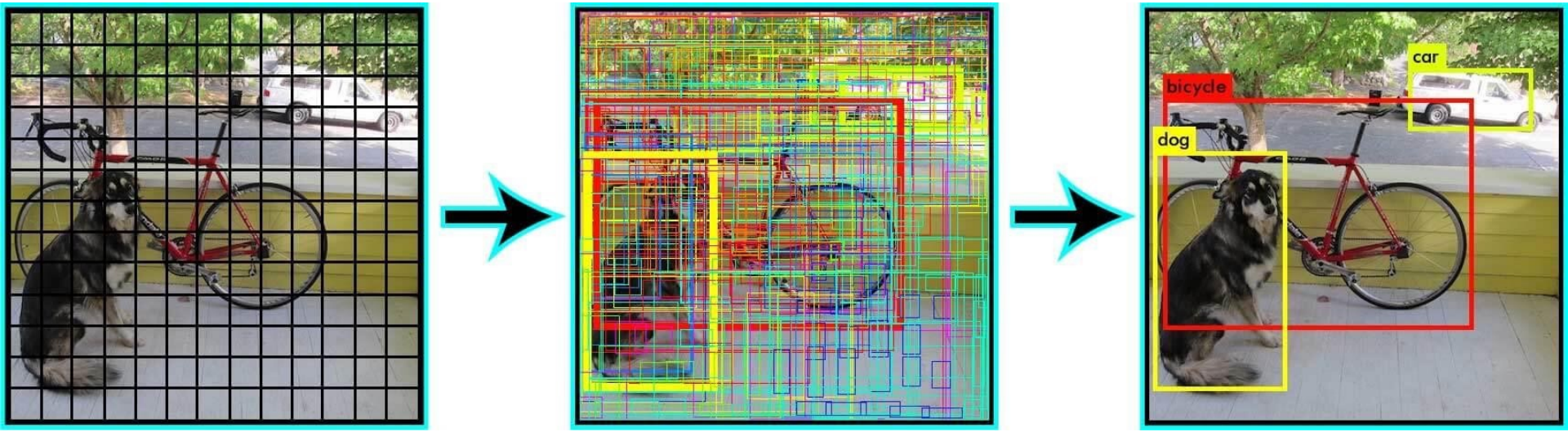


SxS Grid

Many bounding boxes with different object probabilities

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

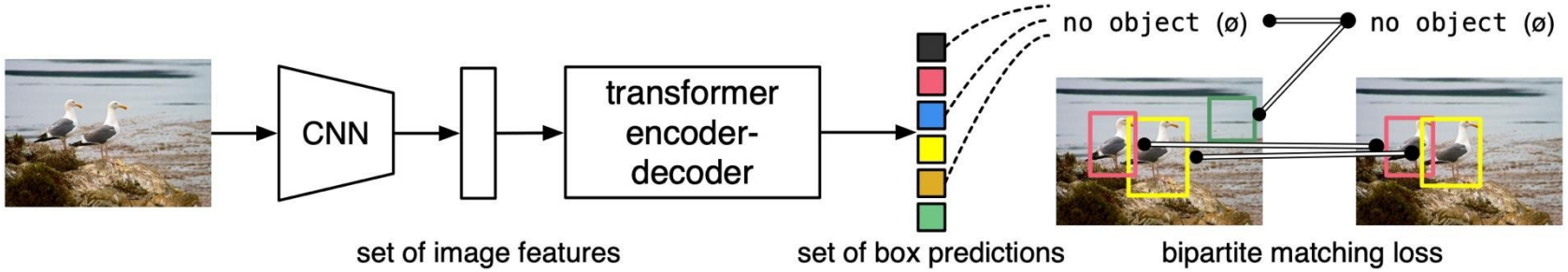
Redmon et al. "You only look once: unified, real-time object detection (2015)."

Object **De**tection with **Tr**ansformers: DETR

Simple object detection pipeline: directly output a set of boxes from a Transformer

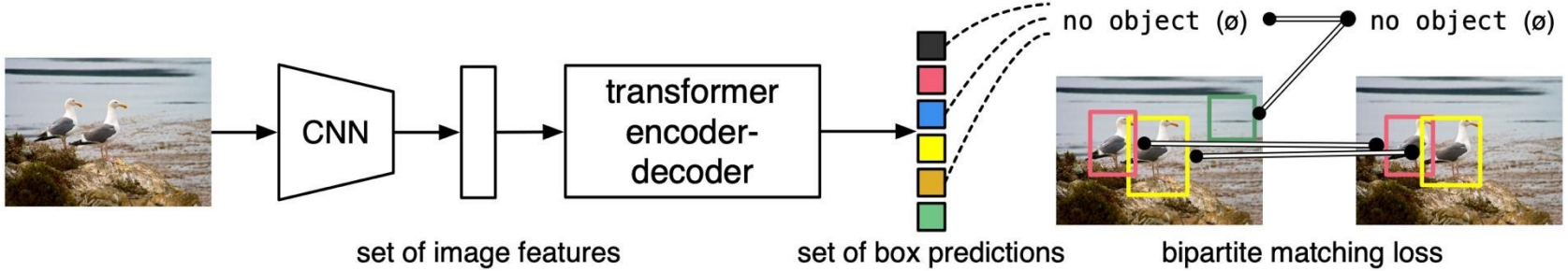
No anchors, no regression of box transforms

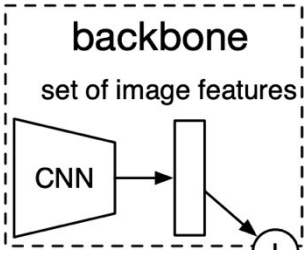
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates



Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

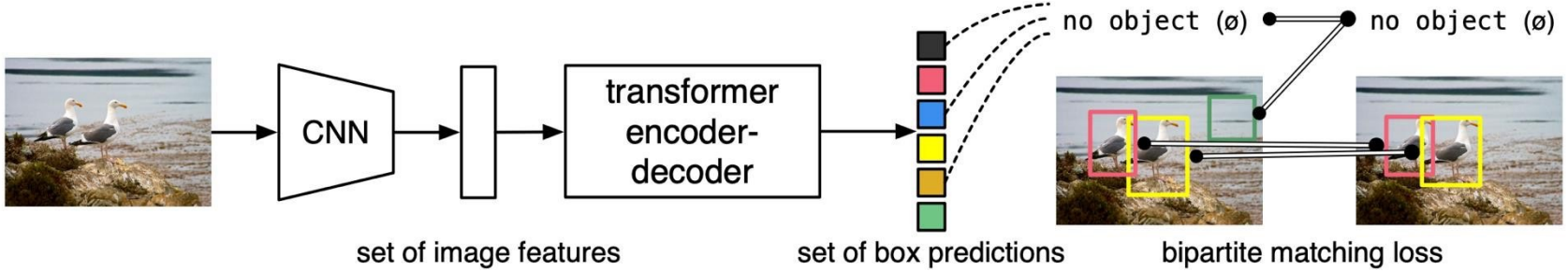
Object Detection with Transformers: DETR

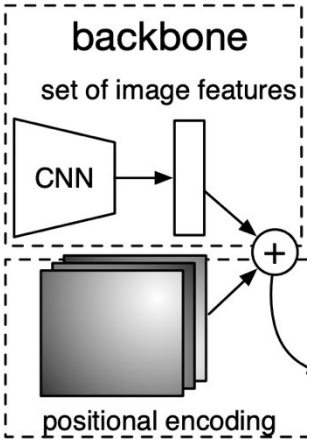




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

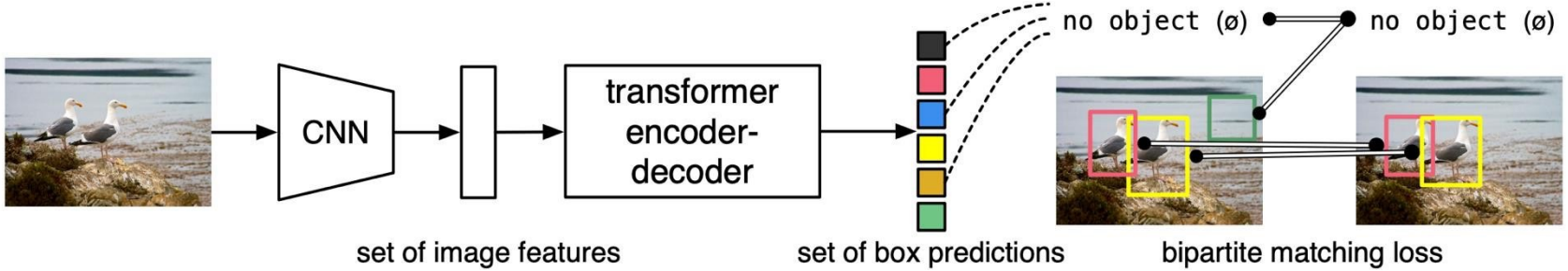
Object Detection with Transformers: DETR

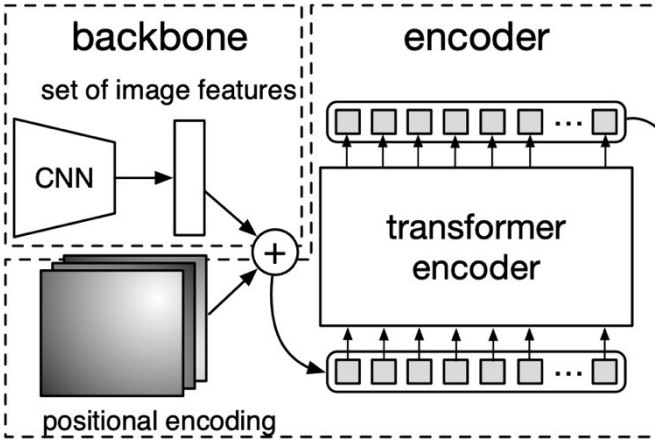




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

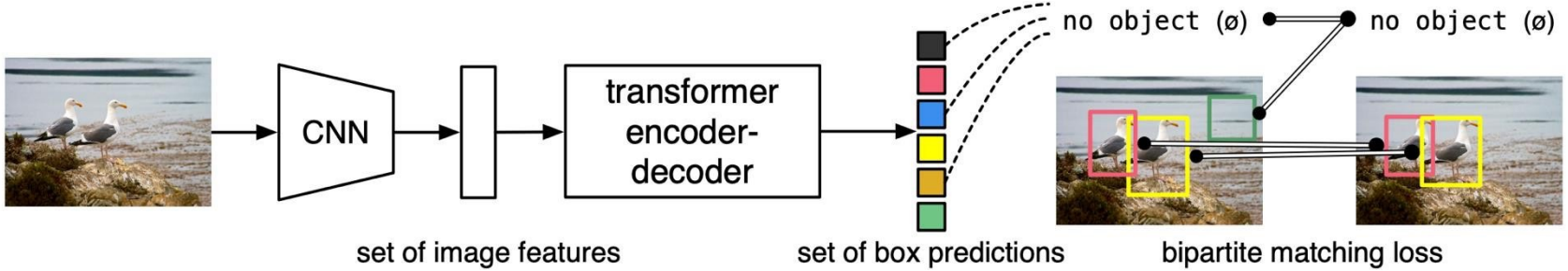
Object Detection with Transformers: DETR

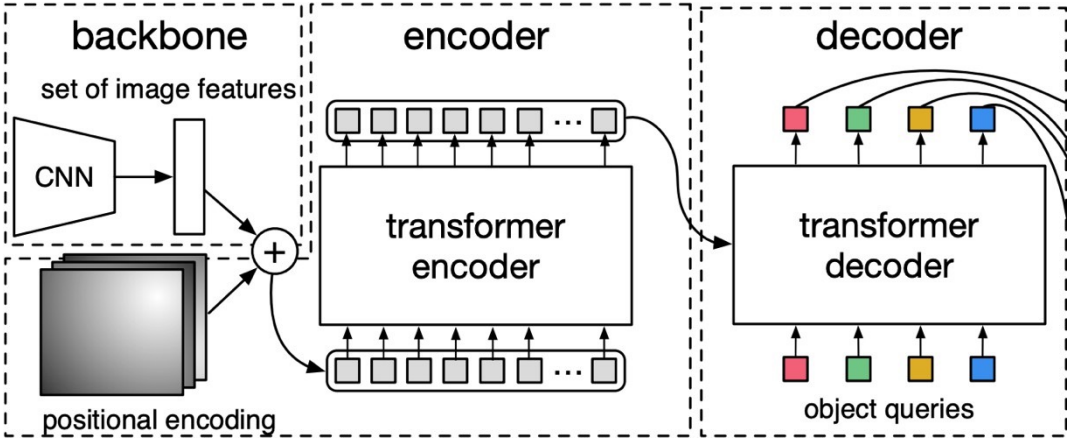




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

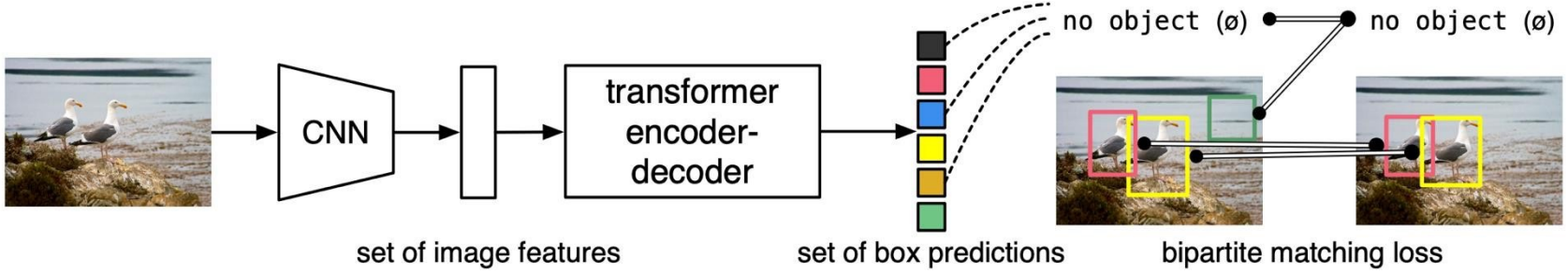
Object Detection with Transformers: DETR

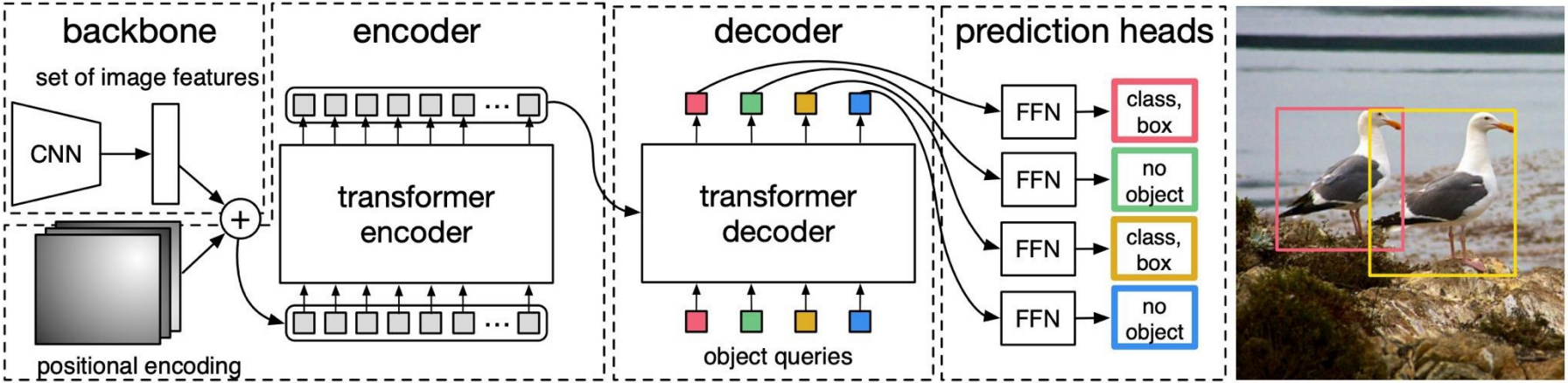




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Object Detection with Transformers: DETR





Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Instance Segmentation

Classification

CAT

No spatial extent

Semantic Segmentation

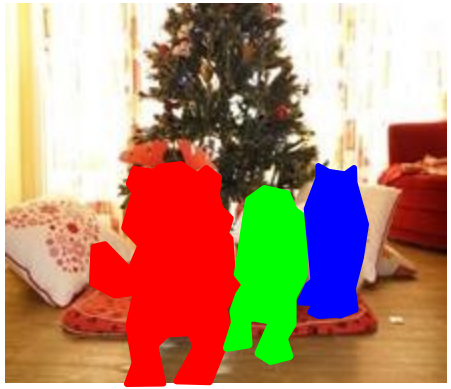
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation

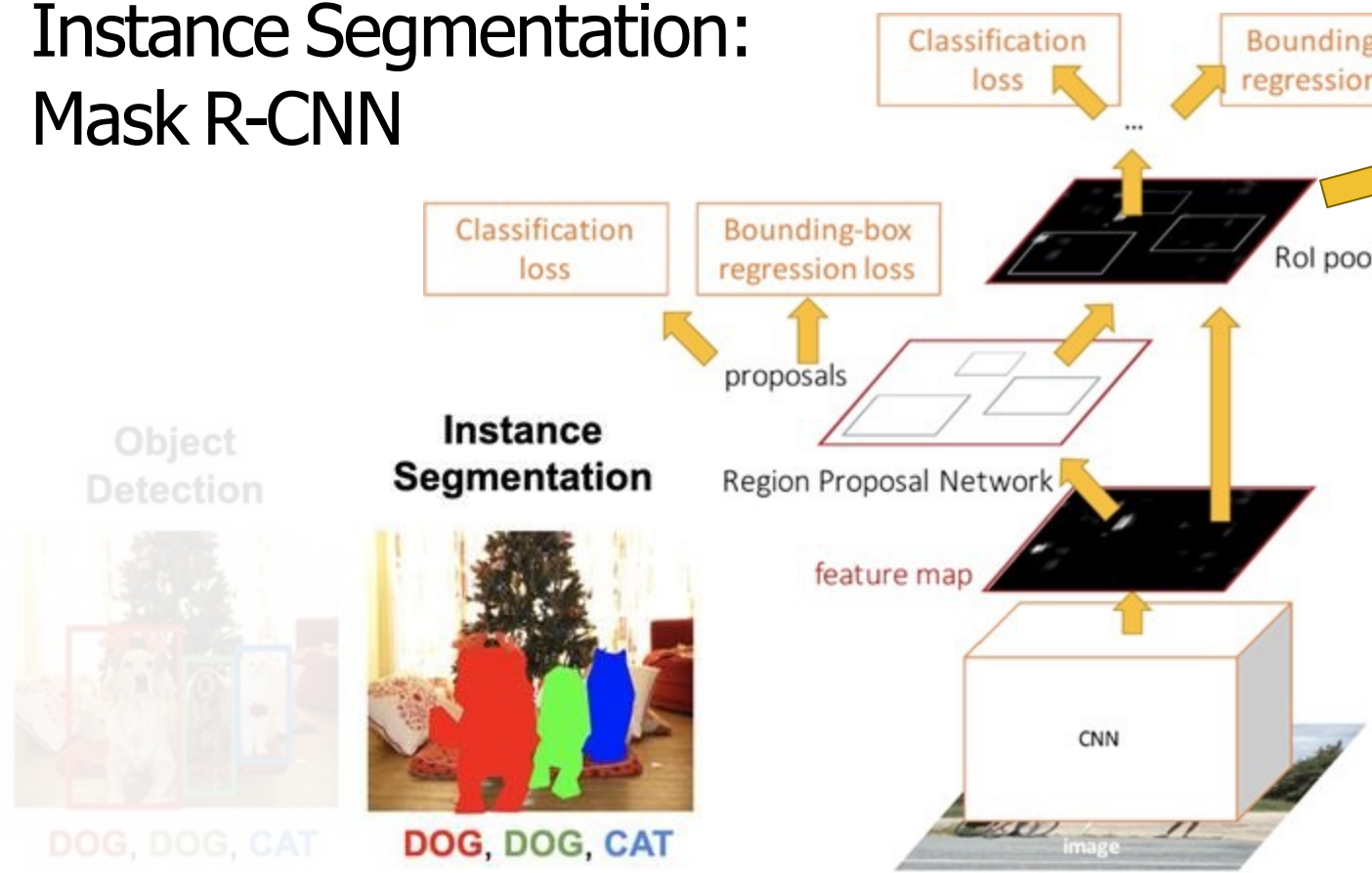


DOG, DOG, CAT

Multiple Object

Object Detection: Faster R-CNN

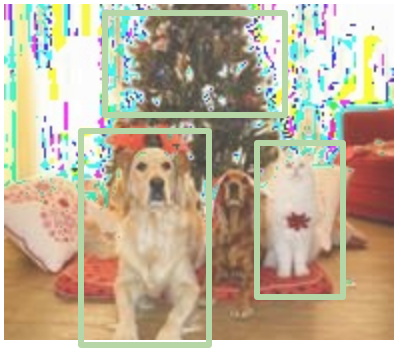
Instance Segmentation: Mask R-CNN



Add a small mask network that operates on each RoI and predicts a 28x28 binary mask

He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN



CNN
+RPN

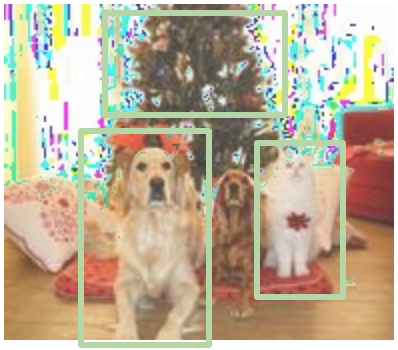
RoI Align

256 x 14 x 14

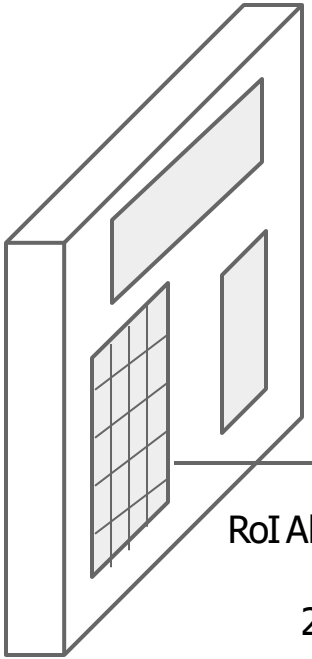
Classification Scores: C
Box coordinates (per class): 4 * C

He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN



CNN
+RPN



RoI Align

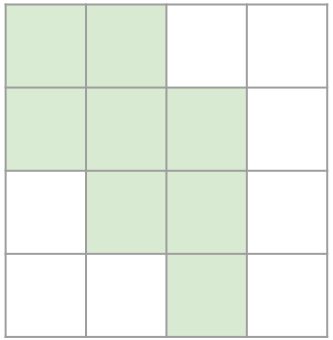
256 x 14 x 14

Classification Scores: C
Box coordinates (per class): 4 * C

Conv

256 x 14 x 14

Conv

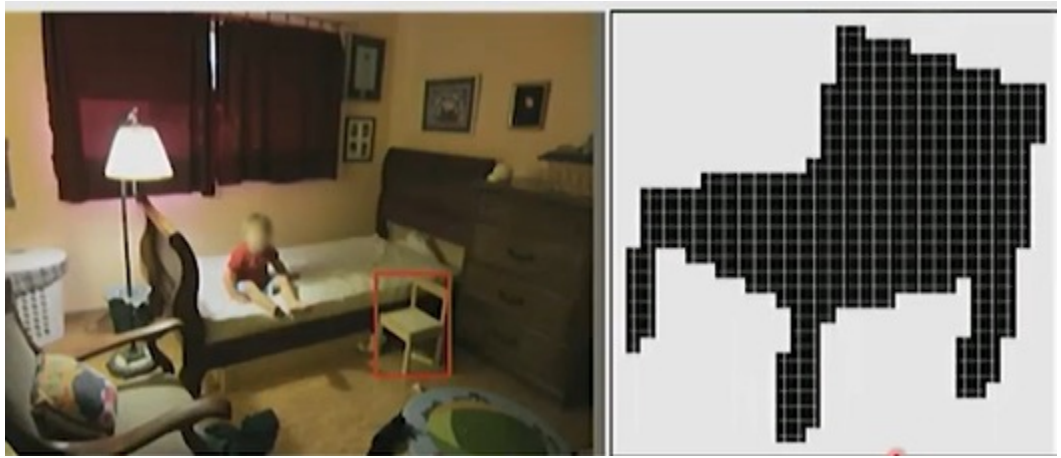


Predict a mask for
each of C classes

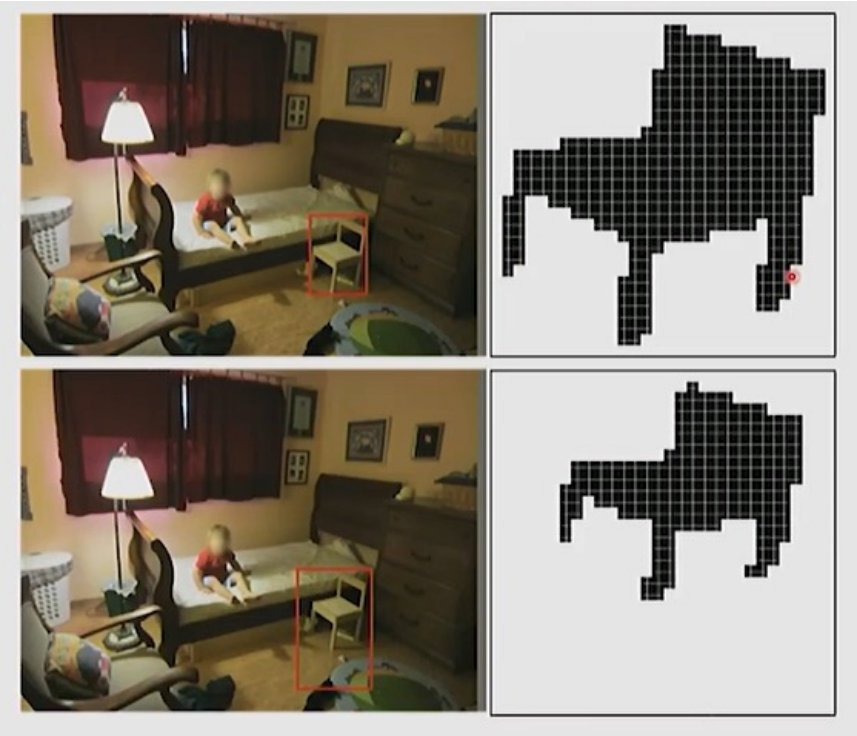
C x 28 x 28

He et al, "Mask R-CNN", arXiv 2017

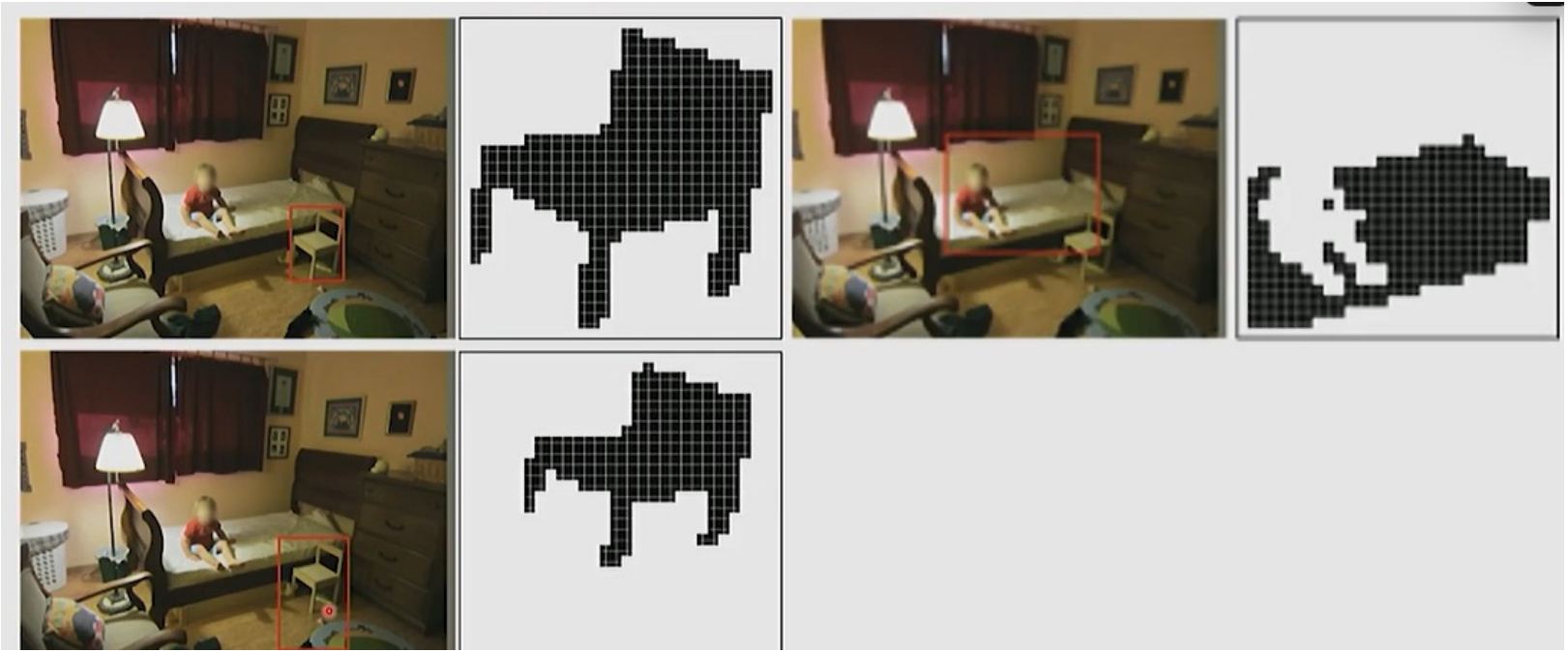
Mask R-CNN: Example Mask Training Targets



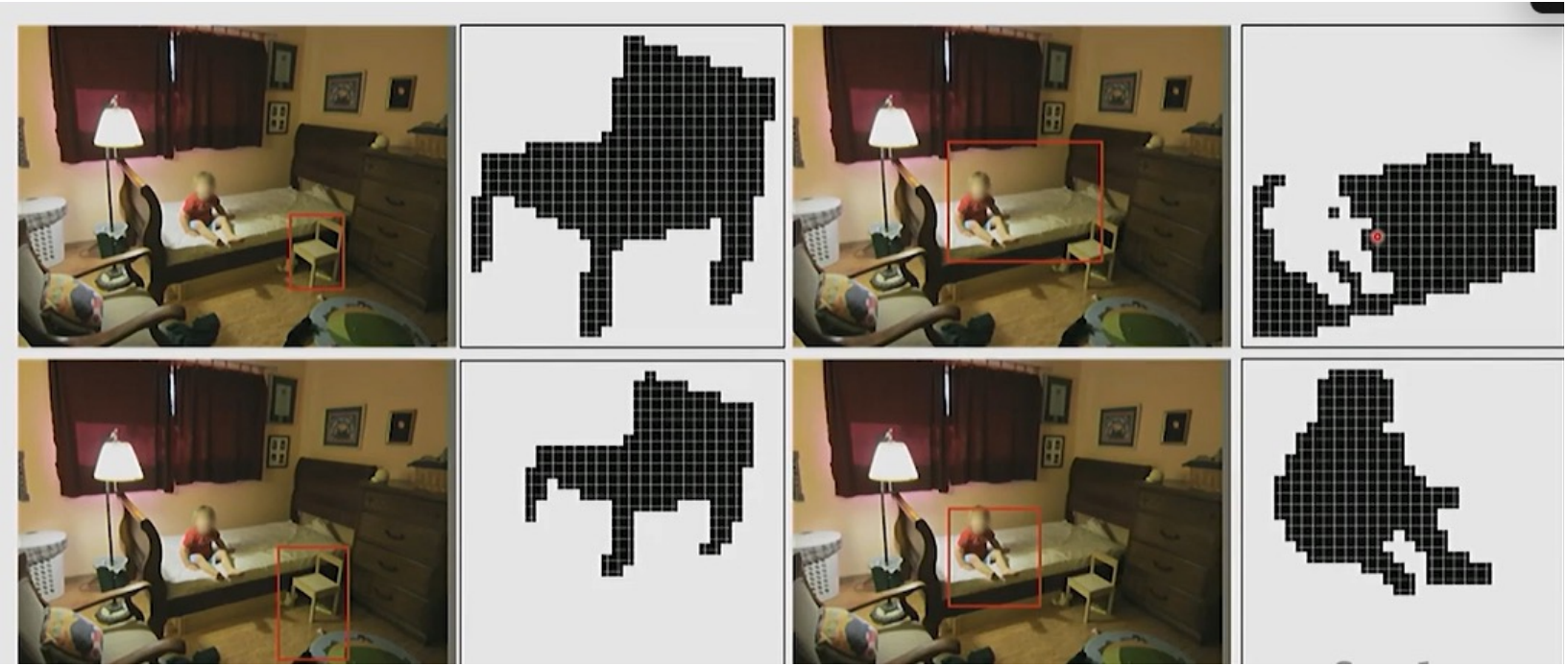
Mask R-CNN: Example Mask Training Targets



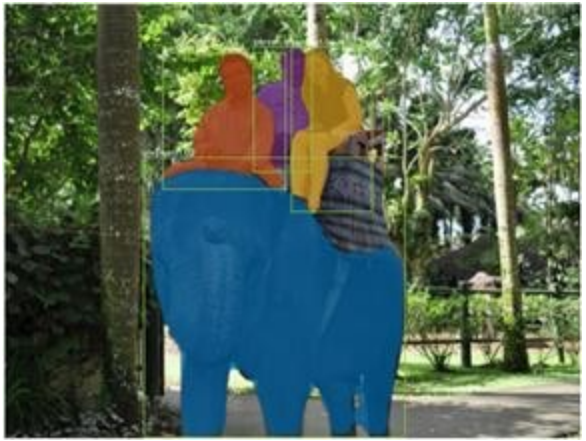
Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Example Mask Training Targets

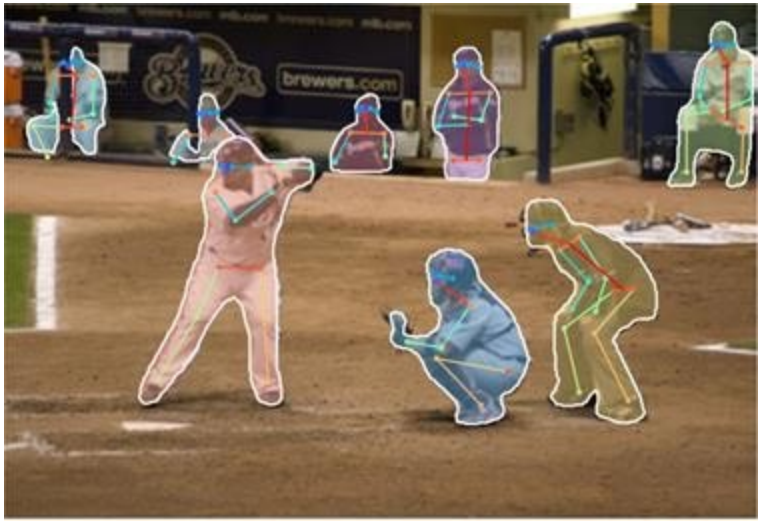


Mask R-CNN: Very Good Results!



He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN
Also does pose




He et al, "Mask R-CNN", ICCV 2017

Open Source Frameworks

Lots of good implementations on GitHub!

TensorFlow Detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection
Faster RCNN, SSD, RFCN, Mask R-CNN, ...

Detectron2 (PyTorch)
https://github.com/facebookresearch/detectron2
Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN, ...

Finetune on your own dataset with pre-trained models

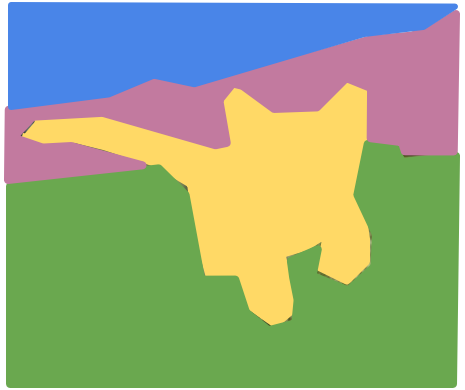
Recap: Lots of computer vision tasks!

Classification

CAT

No spatial extent

Semantic Segmentation



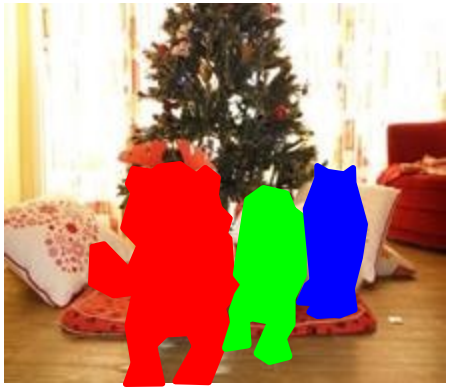
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

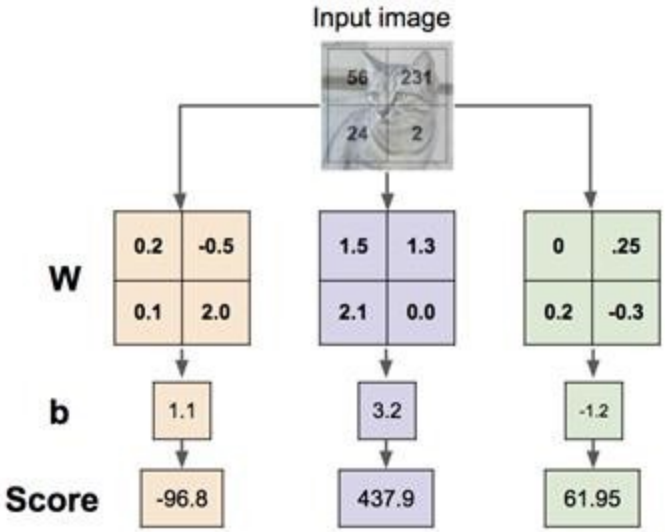
Today

- Transformers Recap

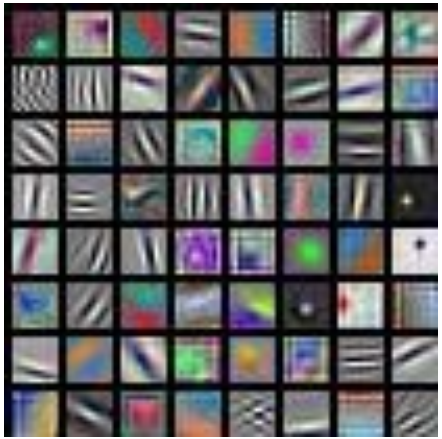
- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation

- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

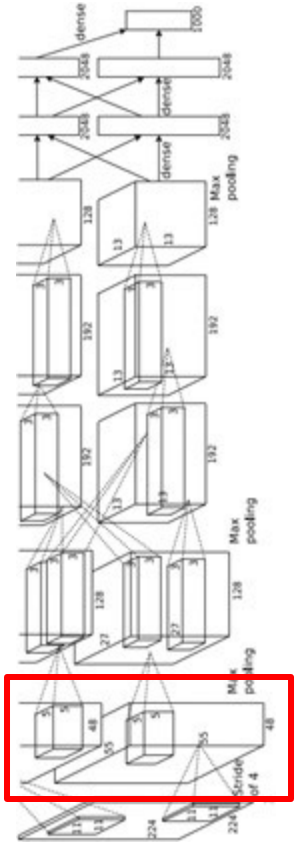
Interpreting a Linear Classifier: Visual Viewpoint



First Layer: Visualize Filters

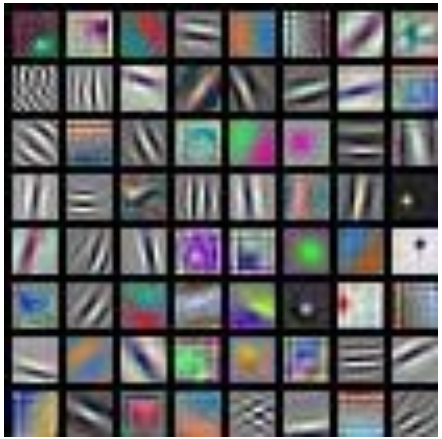


AlexNet:
64 x 3 x 11 x 11



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

First Layer: Visualize Filters

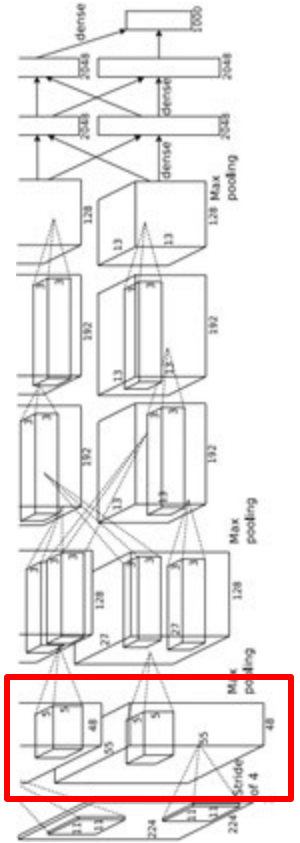


AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

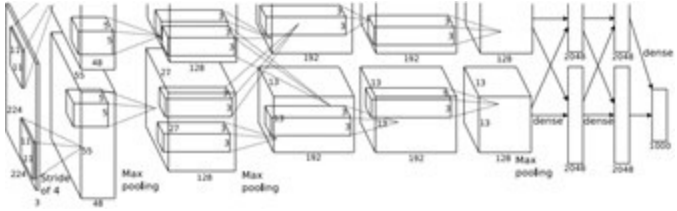
DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

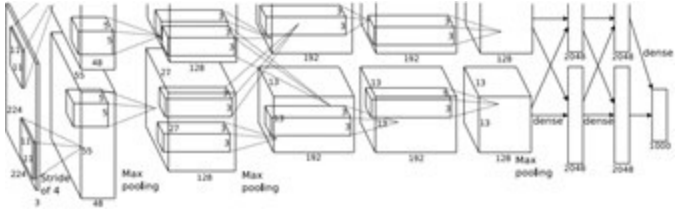


Dog

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

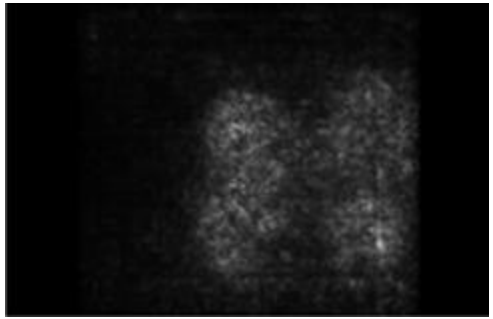
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



Dog

Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels

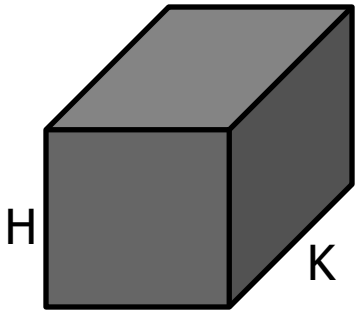


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Saliency Maps

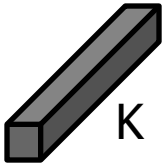
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Class Activation Mapping (CAM)



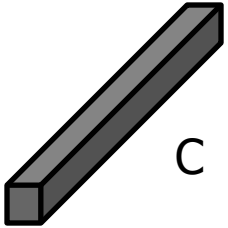
Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Global Average Pooling



Pooled features:
$F \in \mathbb{R}^{K}$

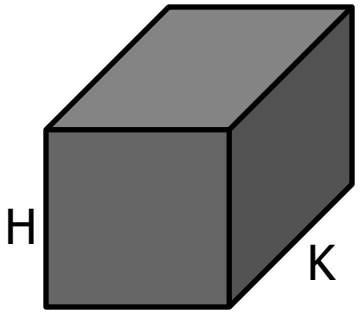
Fully Connected Layer, weights
$w \in \mathbb{R}^{K \times C}$



Class Scores:
$S \in \mathbb{R}^{C}$

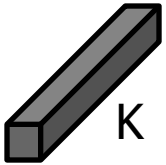
Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



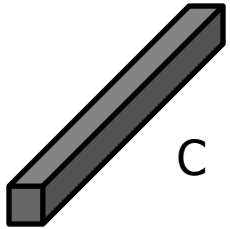
Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Global Average Pooling



Pooled features:
$F \in \mathbb{R}^{K}$

Fully Connected Layer, weights
$w \in \mathbb{R}^{K \times C}$

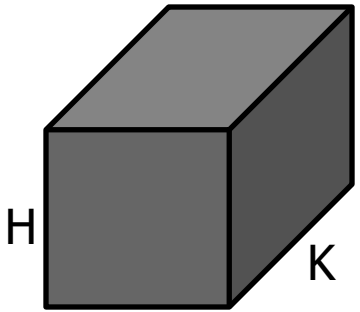


Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

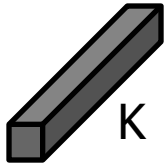
Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



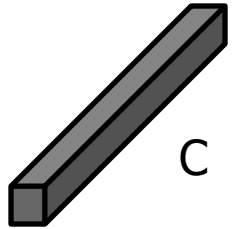
Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Global Average Pooling



Pooled features:
$F \in \mathbb{R}^{K}$

Fully Connected Layer, weights
$w \in \mathbb{R}^{K \times C}$

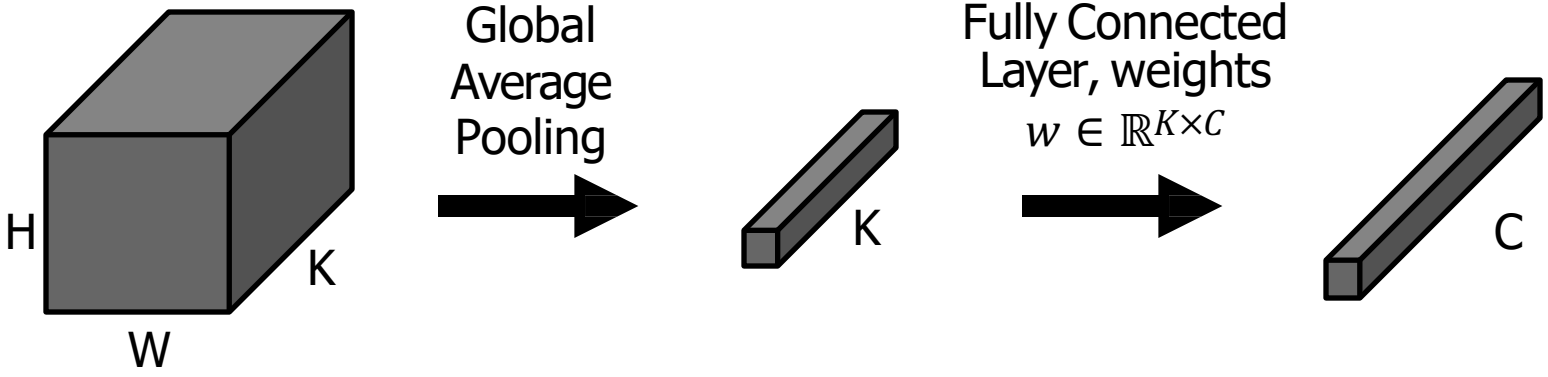


Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_{k} w_{k,c} F_k$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

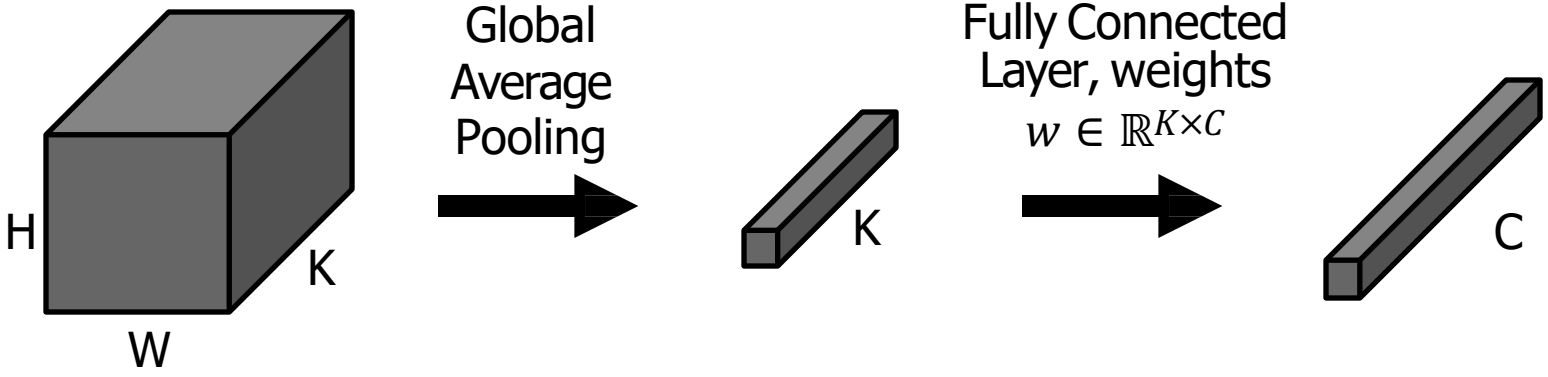
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

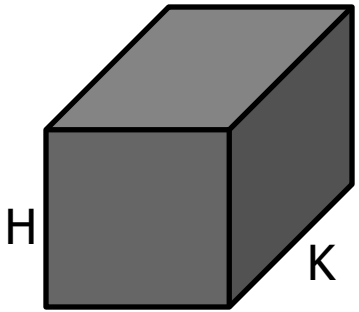
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$
$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

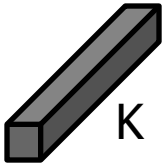
Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



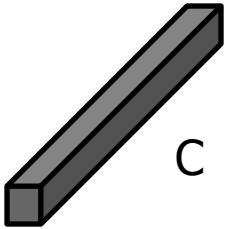
Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Global Average Pooling



Pooled features:
$F \in \mathbb{R}^{K}$

Fully Connected Layer, weights
$w \in \mathbb{R}^{K \times C}$



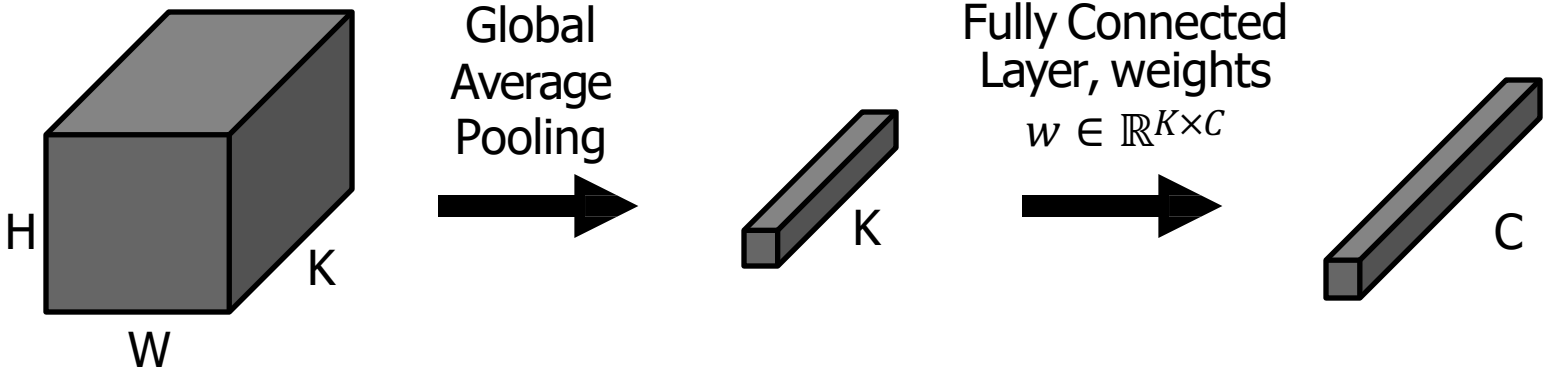
Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

$$= \frac{1}{HW} \sum_{h,w} \sum_k \boldsymbol{w_{k,c} f_{h,w,k}}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

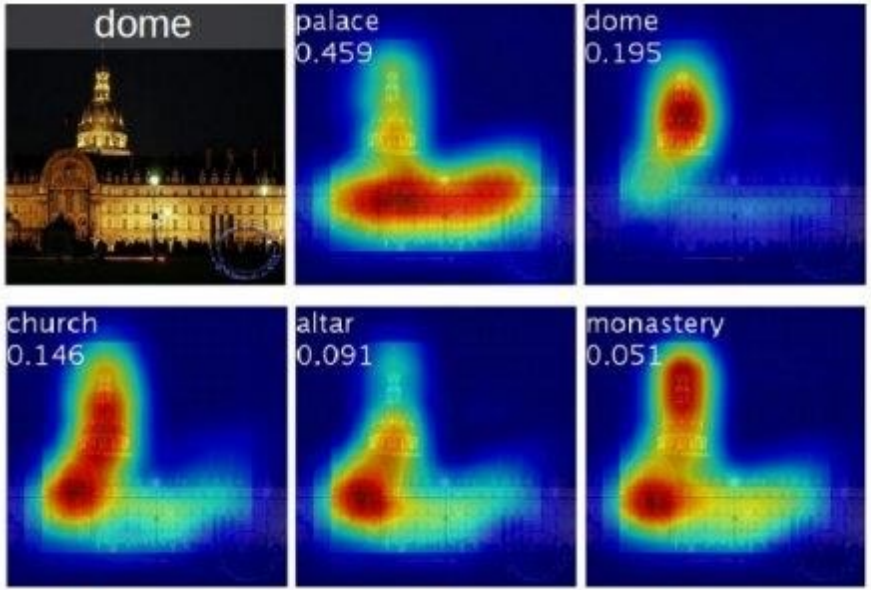
$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

Class Activation Maps:
$\boldsymbol{M} \in \mathbb{R}^{C,H,W}$

$$\boldsymbol{M_{c,h,w}} = \sum_k \boldsymbol{w_{k,c} f_{h,w,k}}$$

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)



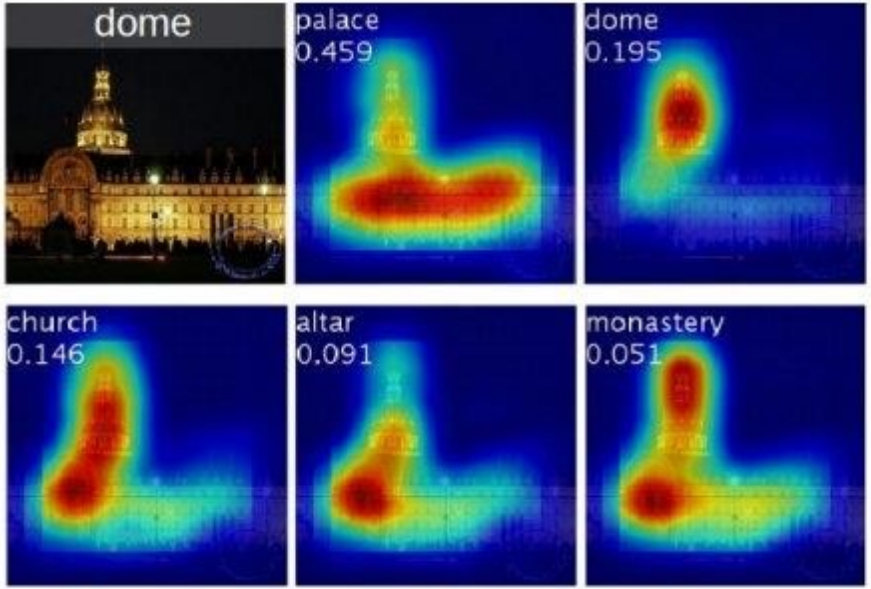
Class activation maps of top 5 predictions

Class activation maps for one object class

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)

Problem: Can only apply to last conv layer



Class activation maps of top 5 predictions

Class activation maps for one object class

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

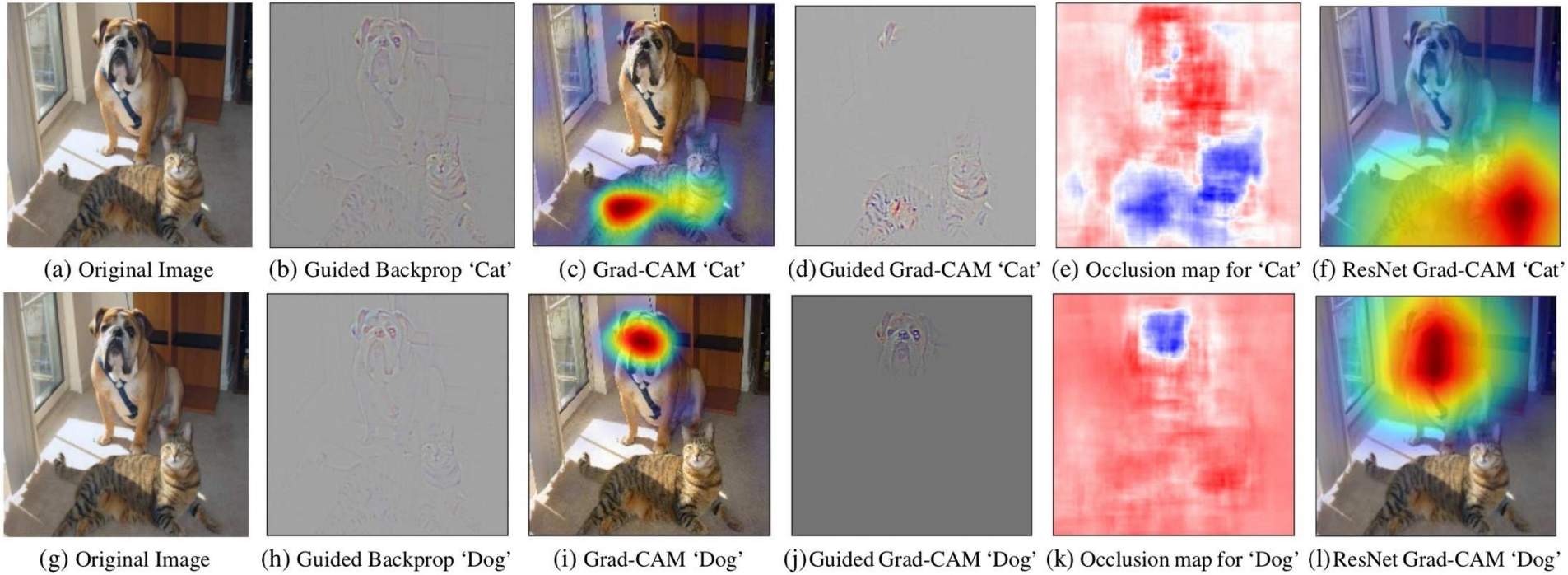
$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

4. Compute activation map $M^c \in \mathbb{R}^{H,W}$:

$$M^c_{h,w} = ReLU\left(\sum_k \alpha_k A_{h,w,k} \right)$$

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

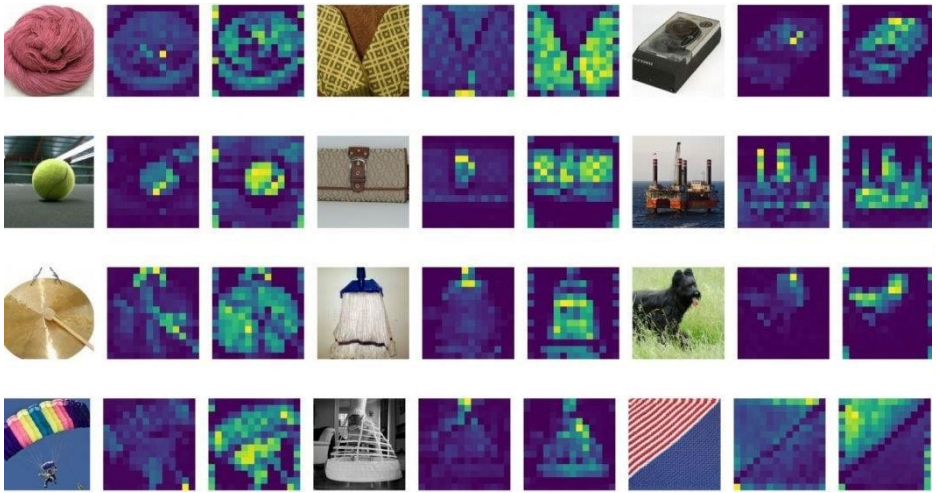


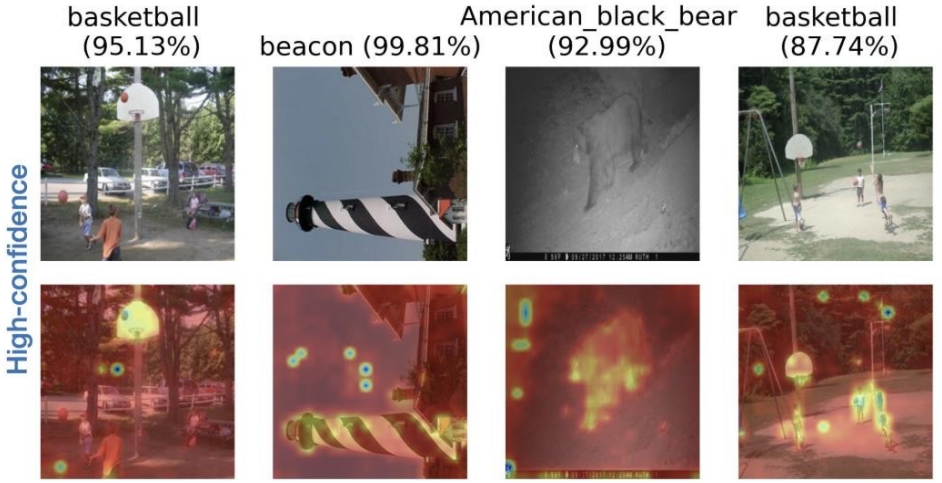
(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat' (e) Occlusion map for 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j) Guided Grad-CAM 'Dog' (k) Occlusion map for 'Dog' (l) ResNet Grad-CAM 'Dog'

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Visualizing ViT features





Chen et al., When Vision Transformers OutperformResnets Without Pre-training Or Strong Data Augmentations, ICLR 2022; Paul and Chen, Vision Transformers are Robust Learners, AAAI 2022. Reproduced for educational purposes.

Today

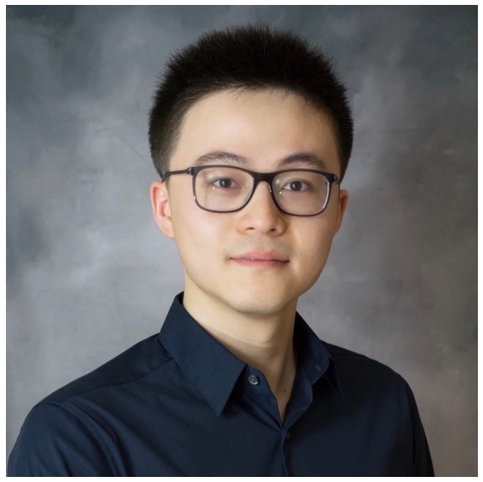
- Transformers Recap

- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation

- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

Lecture 10:
Video Understanding

Instructor today



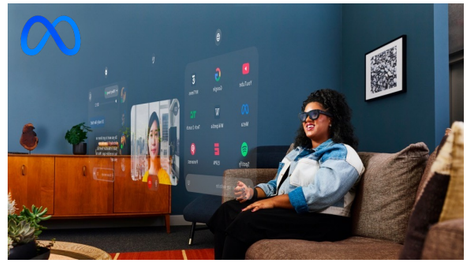
Ruohan Gao
https://ruohangao.github.io/

Ph.D. at UT Austin

I taught CS231N at Stanford from 2021-2023

Postdoc at Stanford

Some time at Meta



Teaching multimodal compute vision now.

University of Maryland, College Park

Recall: (2D) Image classification

This image by Nikita is licensed under CC-BY 2.0

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

→ cat

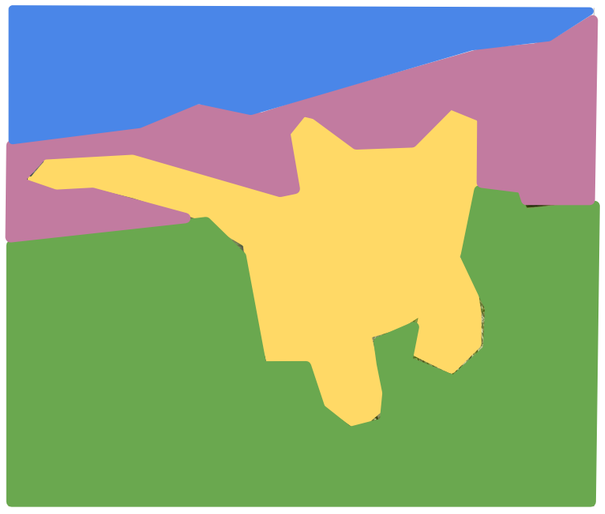
Last Lecture: (2D) Detection and Segmentation

Classification

CAT

No spatial extent

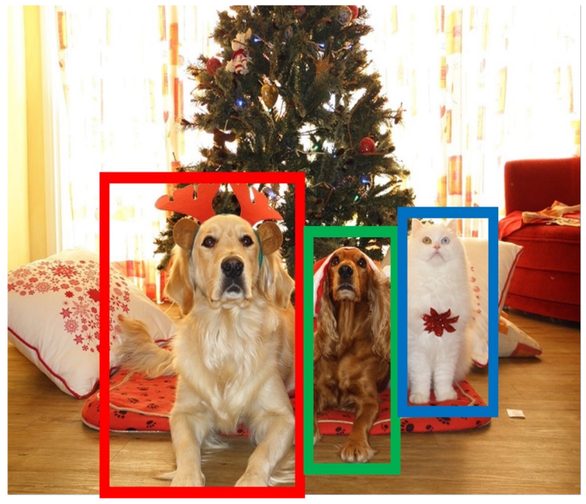
Semantic Segmentation



GRASS, CAT, TREE, SKY

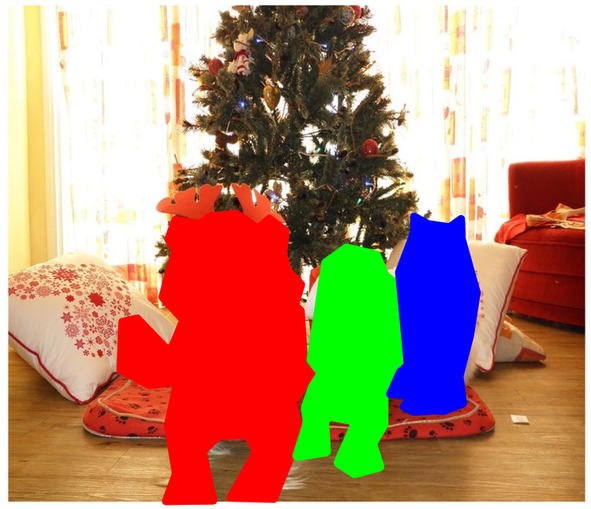
No objects, just pixels

Object Detection



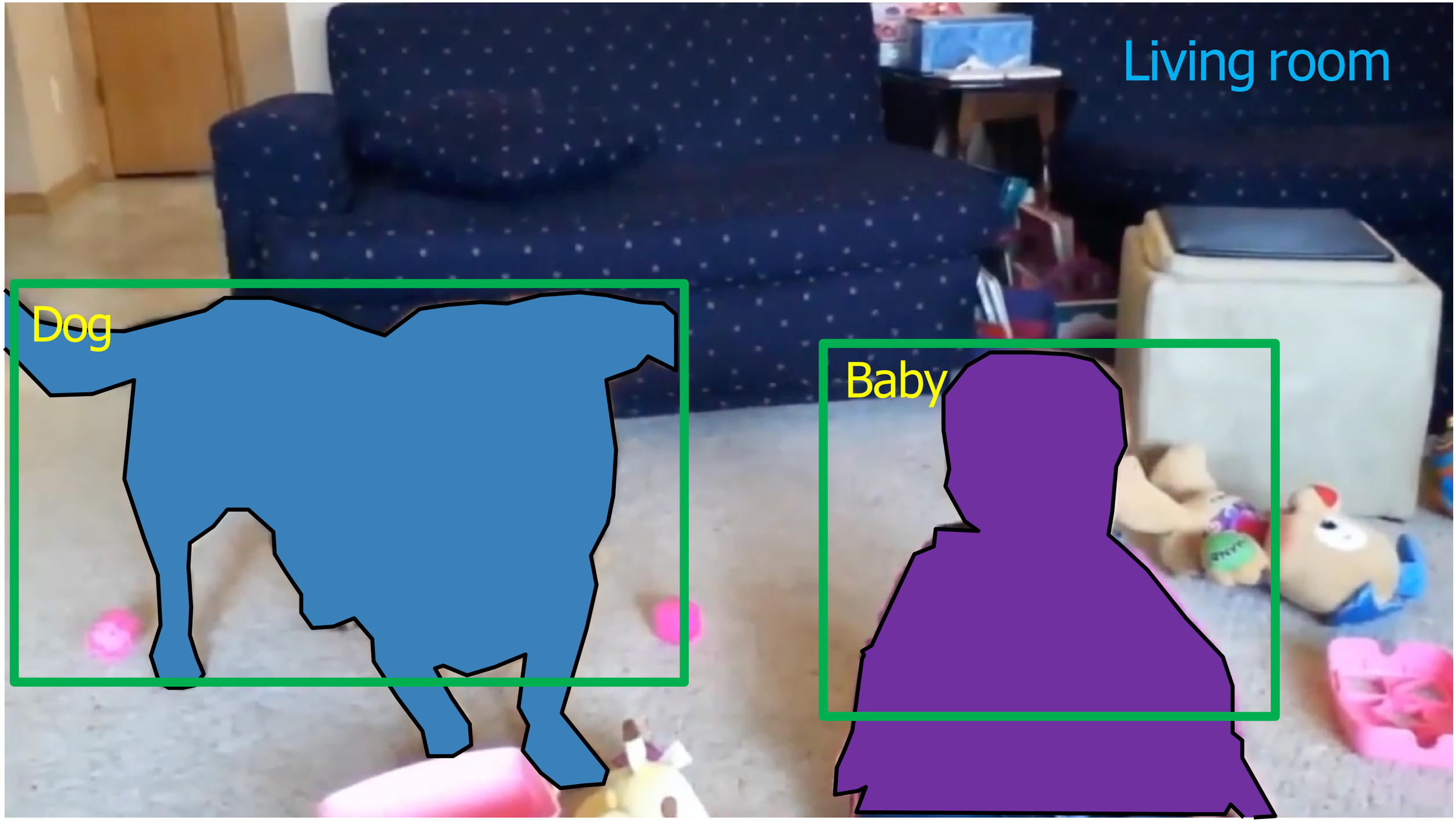
DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Objects


Living room
Dog
Baby

Today: Video = 2D + Time

A video is a sequence of images
4D tensor: T x 3 x H x W
(or 3 x T x H x W)

This image is CC0 public domain

Example task: Video Classification

Input video:
T x 3 x H x W

Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

Example task: Video Classification

Images: Recognize objects

Dog
Cat
Fish
Truck

Videos: Recognize actions

Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

Problem: Videos are big!

Videos are ~30 frames per second (fps)

Input video:
T x 3 x H x W

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): ~1.5 GB per minute
HD (1920 x 1080): ~10 GB per minute

Problem: Videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): ~1.5 GB per minute
HD (1920 x 1080): ~10 GB per minute

Solution: Train on short clips: low fps and low spatial resolution
e.g. T = 16, H=W=112
(3.2 seconds at 5 fps, 588 KB)

Training on Clips

Raw video: Long, high FPS

Training on Clips

Raw video: Long, high FPS

Training: Train model to classify short clips with low FPS

Training on Clips

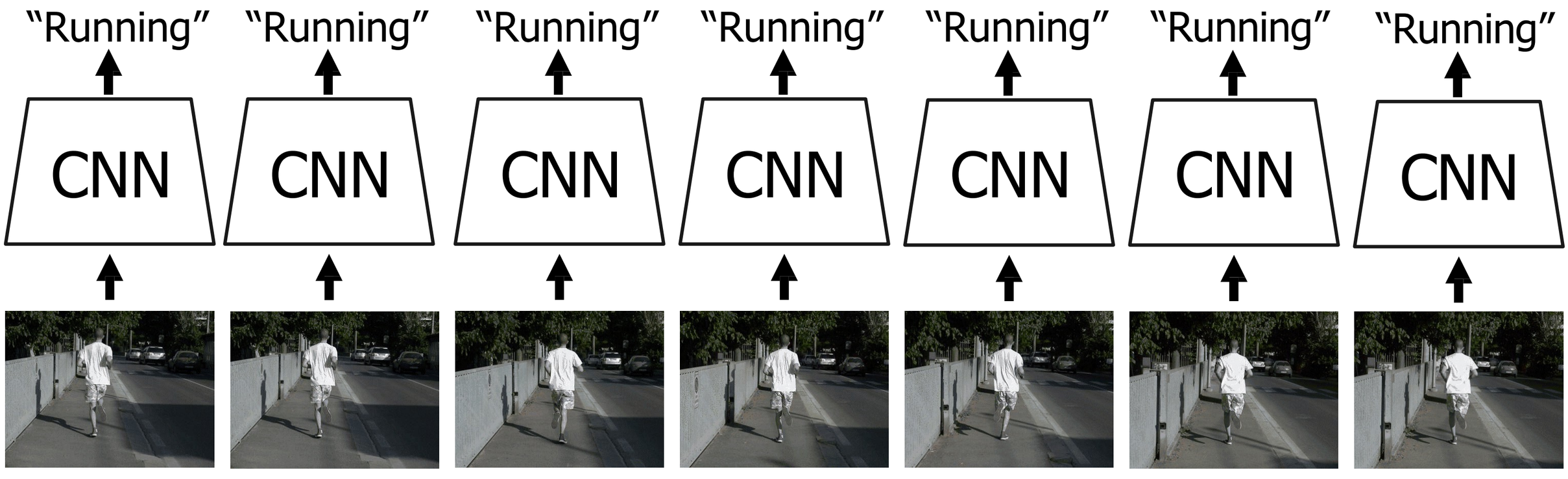
Raw video: Long, high FPS

Training: Train model to classify short clips with low FPS

Testing: Run model on different clips, average predictions

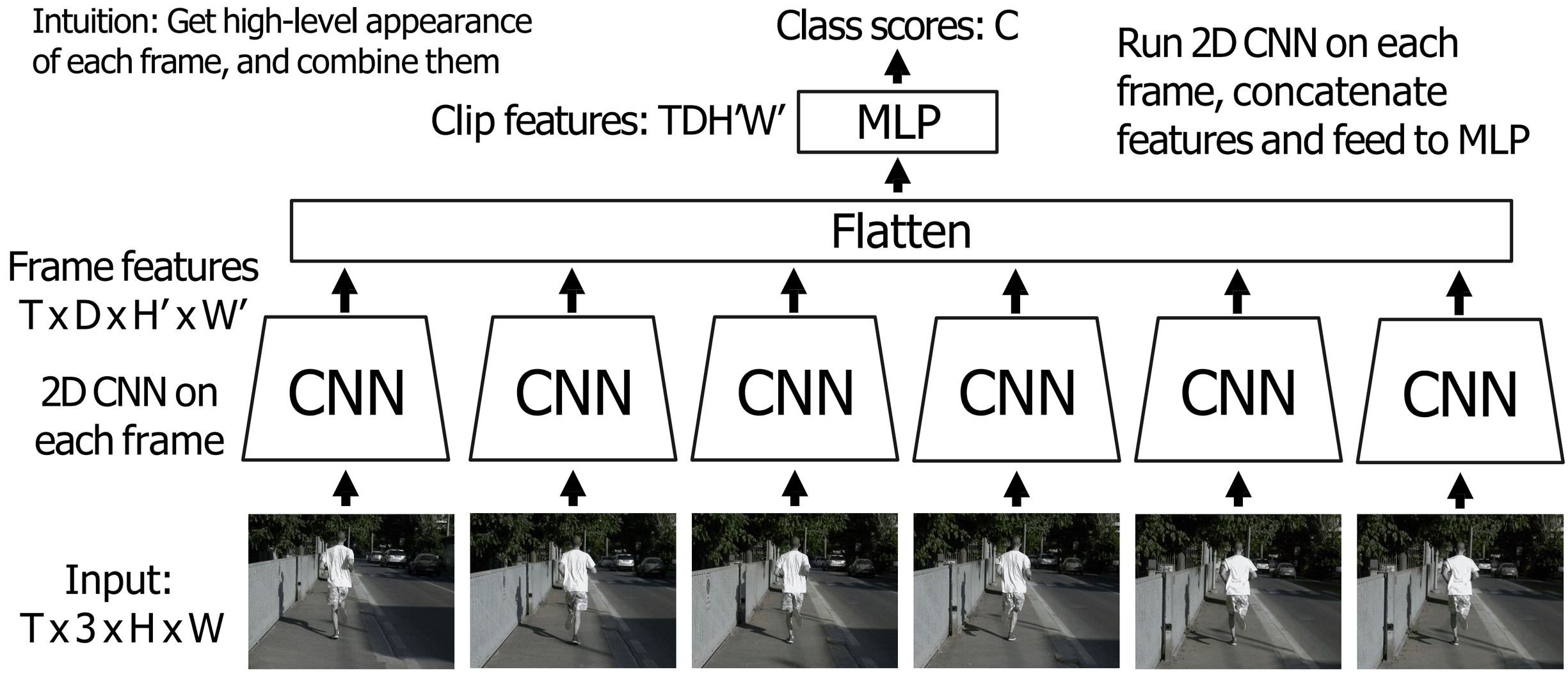
Video Classification: Single-Frame CNN

Simple idea: train normal 2D CNN to classify video frames independently!
(Average predicted probs at test-time)
Often a very strong baseline for video classification



Video Classification: Late Fusion (with FC layers)

Intuition: Get high-level appearance of each frame, and combine them

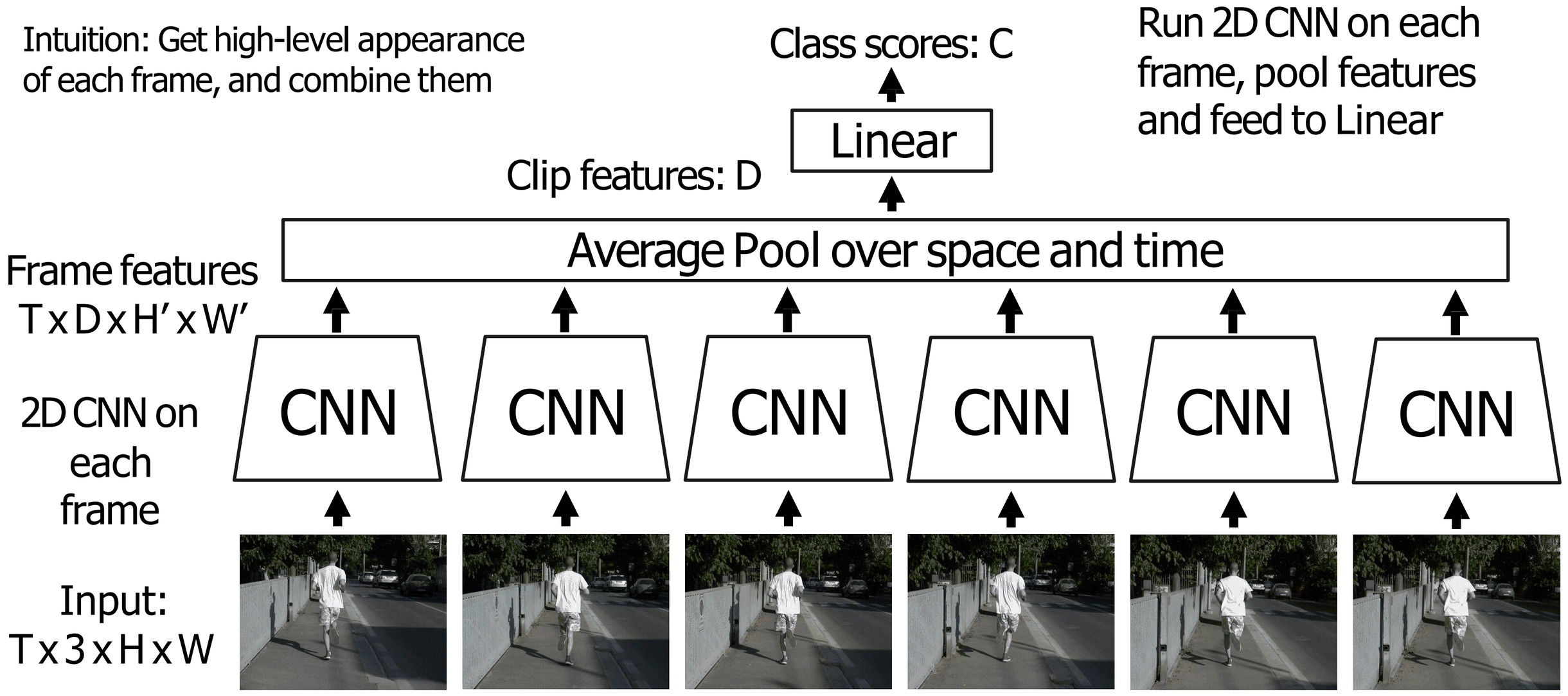


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Video Classification: Late Fusion (with pooling)

Intuition: Get high-level appearance of each frame, and combine them

Run 2D CNN on each frame, pool features and feed to Linear

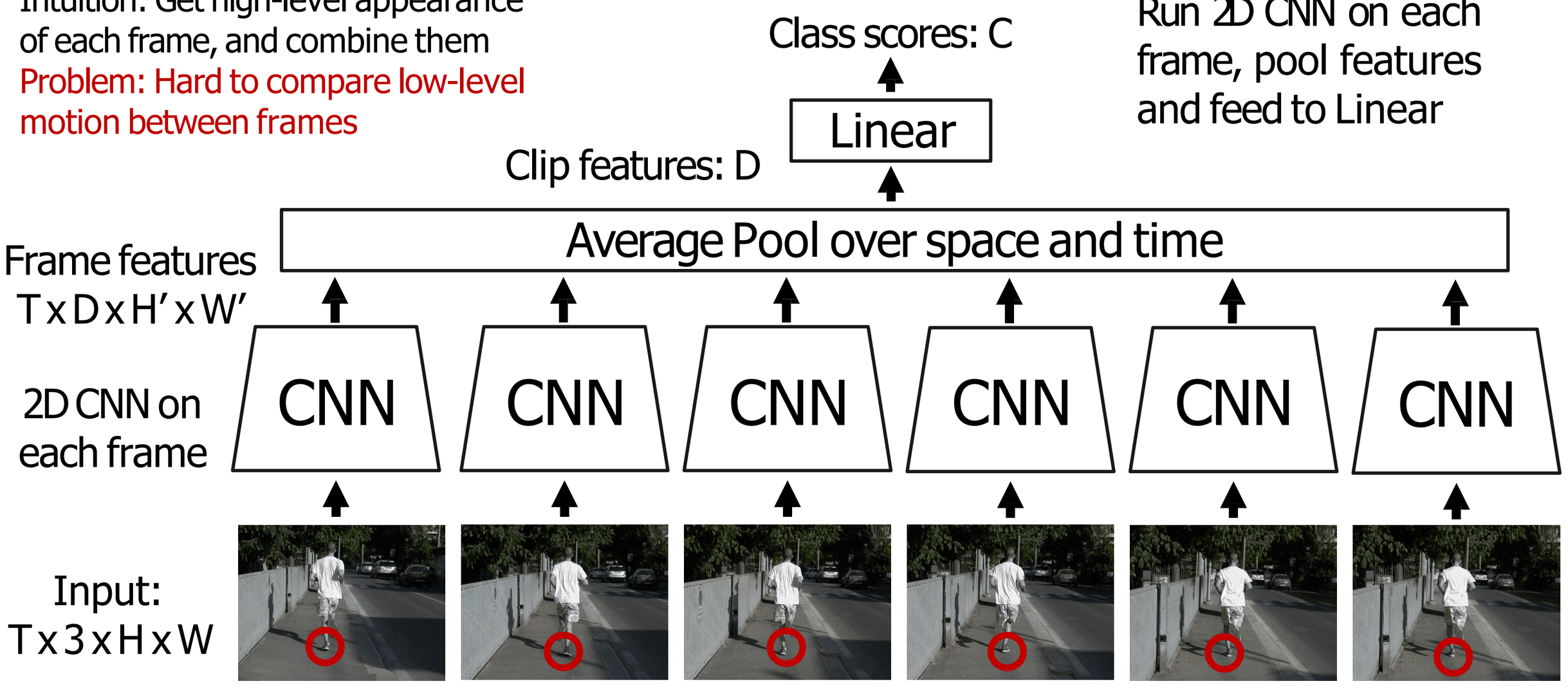


Video Classification: Late Fusion (with pooling)

Intuition: Get high-level appearance of each frame, and combine them

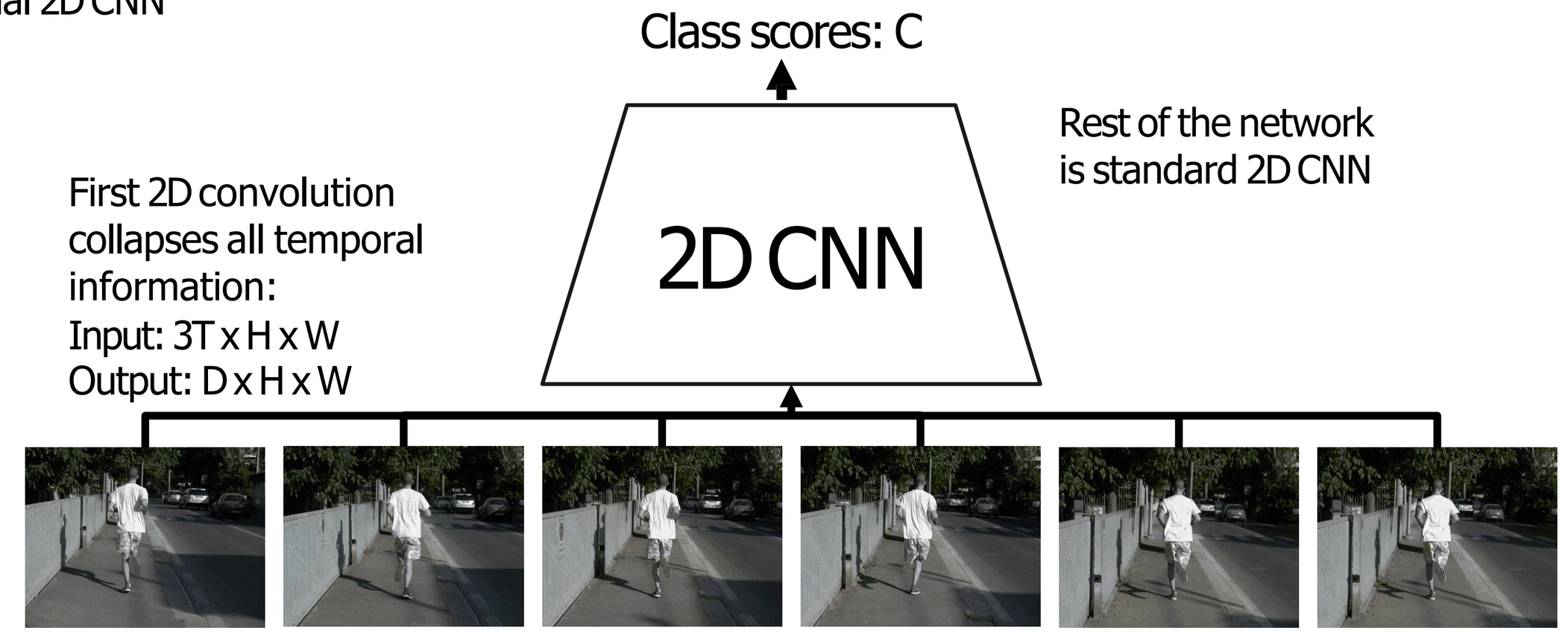
Problem: Hard to compare low-level motion between frames

Run 2D CNN on each frame, pool features and feed to Linear



Video Classification: Early Fusion

Intuition: Compare frames with very first conv layer, after that normal 2D CNN



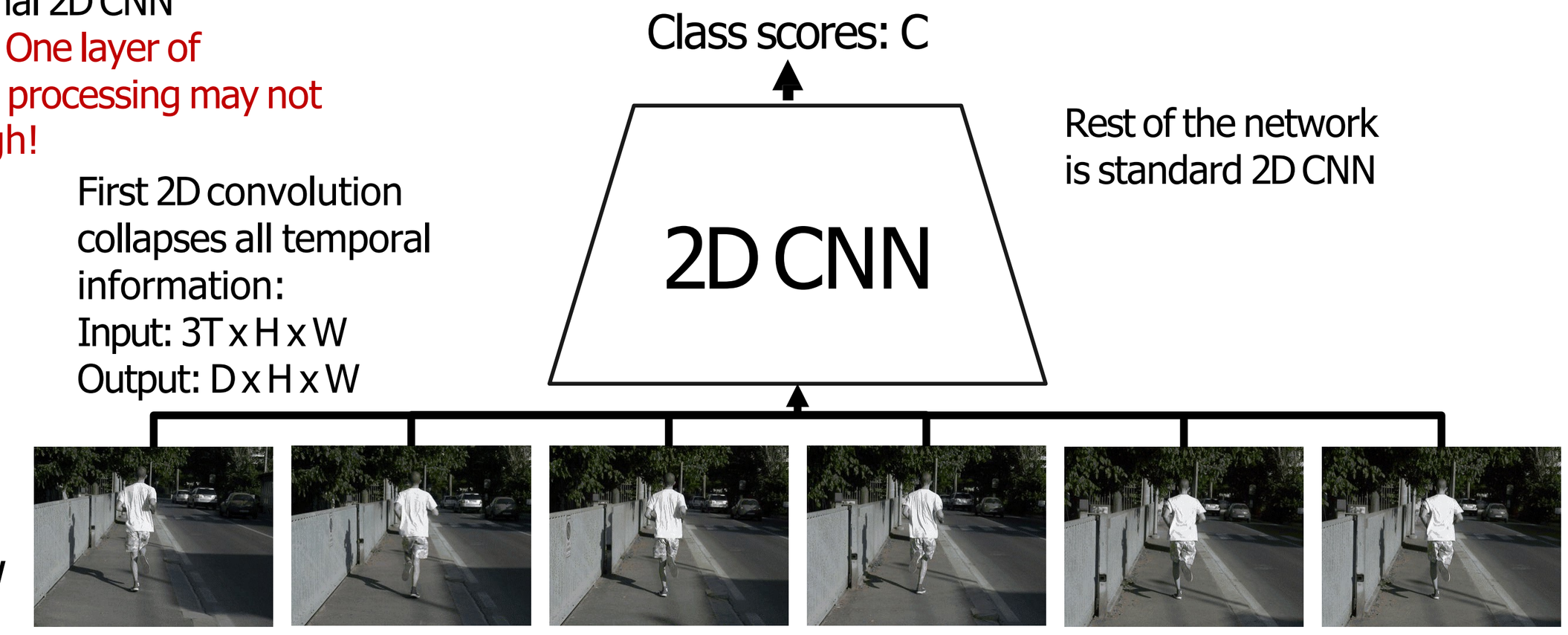
Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Video Classification: Early Fusion

Intuition: Compare frames with very first conv layer, after that normal 2D CNN

Problem: One layer of temporal processing may not be enough!

First 2D convolution collapses all temporal information:
Input: 3T x H x W
Output: D x H x W

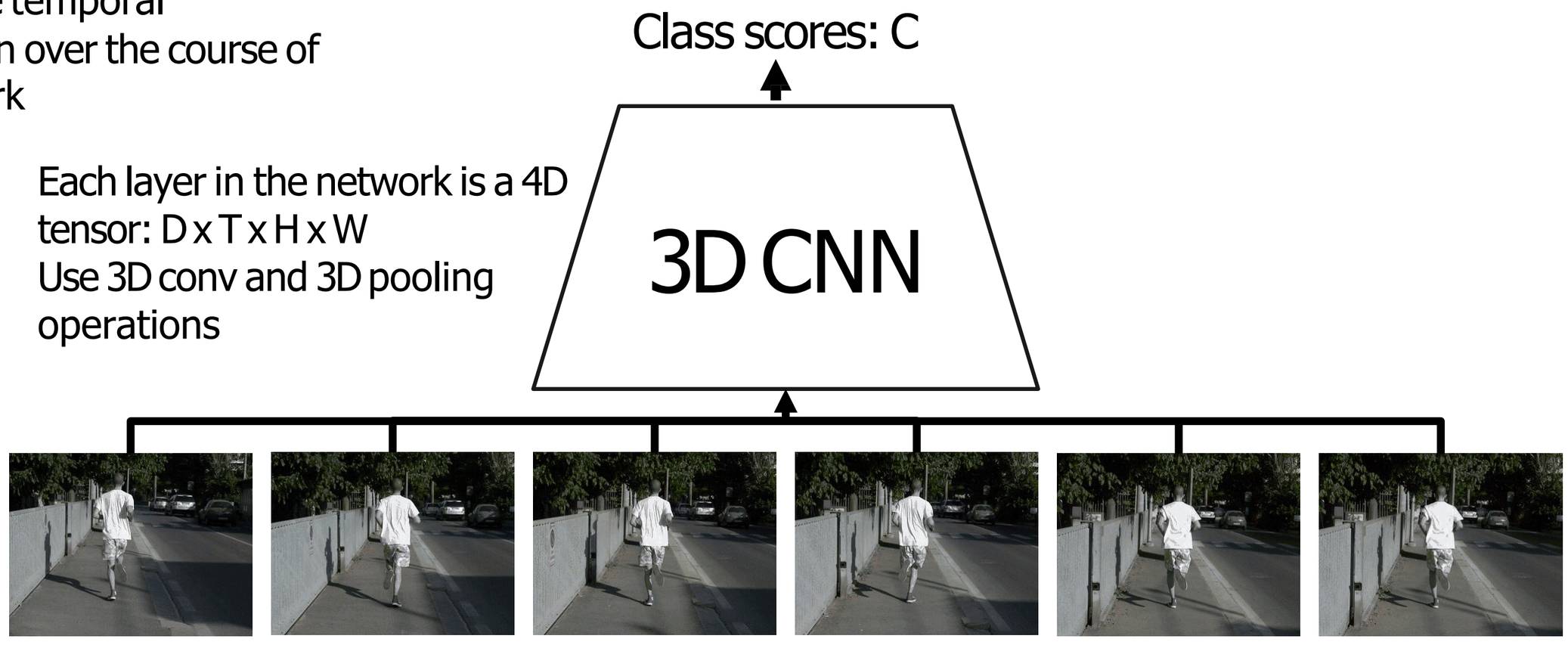


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Video Classification: 3D CNN

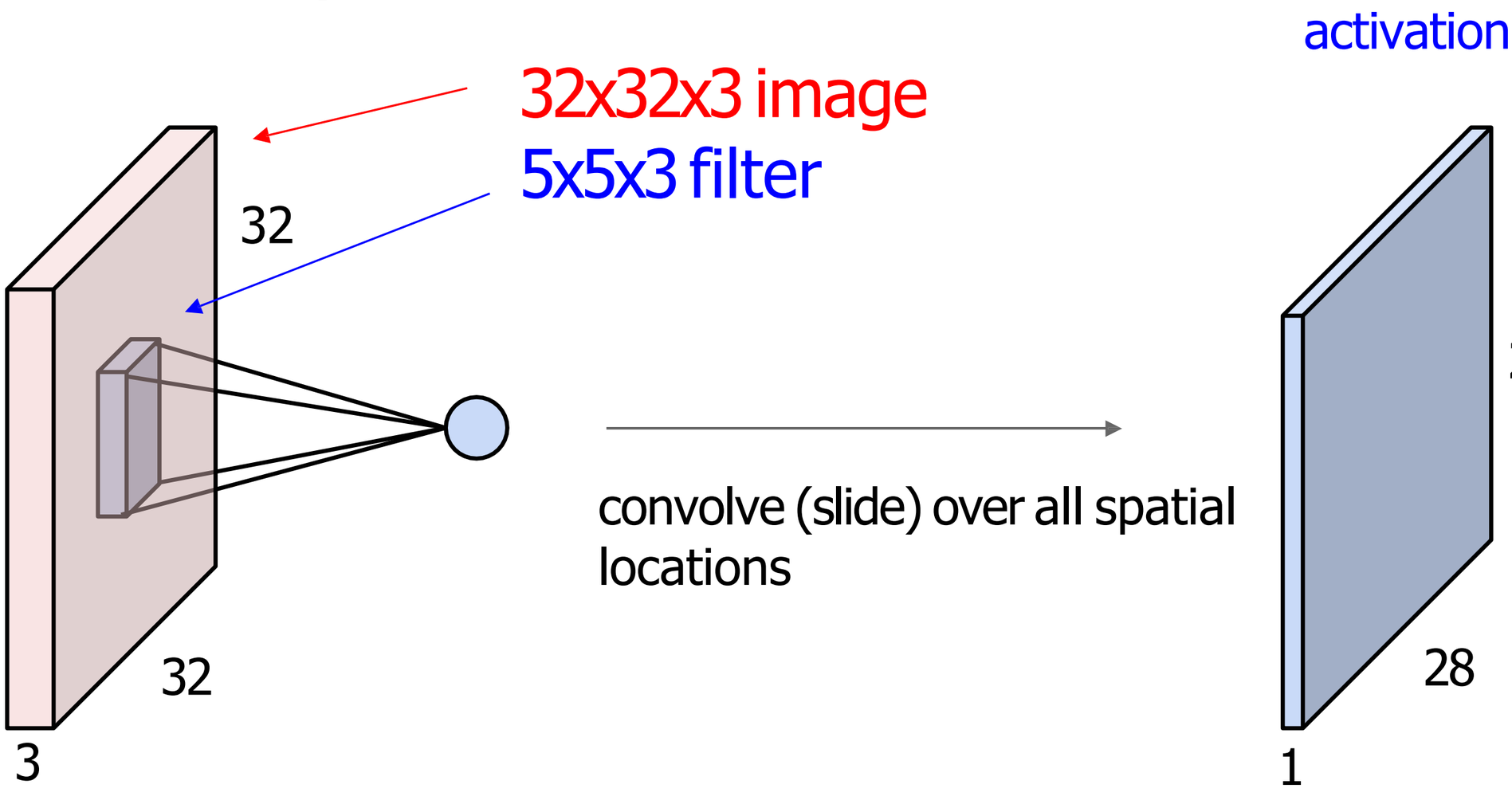
Intuition: Use 3D versions of convolution and pooling to slowly fuse temporal information over the course of the network

Each layer in the network is a 4D tensor: D x T x H x W
Use 3D conv and 3D pooling operations

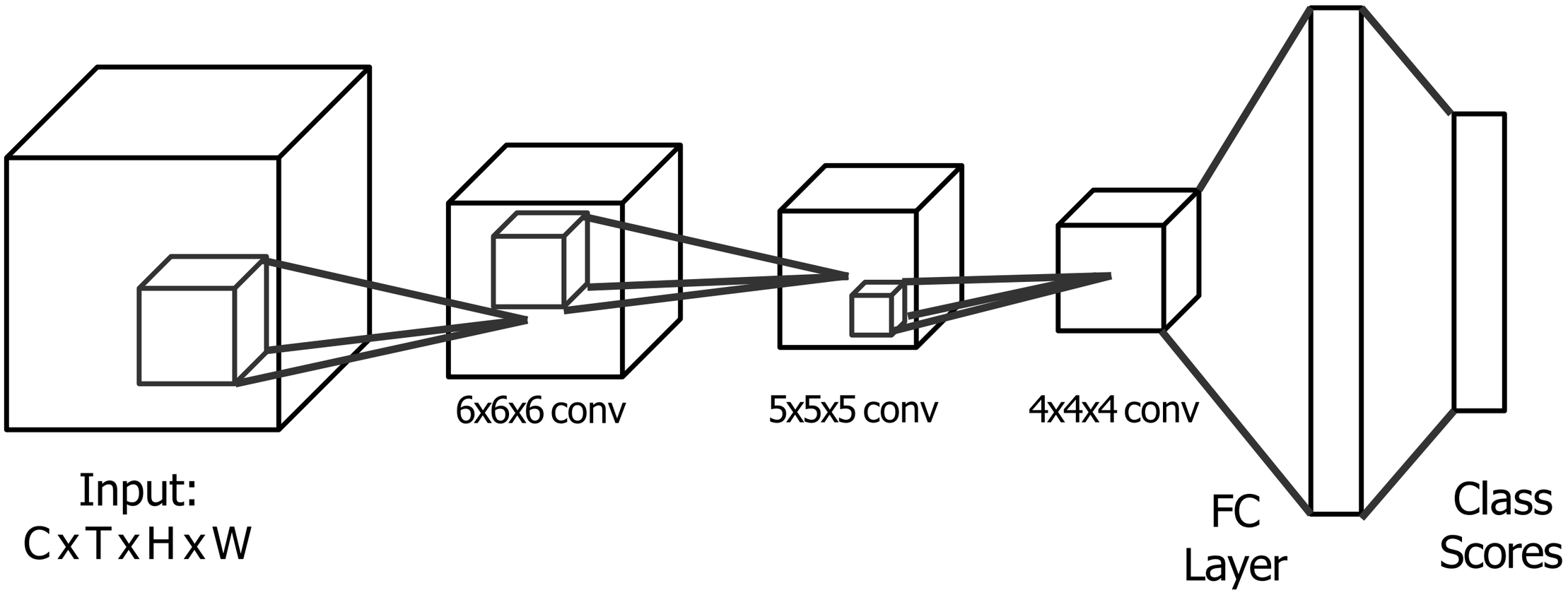


Ji et al, "3D Convolutional Neural Networks for Human Action Recognition", TPAMI 2010 ; Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Convolution Layer



3D Convolution



Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

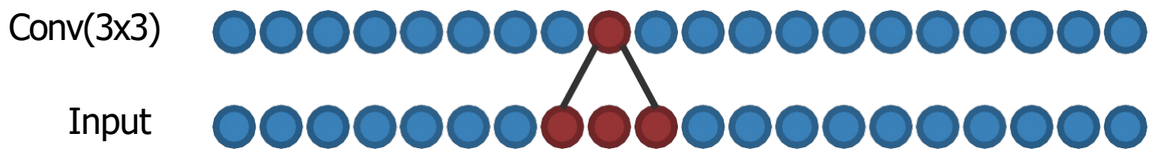
Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3

(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3

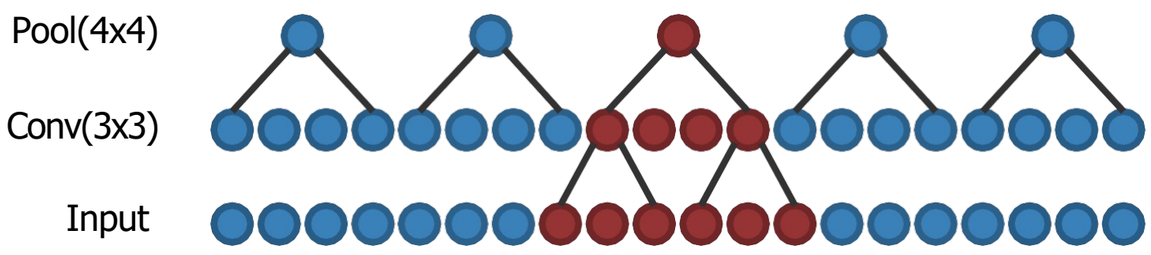


(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6



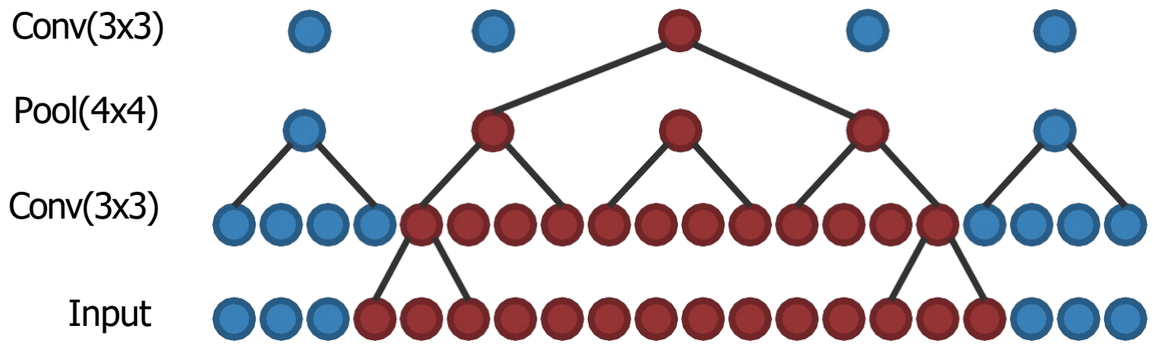
(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14

Build slowly in space



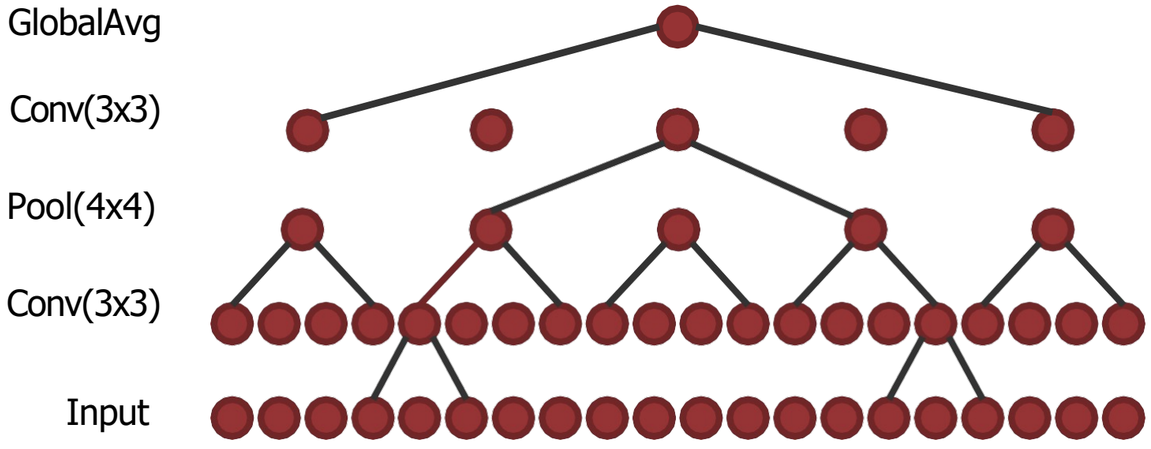
(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64

Build slowly in space,
All-at-once in time at end



(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64

Build slowly in space,
All-at-once in time at end

Early Fusion

Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3
Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6
Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14
GlobalAvgPool	24 x 1 x 1	20 x 64 x 64

Build slowly in space,
All-at-once in time at start

(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

	Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**	
Late Fusion	Input	3 x 20 x 64 x 64		
	Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3	Build slowly in space, All-at-once in time at end
	Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6	
	Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64	
Early Fusion	Input	3 x 20 x 64 x 64		
	Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3	Build slowly in space, All-at-once in time at start
	Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6	
	Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64	
3D CNN	Input	3 x 20 x 64 x 64		
	Conv3D(3x3x3, 3->12)	12 x 20 x 64 x 64	3 x 3 x 3	Build slowly in space, Build slowly in time ”Slow Fusion”
	Pool3D(4x4x4)	12 x 5 x 16 x 16	6 x 6 x 6	
	Conv3D(3x3x3, 12->24)	24 x 5 x 16 x 16	14 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64	

(Small example architectures, in practice much bigger)

Early Fusion vs Late Fusion vs 3D CNN

	Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Late Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
	Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
	GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64
Early Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3
	Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64
3D CNN	Input	3 x 20 x 64 x 64	
	Conv3D(3x3x3, 3->12)	12 x 20 x 64 x 64	3 x 3 x 3
	Pool3D(4x4x4)	12 x 5 x 16 x 16	6 x 6 x 6
	Conv3D(3x3x3, 12->24)	24 x 5 x 16 x 16	14 x 14 x 14
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64

What is the difference?

Build slowly in space,
All-at-once in time at end

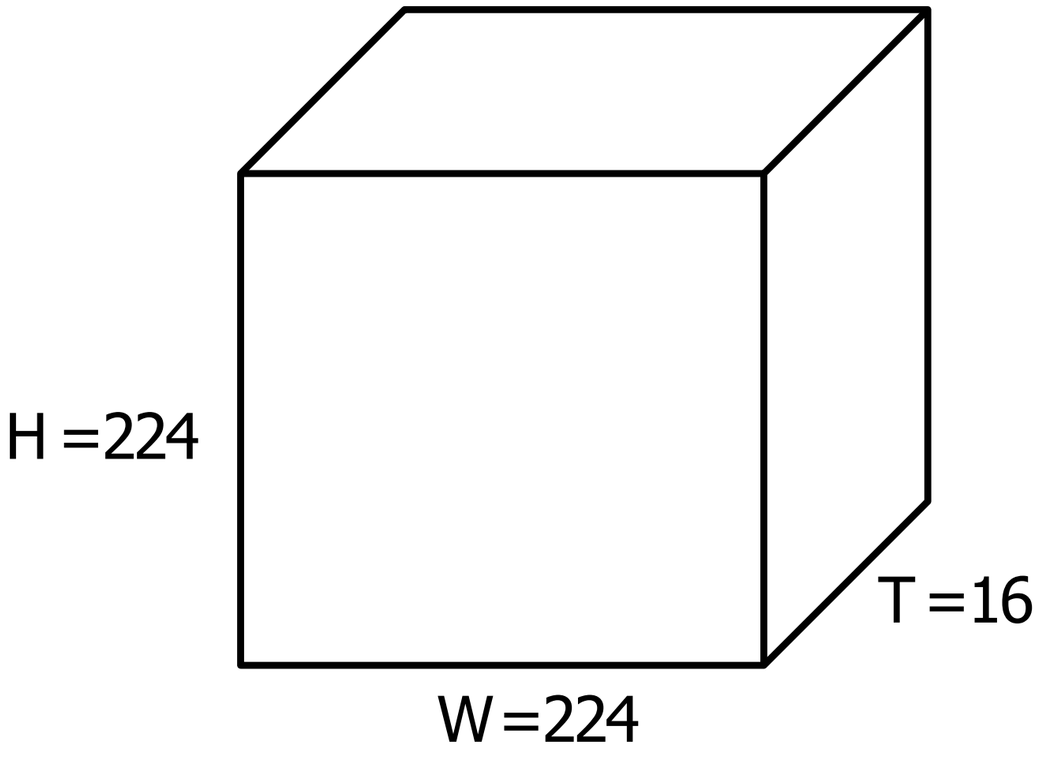
Build slowly in space,
All-at-once in time at start

Build slowly in space,
Build slowly in time
”Slow Fusion”

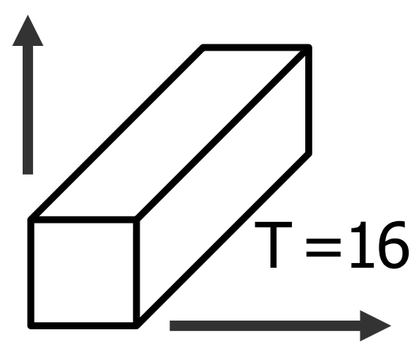
(Small example architectures, in practice much bigger)

2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

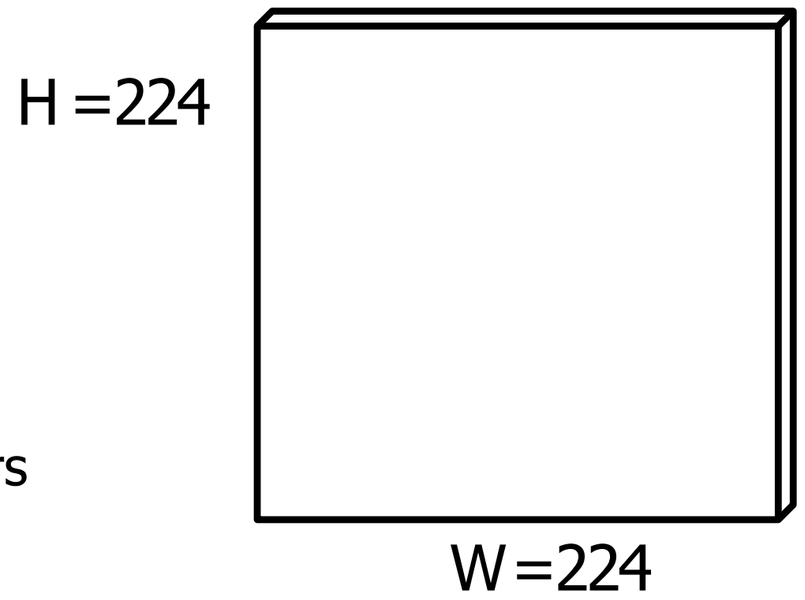


Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y



C_{out} different filters

Output:
C_{out} x H x W
2D grid with C_{out} – dim feat at each point



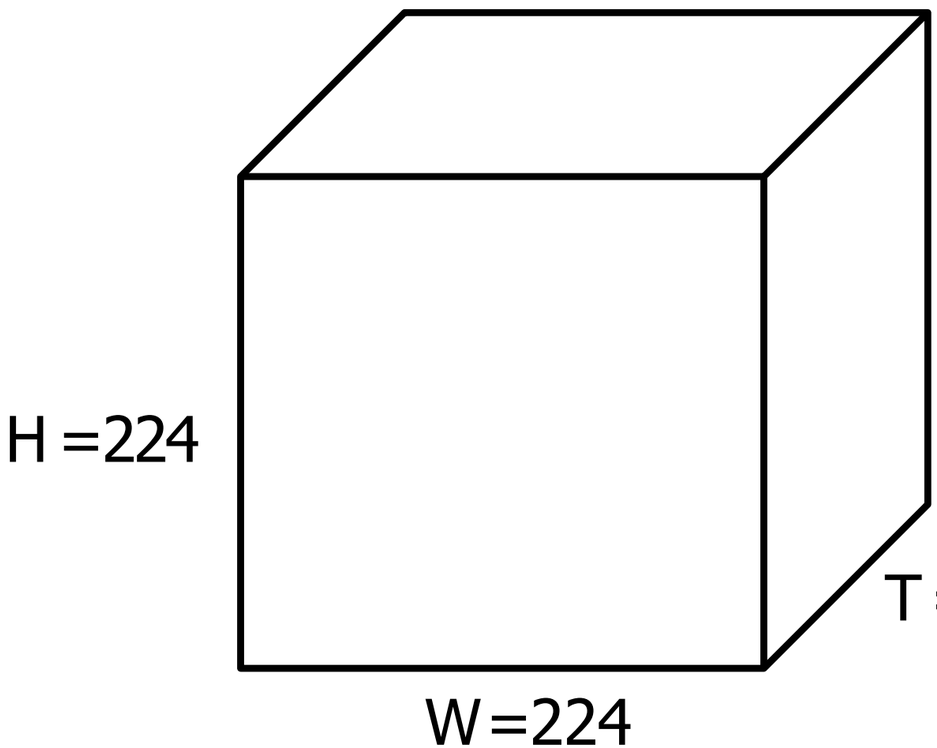
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

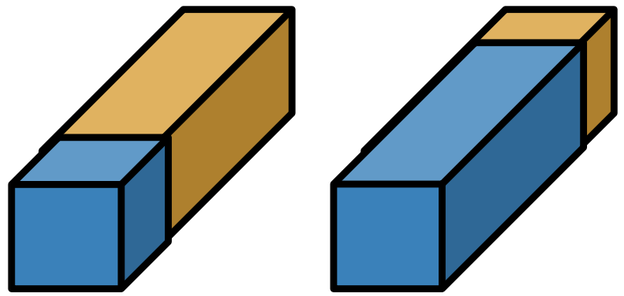
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y

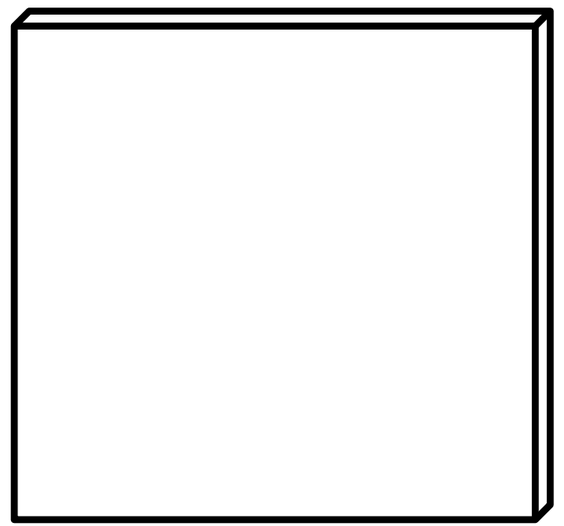
Output:
C_{out} x H x W
2D grid with C_{out} –dim feat at each point

No temporal shift-invariance!
Needs to learn separate filters for the same motion at different times in the clip





C_{out} different filters



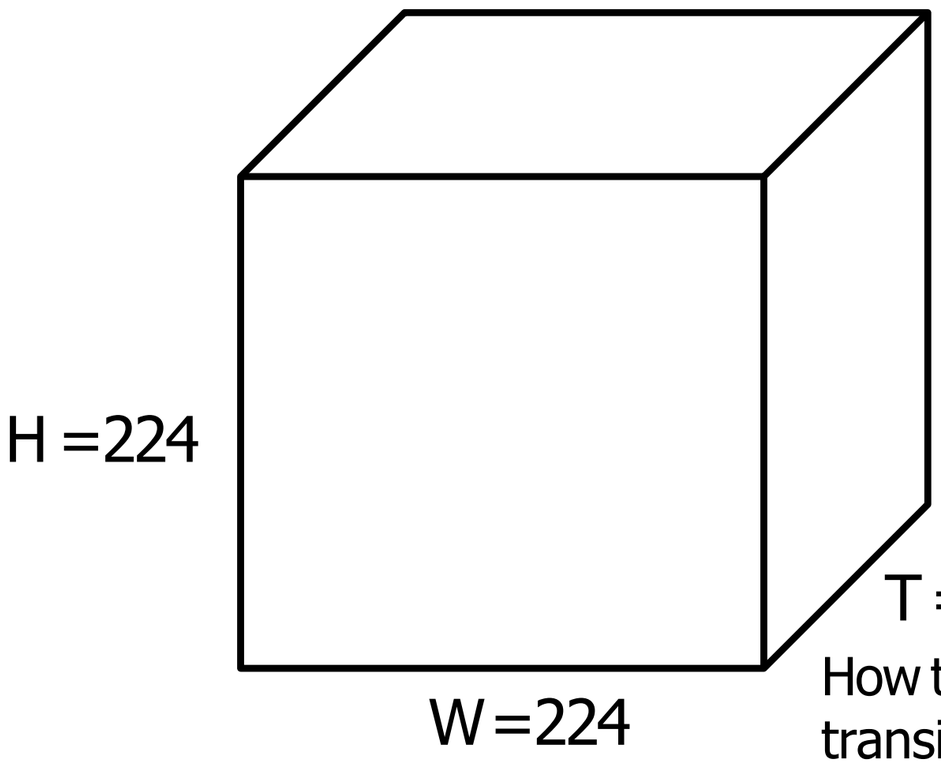
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

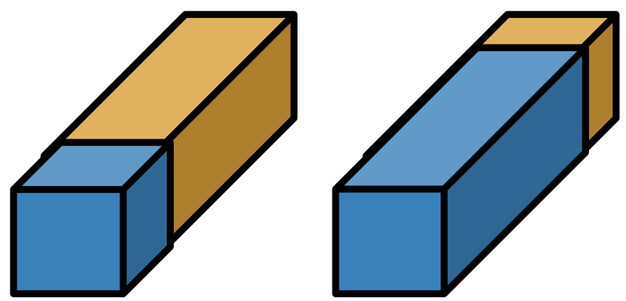
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y

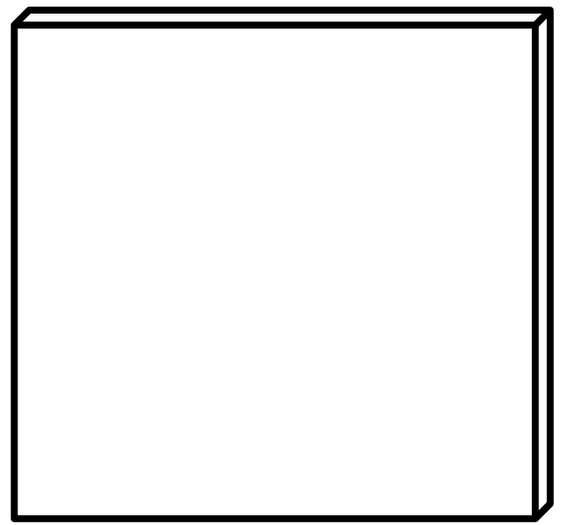
Output:
C_{out} x H x W
2D grid with C_{out} –dim feat at each point

No temporal shift-invariance!
Needs to learn separate filters for the same motion at different times in the clip





C_{out} different filters



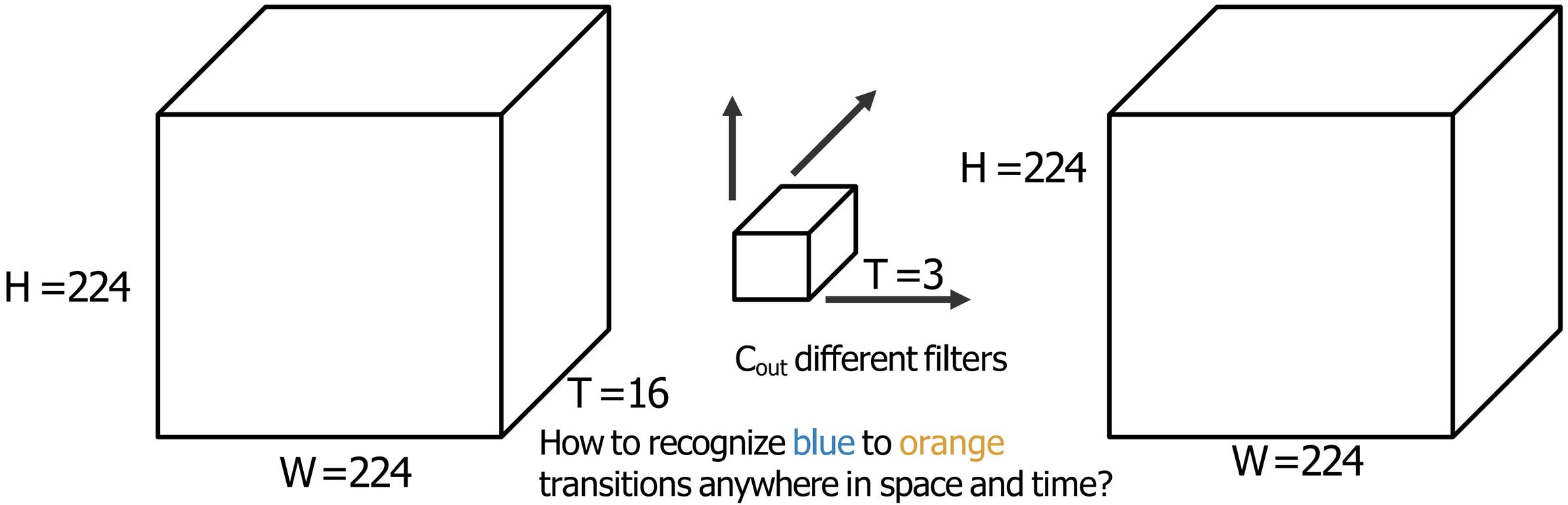
How to recognize blue to orange transitions anywhere in space and time?

2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

Output:
C_{out} x T x H x W
3D grid with C_{out}–dim feat at each point



C_{out} different filters

How to recognize blue to orange transitions anywhere in space and time?

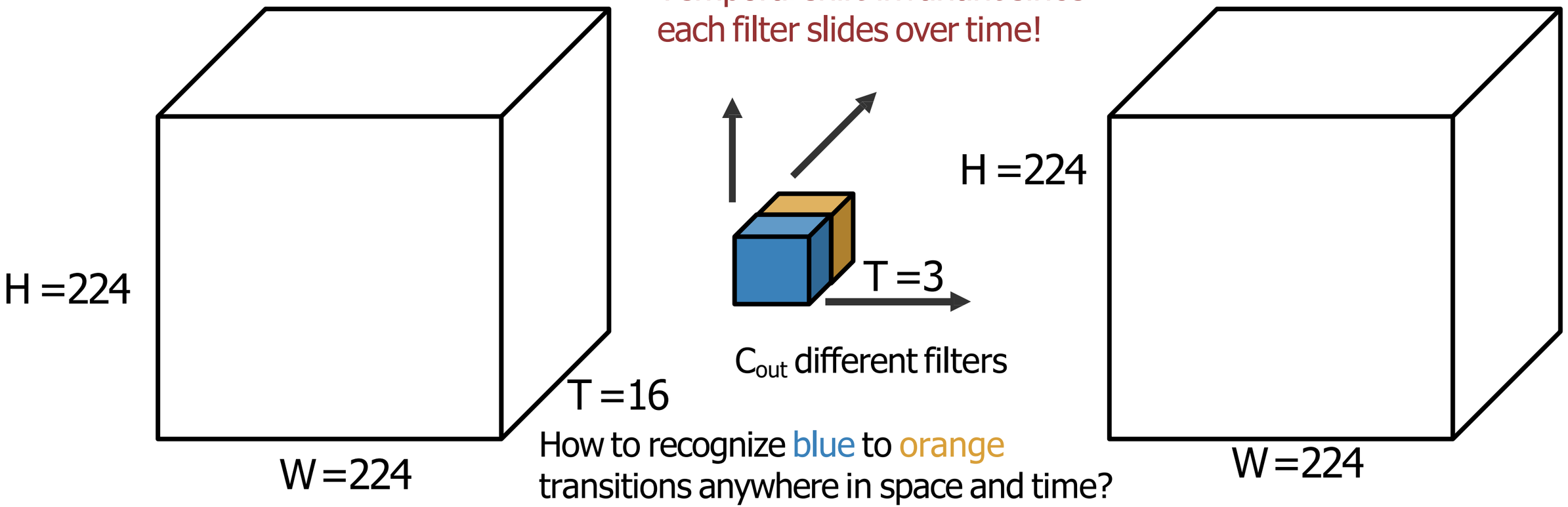
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

Output:
C_{out} x T x H x W
3D grid with C_{out}–dim feat at each point

Temporal shift-invariant since each filter slides over time!

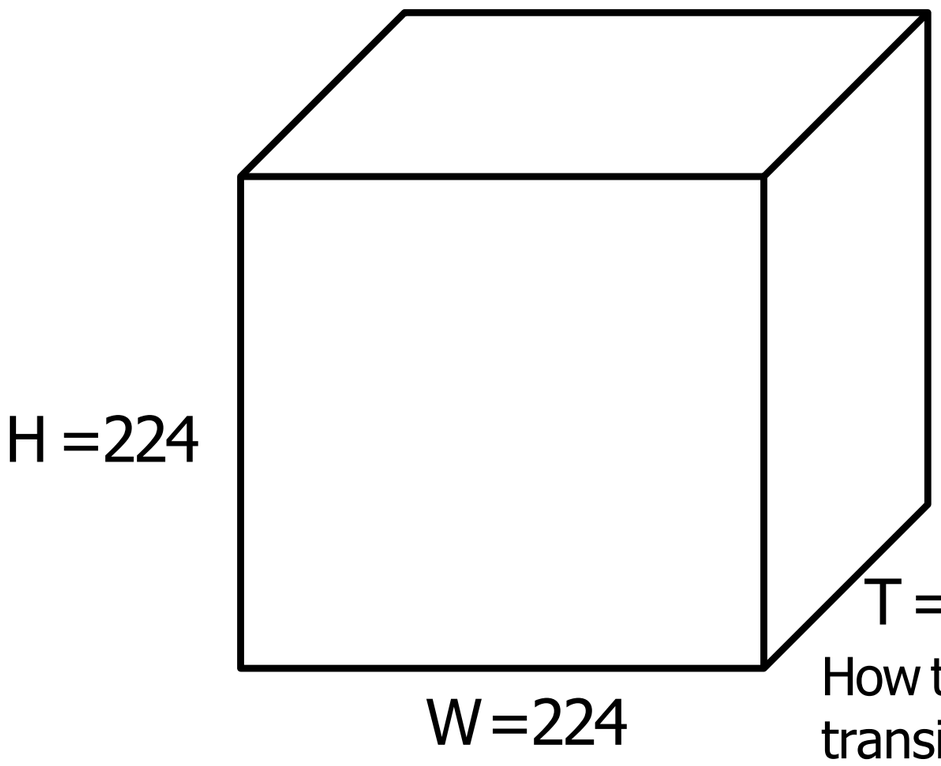


C_{out} different filters

How to recognize blue to orange transitions anywhere in space and time?

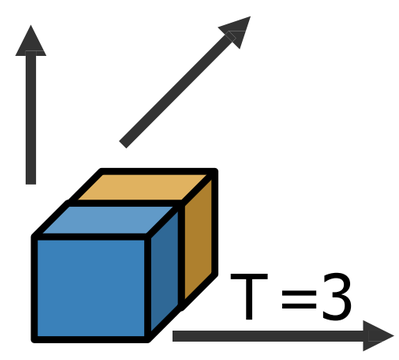
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)



Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

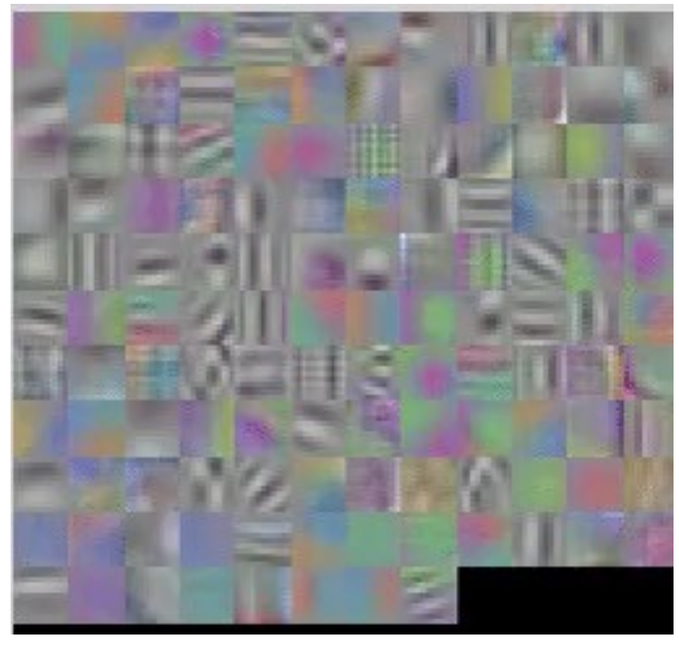
Temporal shift-invariant since each filter slides over time!



C_{out} different filters

How to recognize blue to orange transitions anywhere in space and time?

First-layer filters have shape 3 (RGB) x 4 (frames) x 5 x 5 (space)
Can visualize as video clips!



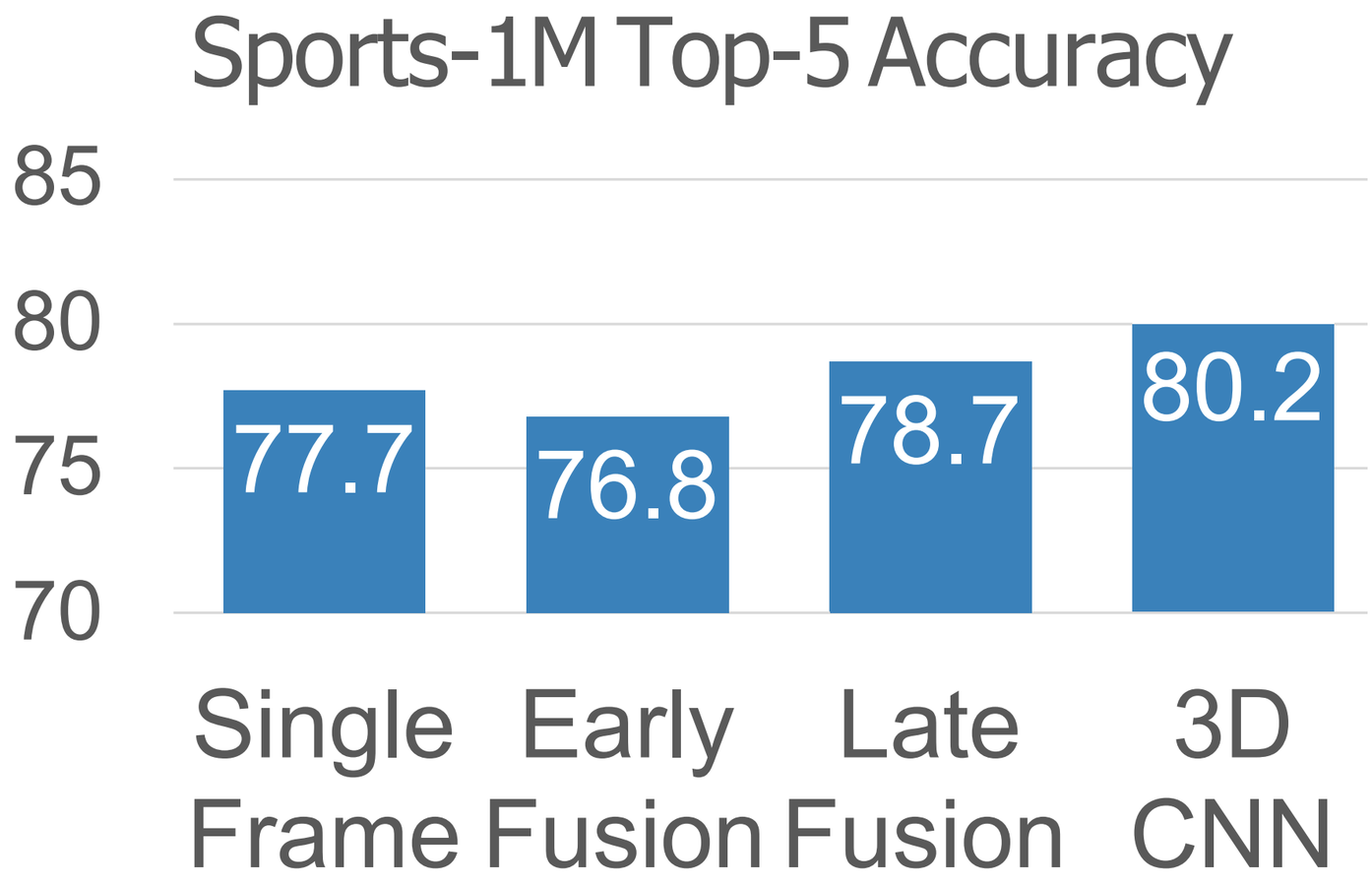
Example Video Dataset: Sports-1M

1 million YouTube videos annotated with labels for 487 different types of sports

Ground Truth
Correct prediction
Incorrect prediction

Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Early Fusion vs Late Fusion vs 3D CNN



Single Frame model works well – always try this first!

3D CNNs have improved a lot since 2014!

Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

C3D: The VGG of 3D CNNs

3D CNN that uses all 3x3x3 conv and 2x2x2 pooling
(except Pool1 which is 1x2x2)

Released model pretrained on Sports-1M: Many people used this as a video feature extractor

Layer	Size
Input	3 x 16 x 112 x 112
Conv1 (3x3x3)	64 x 16 x 112 x 112
Pool1 (1x2x2)	64 x 16 x 56 x 56
Conv2 (3x3x3)	128 x 16 x 56 x 56
Pool2 (2x2x2)	128 x 8 x 28 x 28
Conv3a (3x3x3)	256 x 8 x 28 x 28
Conv3b (3x3x3)	256 x 8 x 28 x 28
Pool3 (2x2x2)	256 x 4 x 14 x 14
Conv4a (3x3x3)	512 x 4 x 14 x 14
Conv4b (3x3x3)	512 x 4 x 14 x 14
Pool4 (2x2x2)	512 x 2 x 7 x 7
Conv5a (3x3x3)	512 x 2 x 7 x 7
Conv5b (3x3x3)	512 x 2 x 7 x 7
Pool5	512 x 1 x 3 x 3
FC6	4096
FC7	4096
FC8	C

Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

C3D: The VGG of 3D CNNs

3D CNN that uses all 3x3x3 conv and 2x2x2 pooling
(except Pool1 which is 1x2x2)

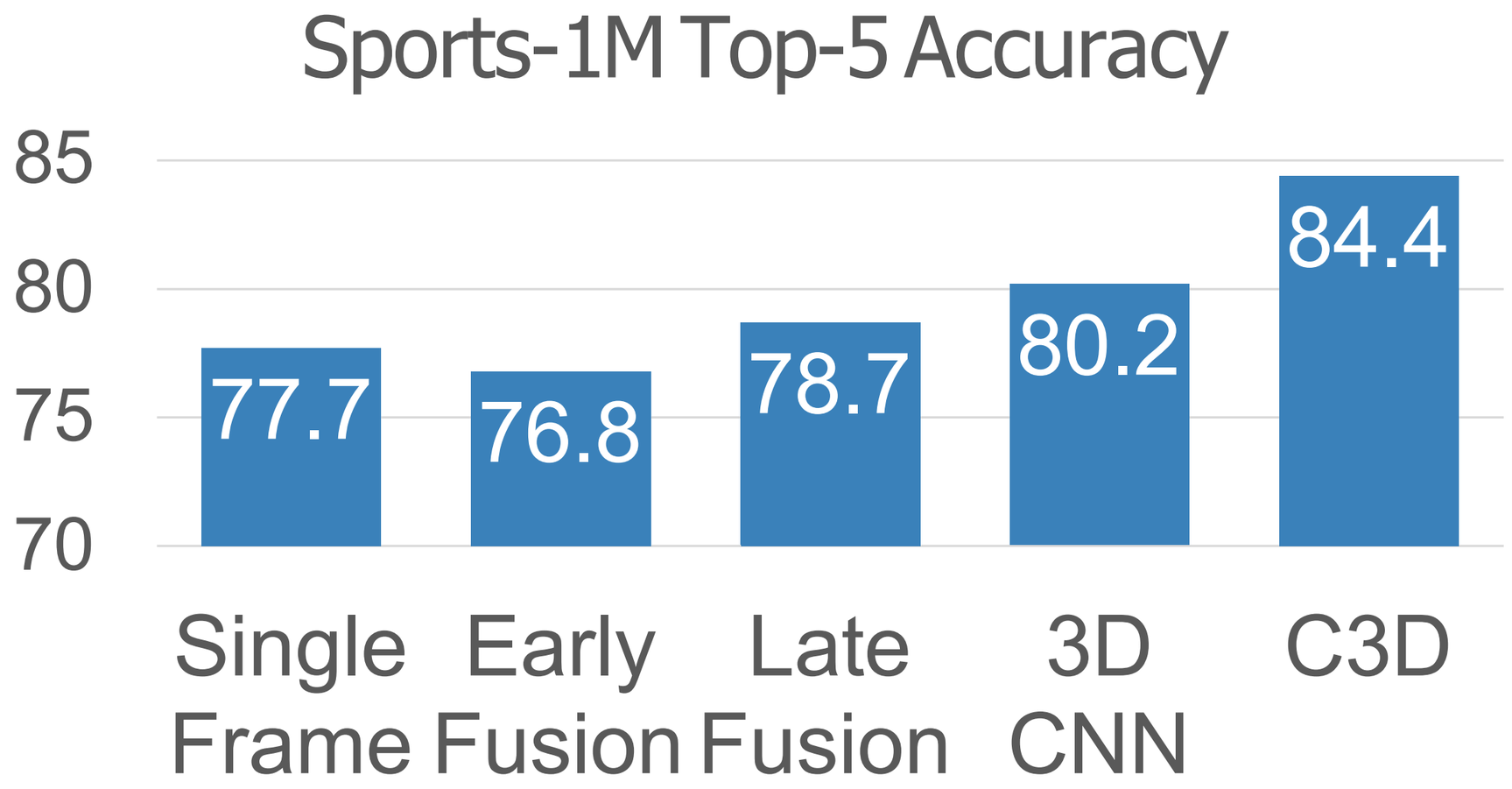
Released model pretrained on Sports-1M: Many people used this as a video feature extractor

Problem: 3x3x3 conv is very expensive!
AlexNet: 0.7 GFLOP
VGG-16: 13.6 GFLOP
C3D: 39.5 GFLOP (2.9x VGG!)

Layer	Size	MFLOPs
Input	3 x 16 x 112 x 112	
Conv1 (3x3x3)	64 x 16 x 112 x 112	1.04
Pool1 (1x2x2)	64 x 16 x 56 x 56	
Conv2 (3x3x3)	128 x 16 x 56 x 56	11.10
Pool2 (2x2x2)	128 x 8 x 28 x 28	
Conv3a (3x3x3)	256 x 8 x 28 x 28	5.55
Conv3b (3x3x3)	256 x 8 x 28 x 28	11.10
Pool3 (2x2x2)	256 x 4 x 14 x 14	
Conv4a (3x3x3)	512 x 4 x 14 x 14	2.77
Conv4b (3x3x3)	512 x 4 x 14 x 14	5.55
Pool4 (2x2x2)	512 x 2 x 7 x 7	
Conv5a (3x3x3)	512 x 2 x 7 x 7	0.69
Conv5b (3x3x3)	512 x 2 x 7 x 7	0.69
Pool5	512 x 1 x 3 x 3	
FC6	4096	0.51
FC7	4096	0.45
FC8	C	0.05

Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

Early Fusion vs Late Fusion vs 3D CNN



Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014
Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

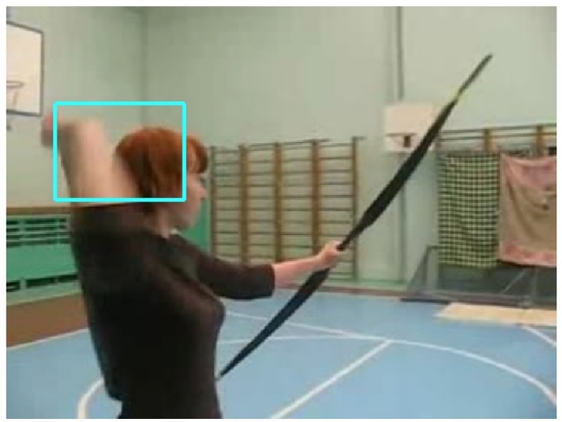
Recognizing Actions from Motion

We can easily recognize actions using only motion information

Johansson, "Visual perception of biological motion and a model for its analysis." Perception & Psychophysics. 14(2):201-211. 1973.

Measuring Motion: Optical Flow

Image at frame t



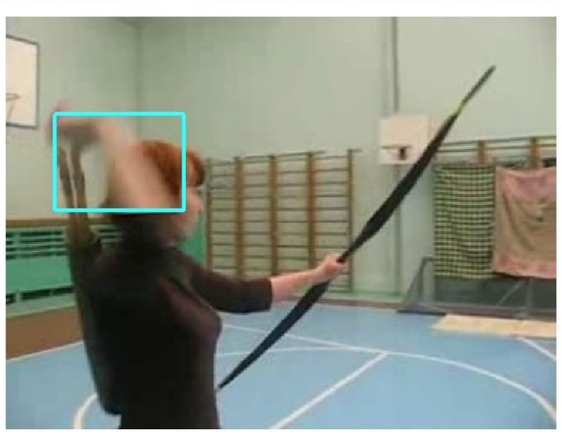
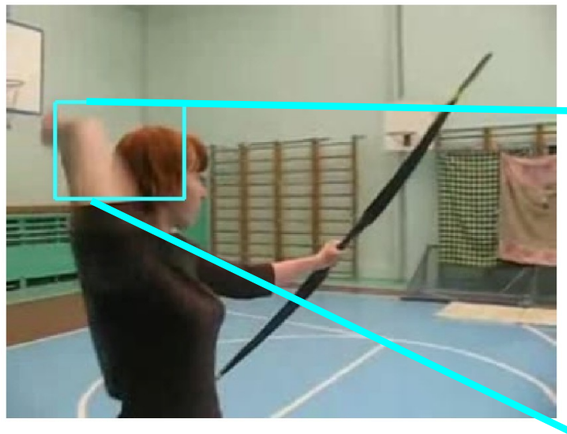


Image at frame t+1

Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014

Measuring Motion: Optical Flow

Image at frame t



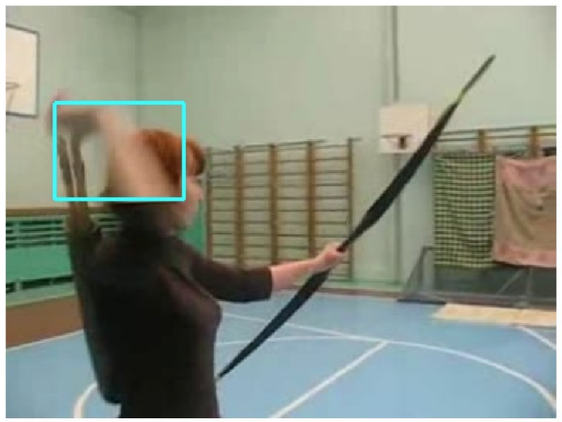
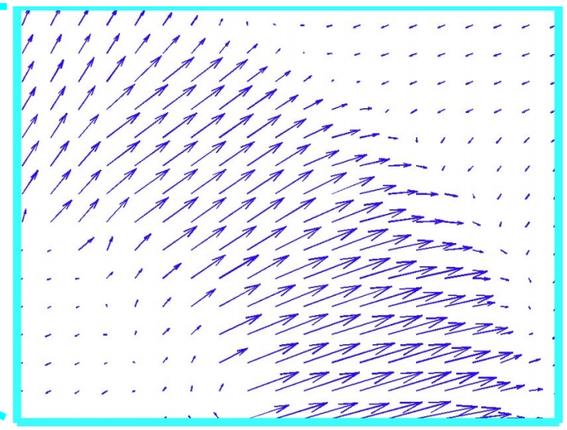


Image at frame t+1

Optical flow gives a displacement field F between images I_t and I_{t+1}



Tells where each pixel will move in the next frame:

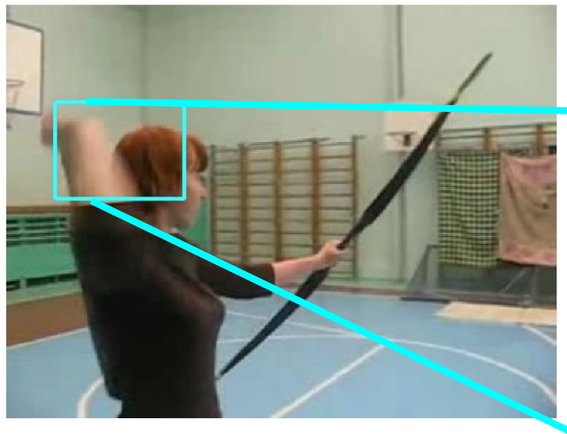
$F(x, y) = (dx, dy)$

$I_{t+1}(x+dx, y+dy) = I_t(x, y)$

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Measuring Motion: Optical Flow

Image at frame t



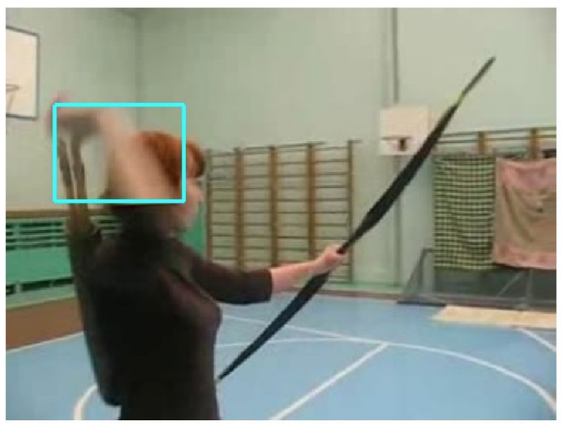
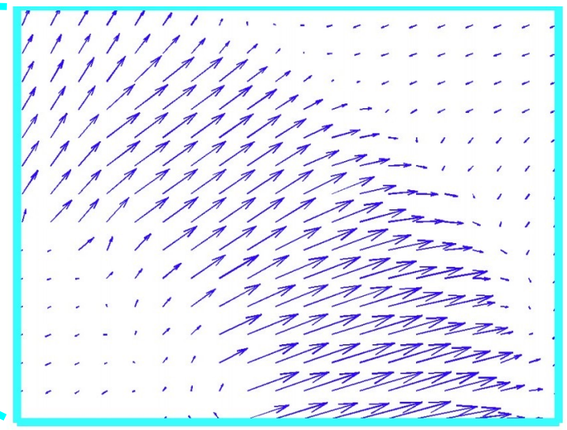


Image at frame t+1

Optical flow gives a displacement field F between images I_t and I_{t+1}



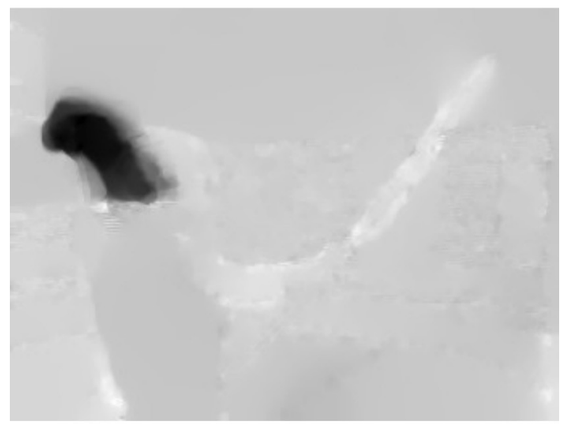
Tells where each pixel will move in the next frame:

$F(x, y) = (dx, dy)$

$I_{t+1}(x+dx, y+dy) = I_t(x, y)$

Optical Flow highlights local motion

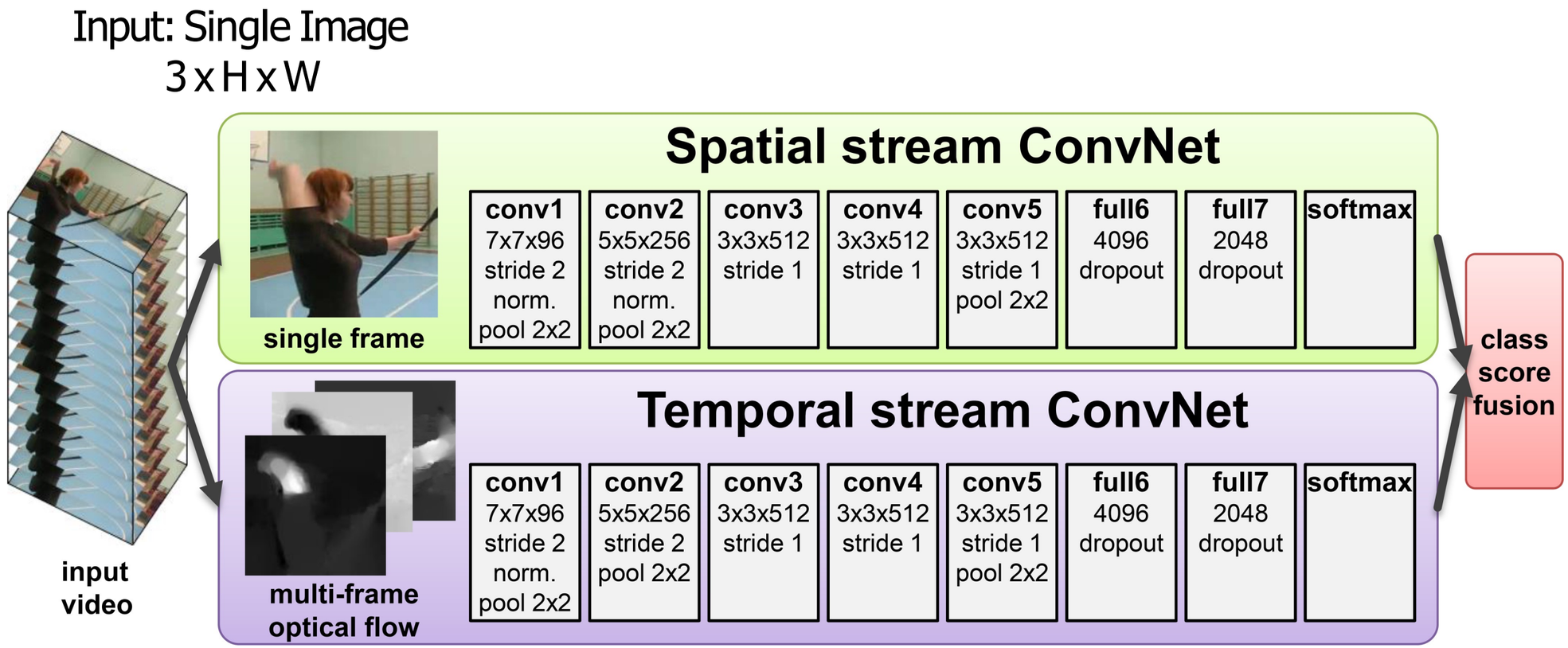
Horizontal flow dx



Vertical Flow dy

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Separating Motion and Appearance: Two-Stream Networks

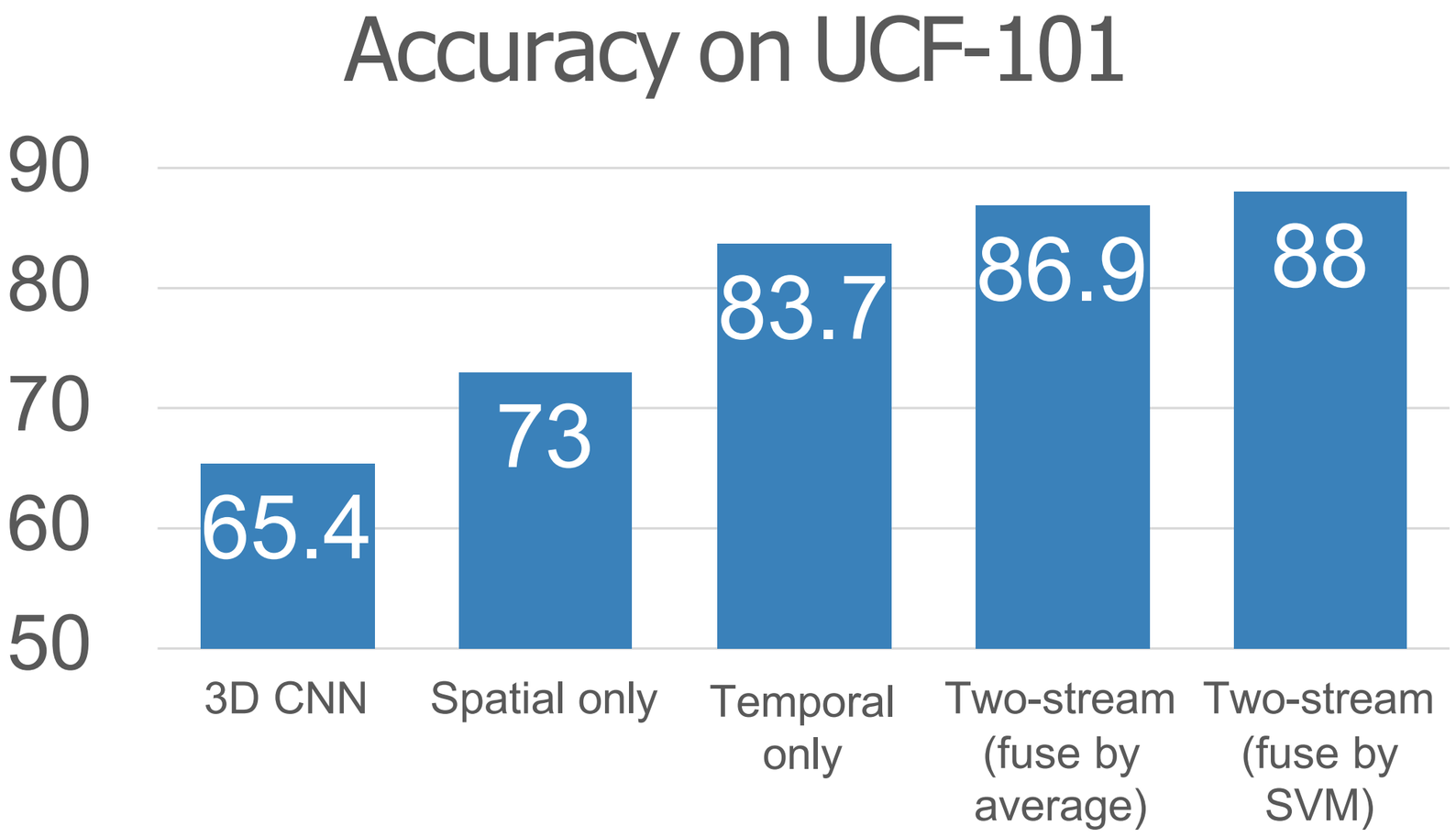


Input: Stack of optical flow:
[2*(T-1)] x H x W

Early fusion: First 2D conv
processes all flow images

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

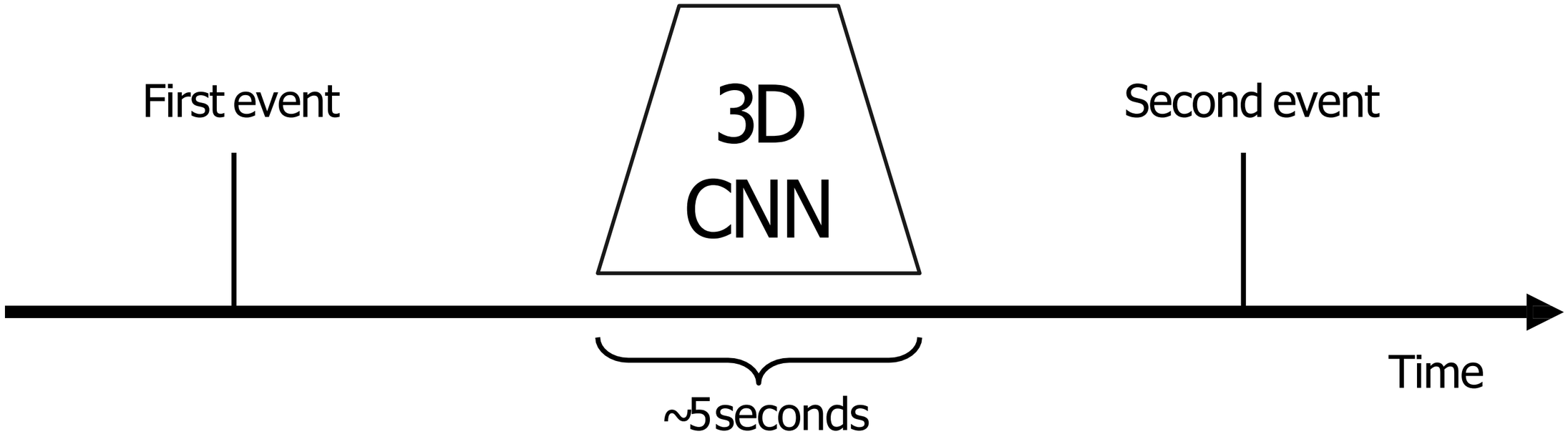
Separating Motion and Appearance: Two-Stream Networks



Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Modeling long-term temporal structure

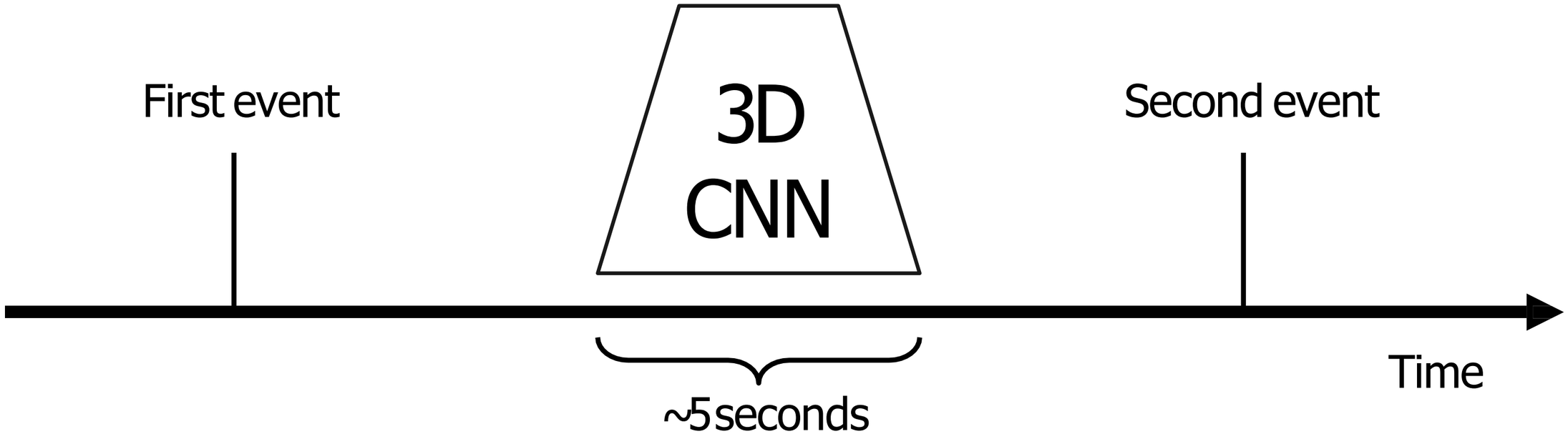
So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?



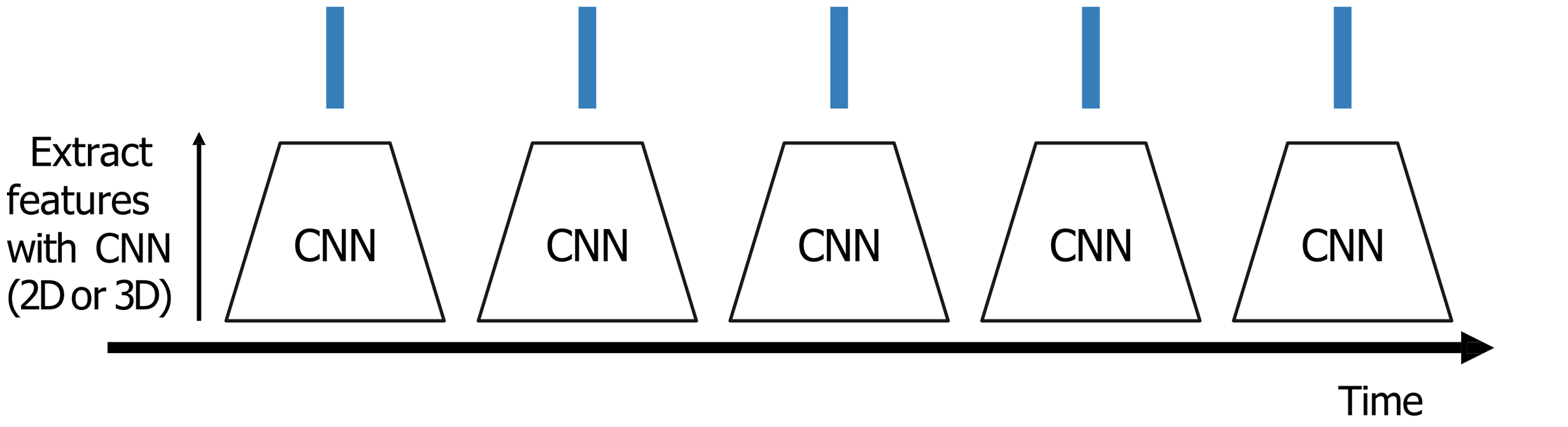
Modeling long-term temporal structure

So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?

We know how to handle sequences! How about recurrent networks?

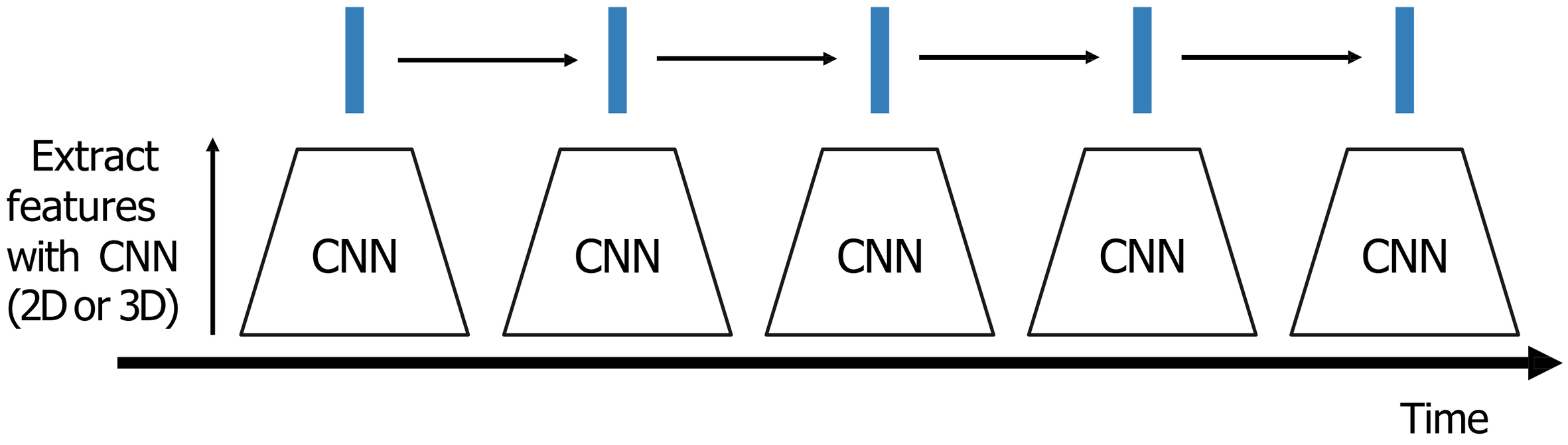


Modeling long-term temporal structure



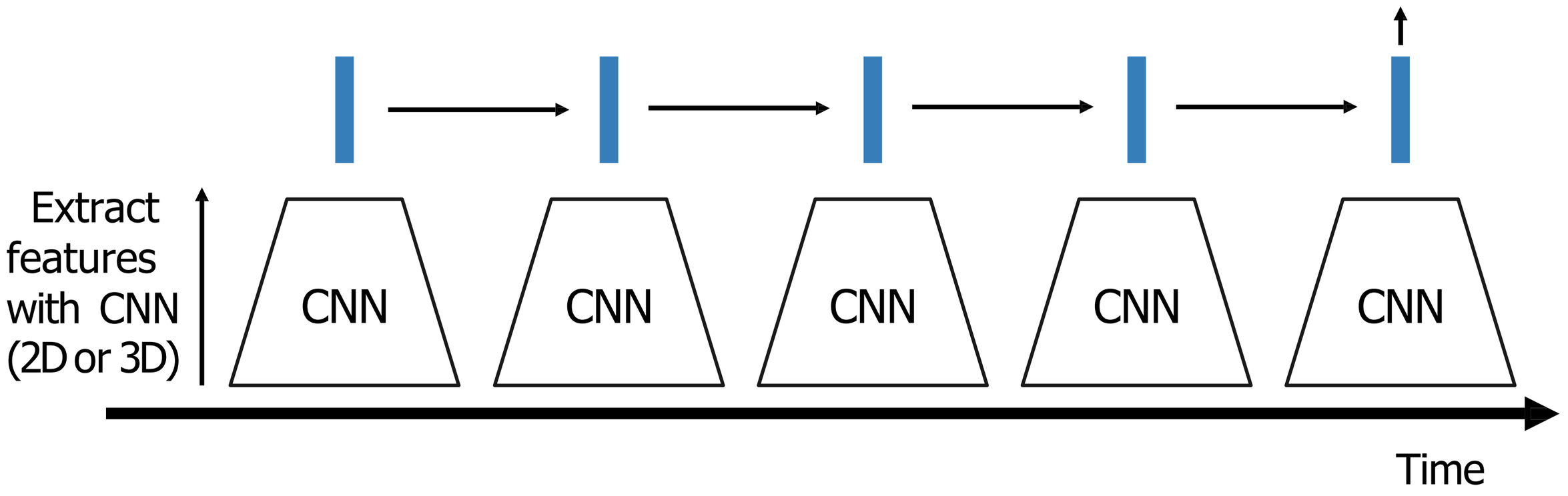
Modeling long-term temporal structure

Process local features using recurrent network (e.g. LSTM)



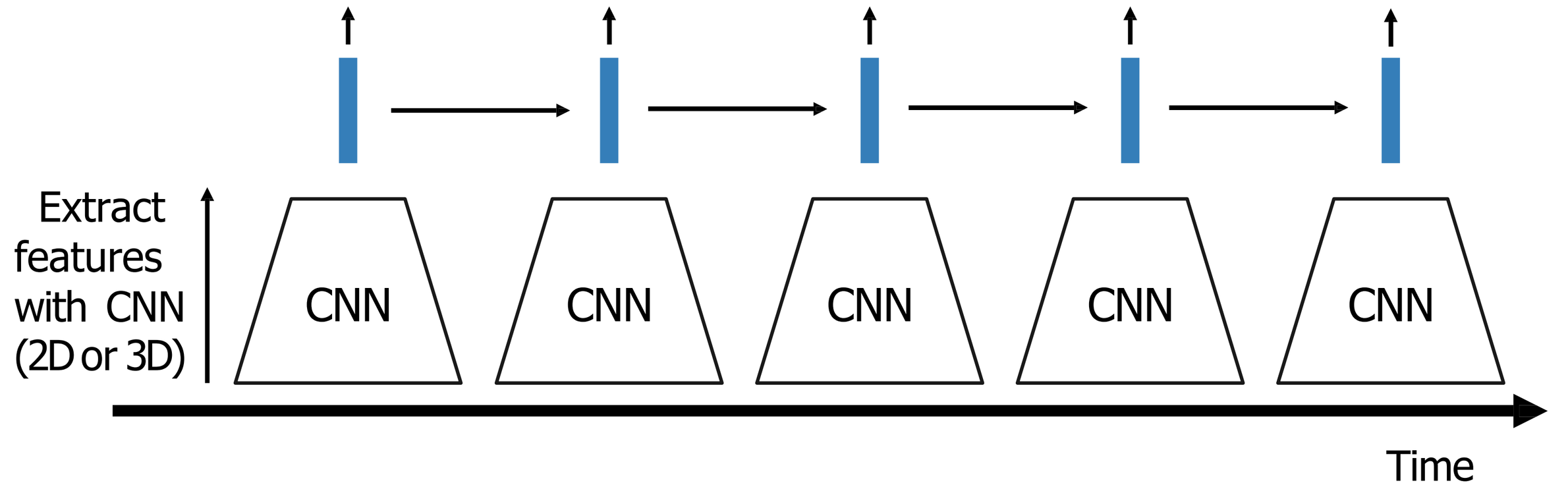
Modeling long-term temporal structure

Process local features using recurrent network (e.g. LSTM)
Many to one: One output at end of video



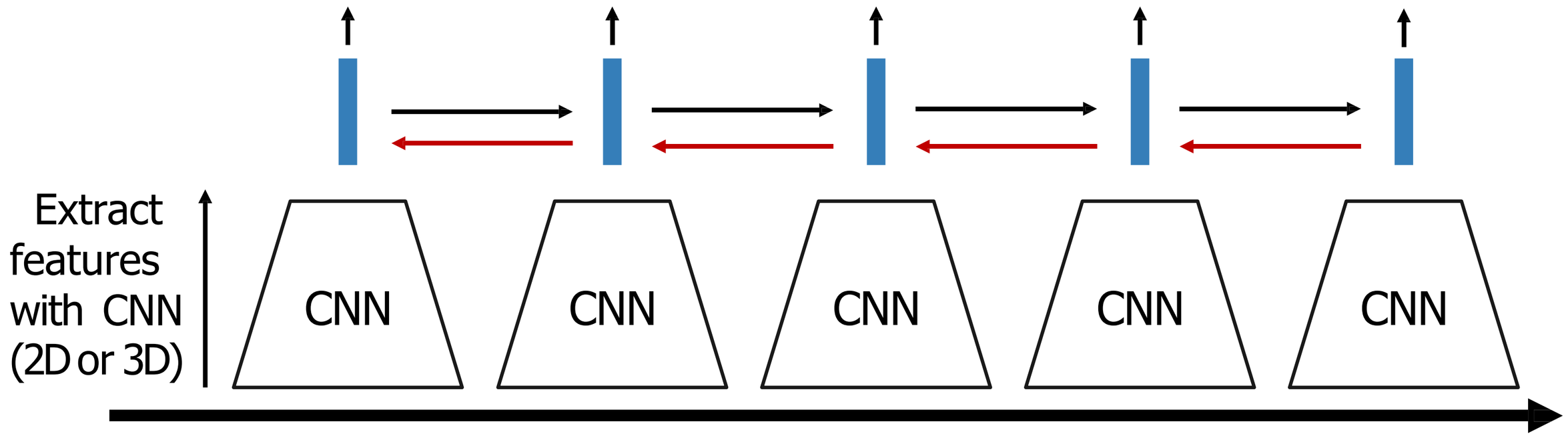
Modeling long-term temporal structure

Process local features using recurrent network (e.g. LSTM)
Many to many: one output per video frame



Modeling long-term temporal structure

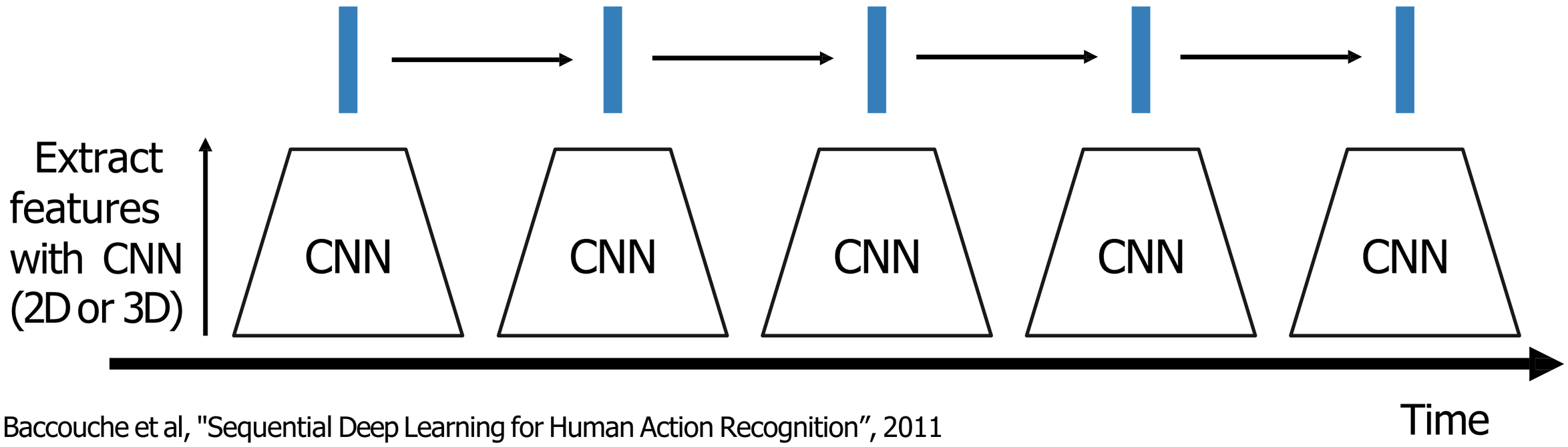
Sometimes don't backprop to CNN to save memory;
pretrain and use it as a feature extractor



Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Modeling long-term temporal structure

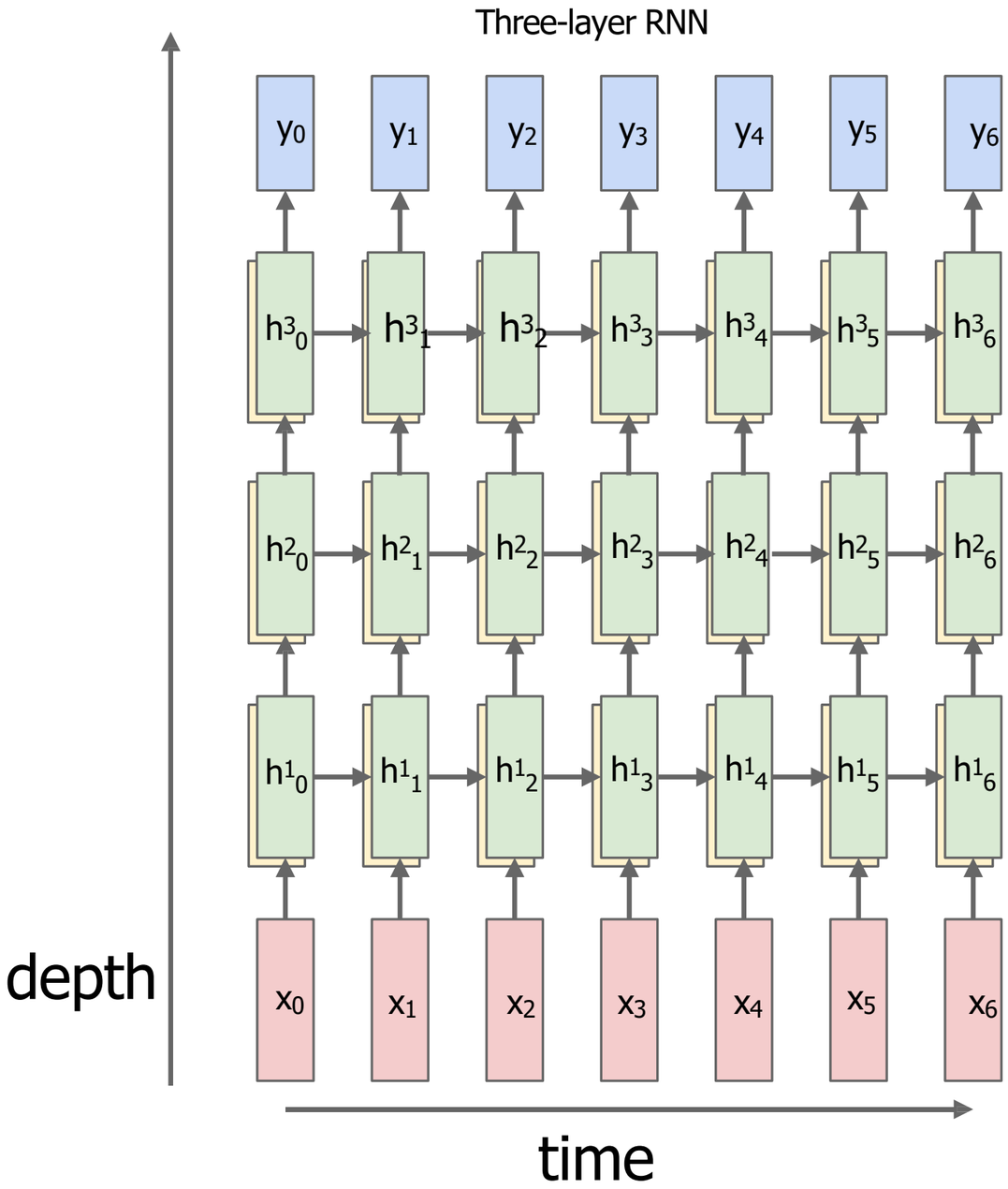
Inside CNN: Each value is a function of a fixed temporal window (local temporal structure)
Inside RNN: Each vector is a function of all previous vectors (global temporal structure)
Can we merge both approaches?



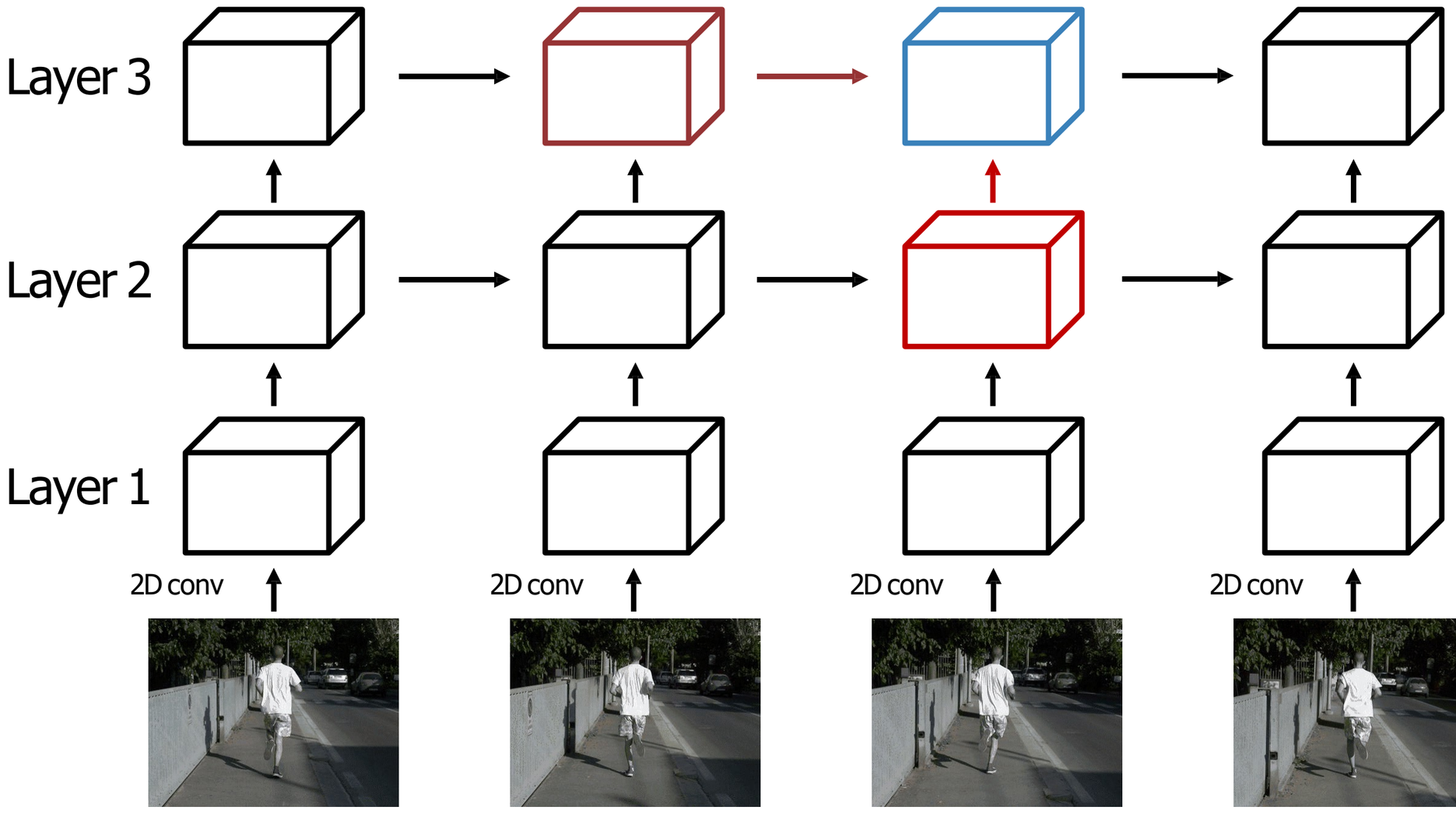
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Recall: Multi-layer RNN

We can use a similar structure to process videos!



Recurrent Convolutional Network



Entire network uses 2D feature maps: C x H x W

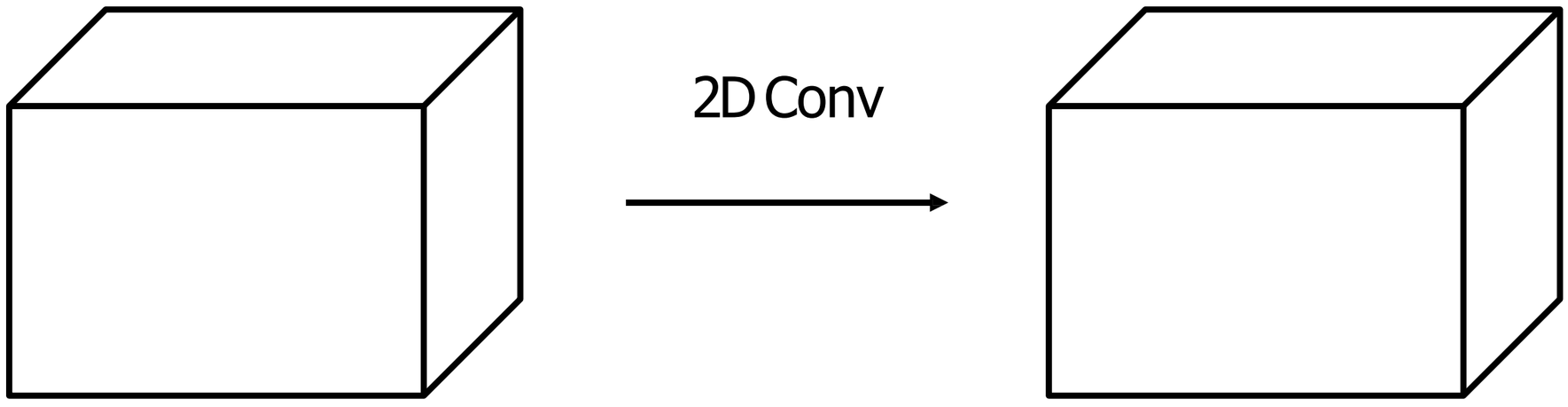
Each depends on two inputs:
1. Same layer, previous timestep
2. Prev layer, same timestep

Use different weights at each layer, share weights across time

Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Recurrent Convolutional Network

Normal 2D CNN:



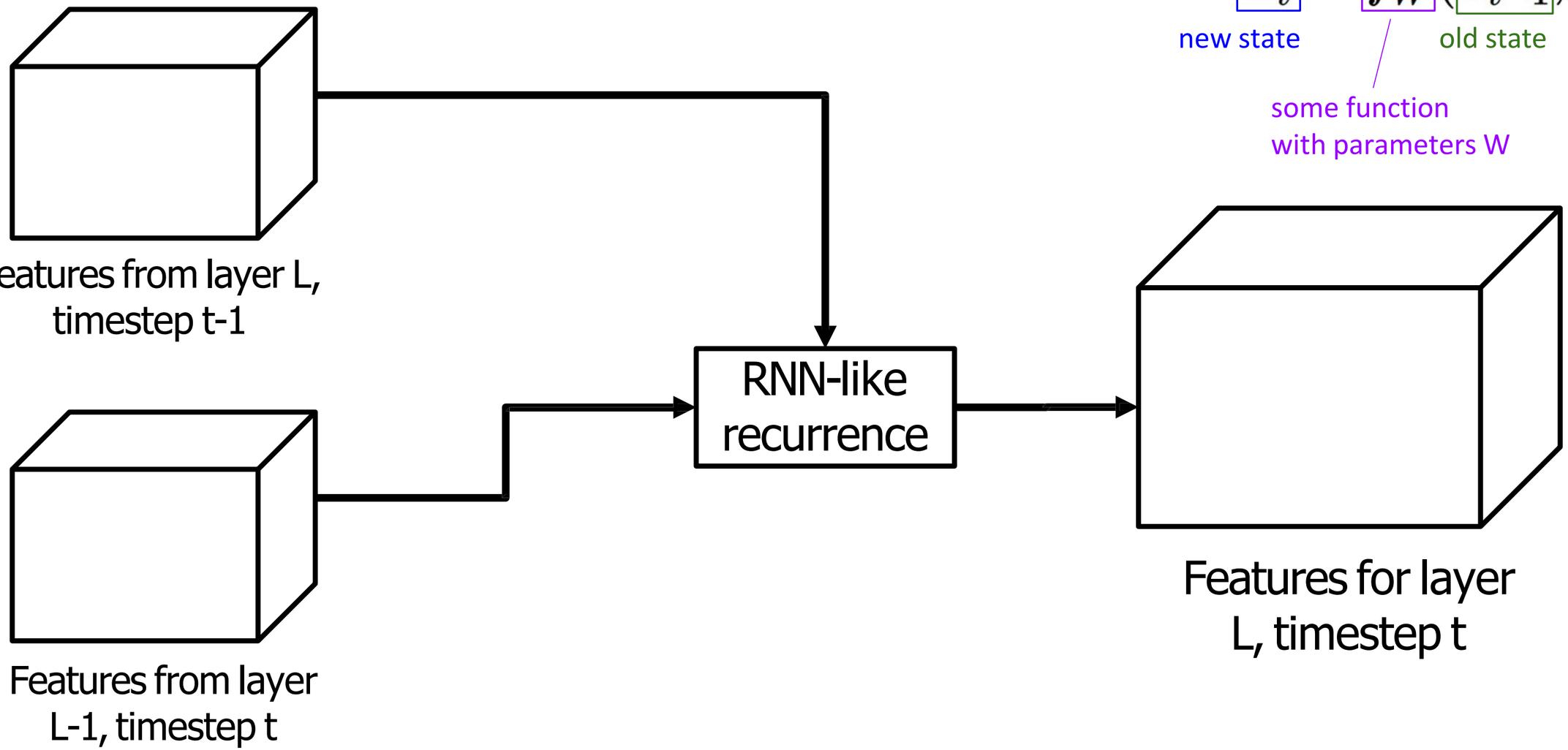
Recurrent Convolutional Network

Recall: Recurrent Network

$$h_t = f_W(h_{t-1}, x_t)$$

new state — old state

some function with parameters W



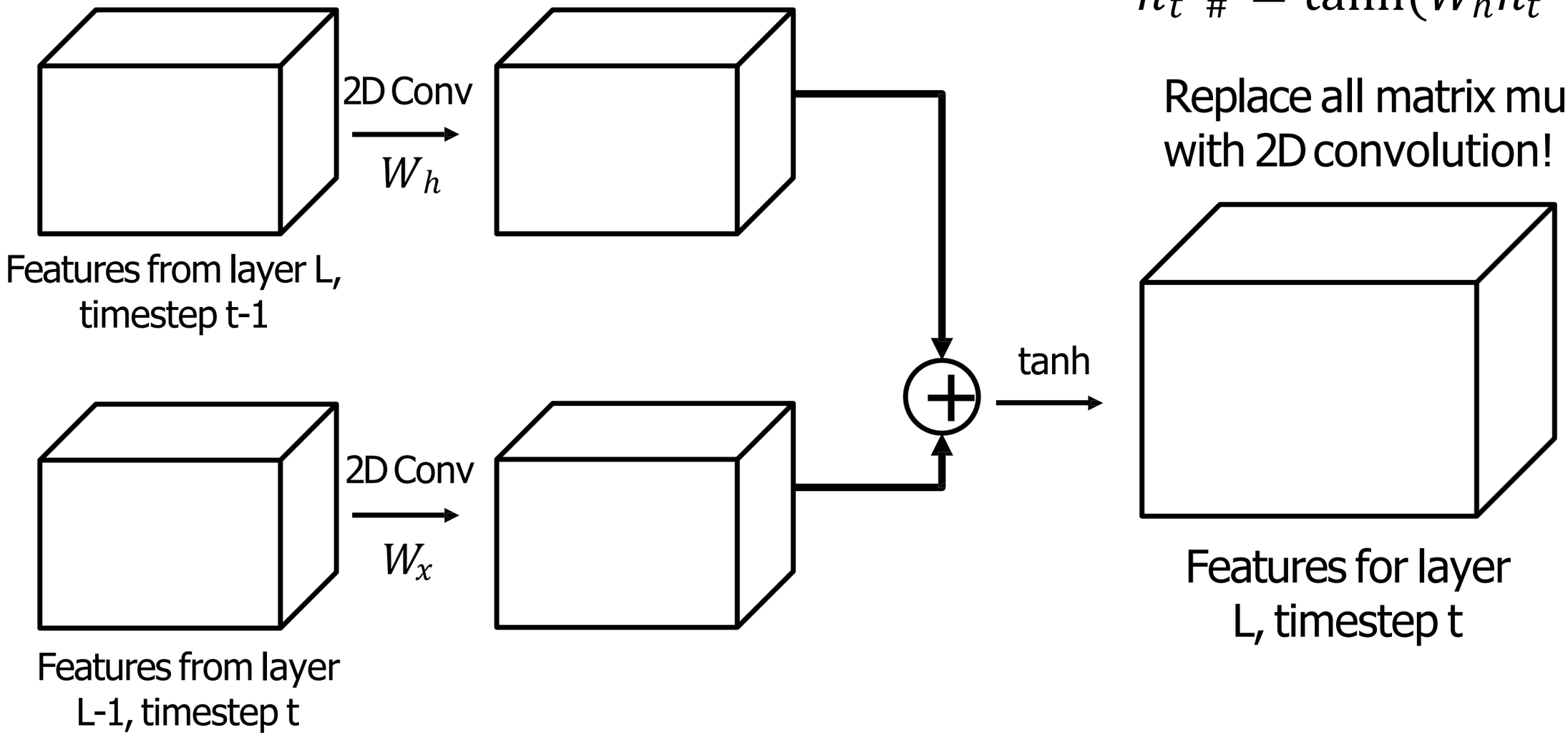
Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Recurrent Convolutional Network

Recall: Vanilla RNN

$$h_{t"\#} = \tanh(W_h h_t + W_x x)$$

Replace all matrix multiply with 2D convolution!

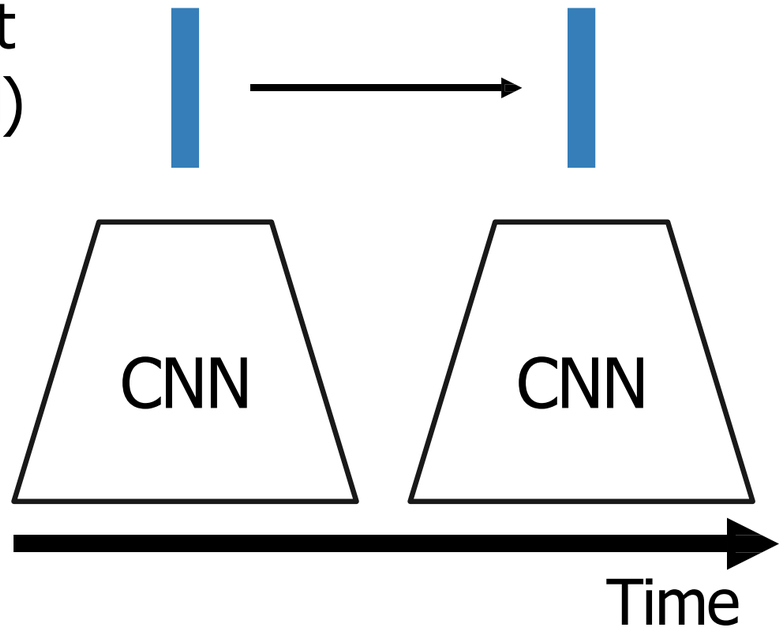


Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

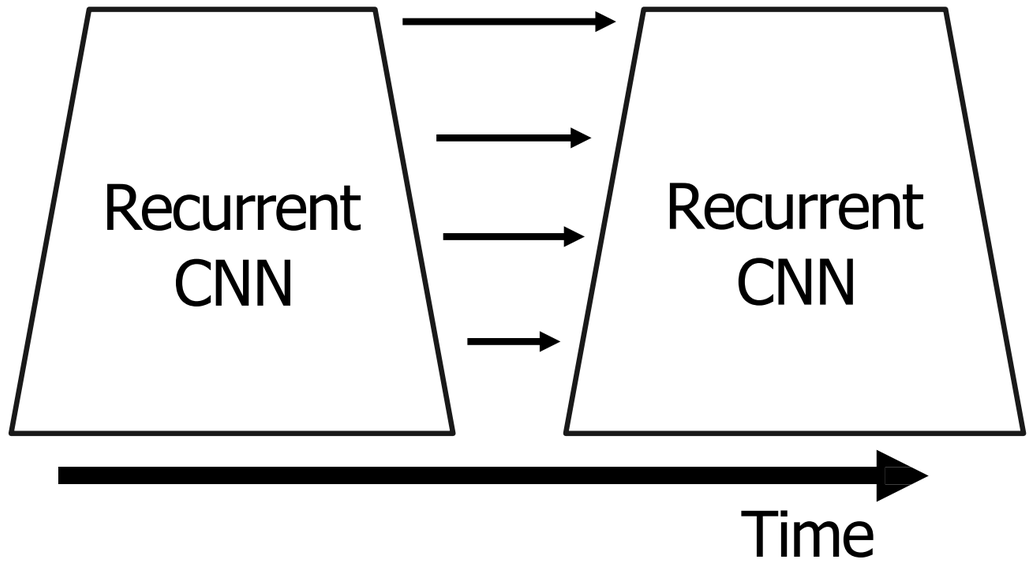
Modeling long-term temporal structure

RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)



Recurrent CNN: Infinite temporal extent (convolutional)



Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

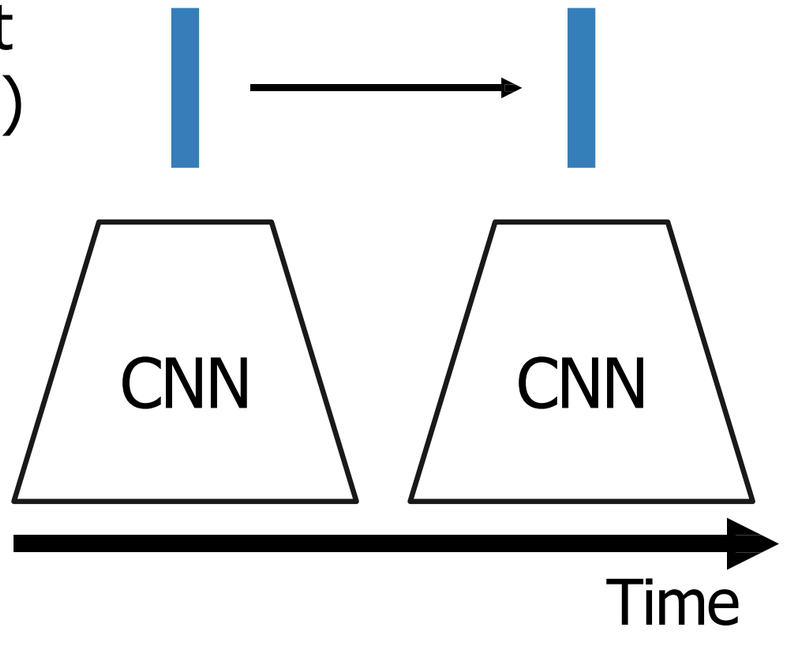
Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Modeling long-term temporal structure

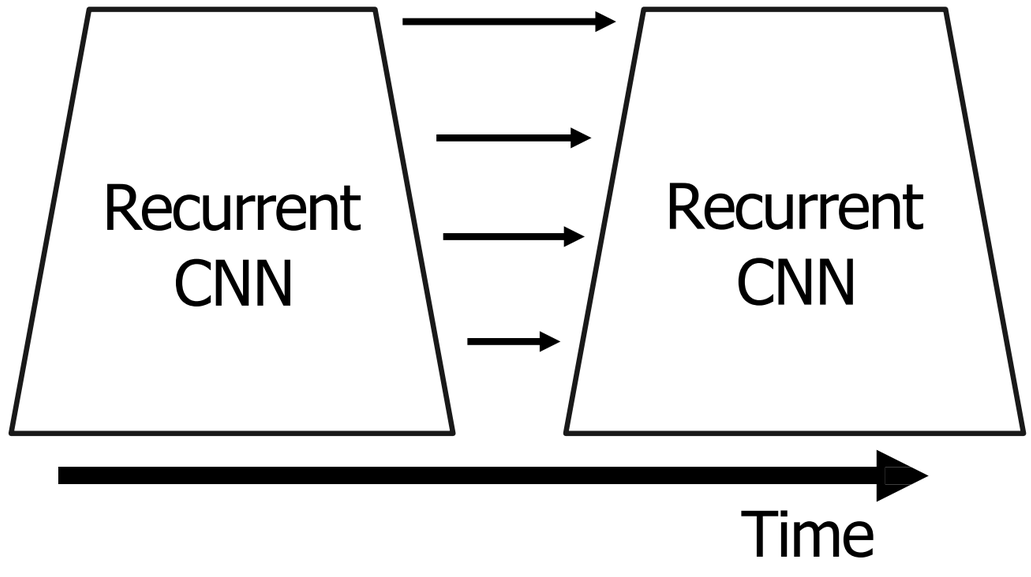
Problem: RNNs are slow for long sequences (can't be parallelized)

RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)



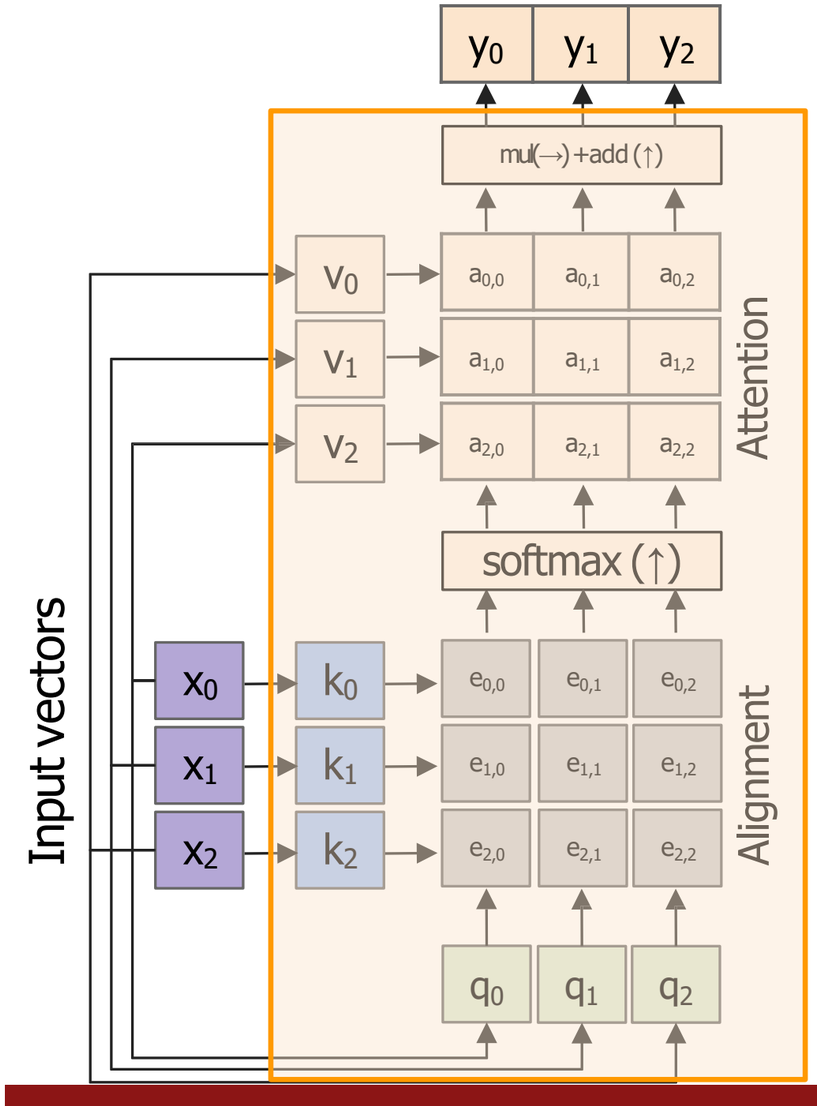
Recurrent CNN: Infinite temporal extent (convolutional)



Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

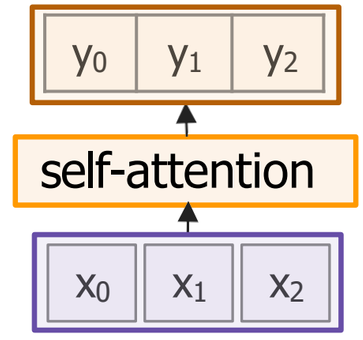
Recall: Self-Attention



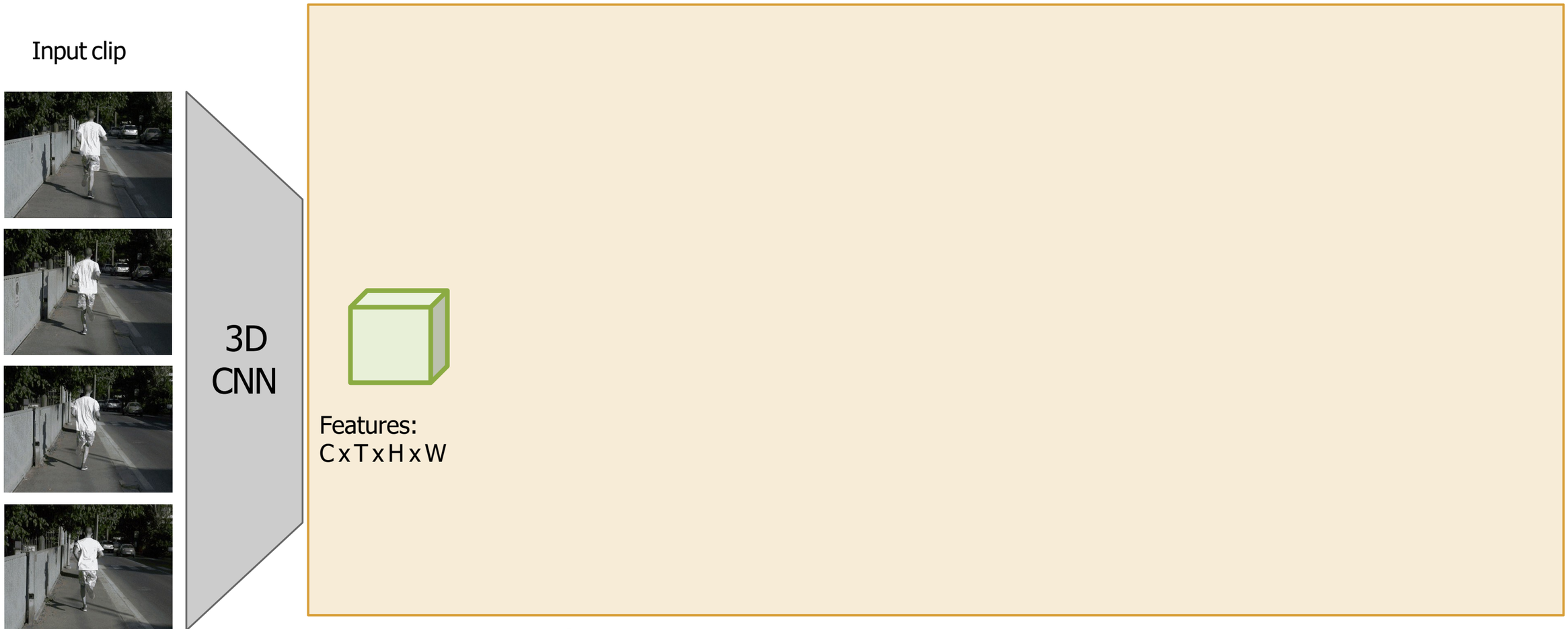
Outputs:
context vectors: y (shape: D_v)

Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Query vectors: $q = xW_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: x (shape: N x D)

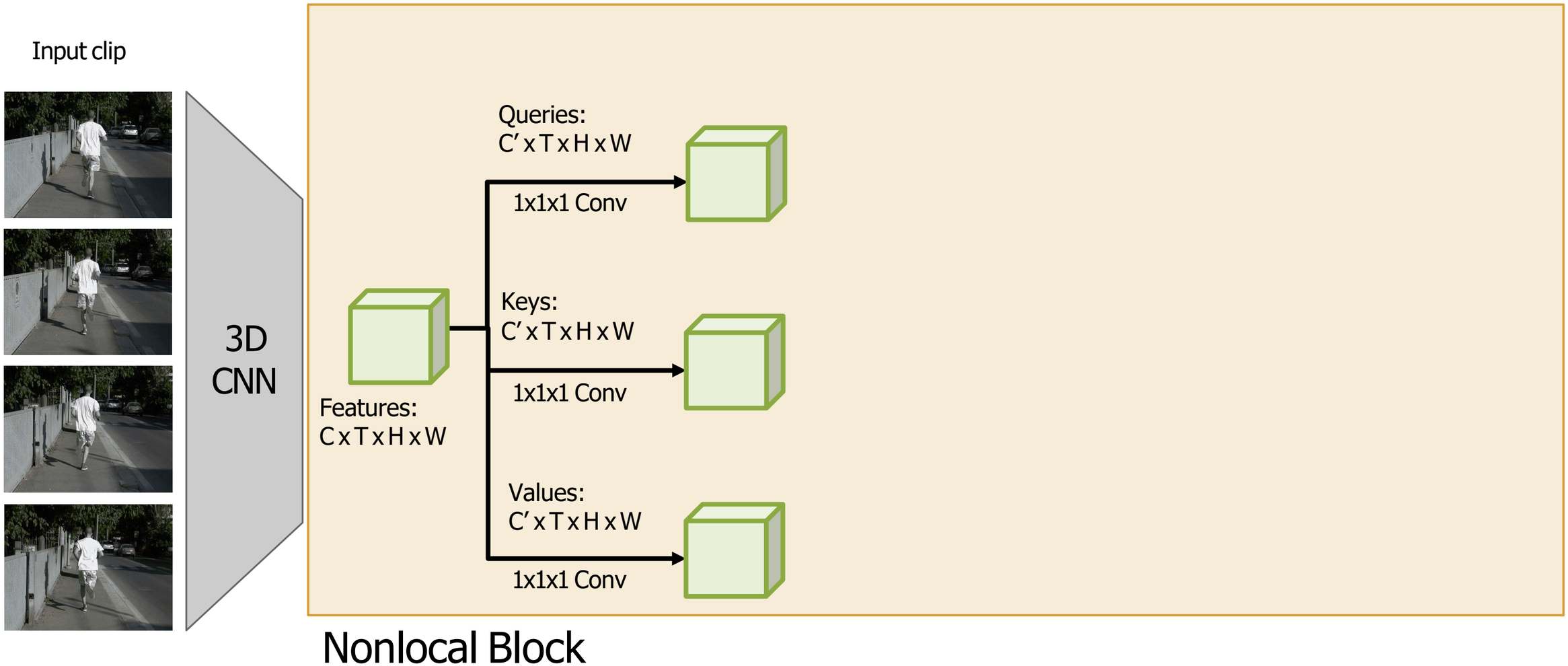


Spatio-Temporal Self-Attention (Nonlocal Block)



Wang et al, "Non-local neural networks", CVPR 2018

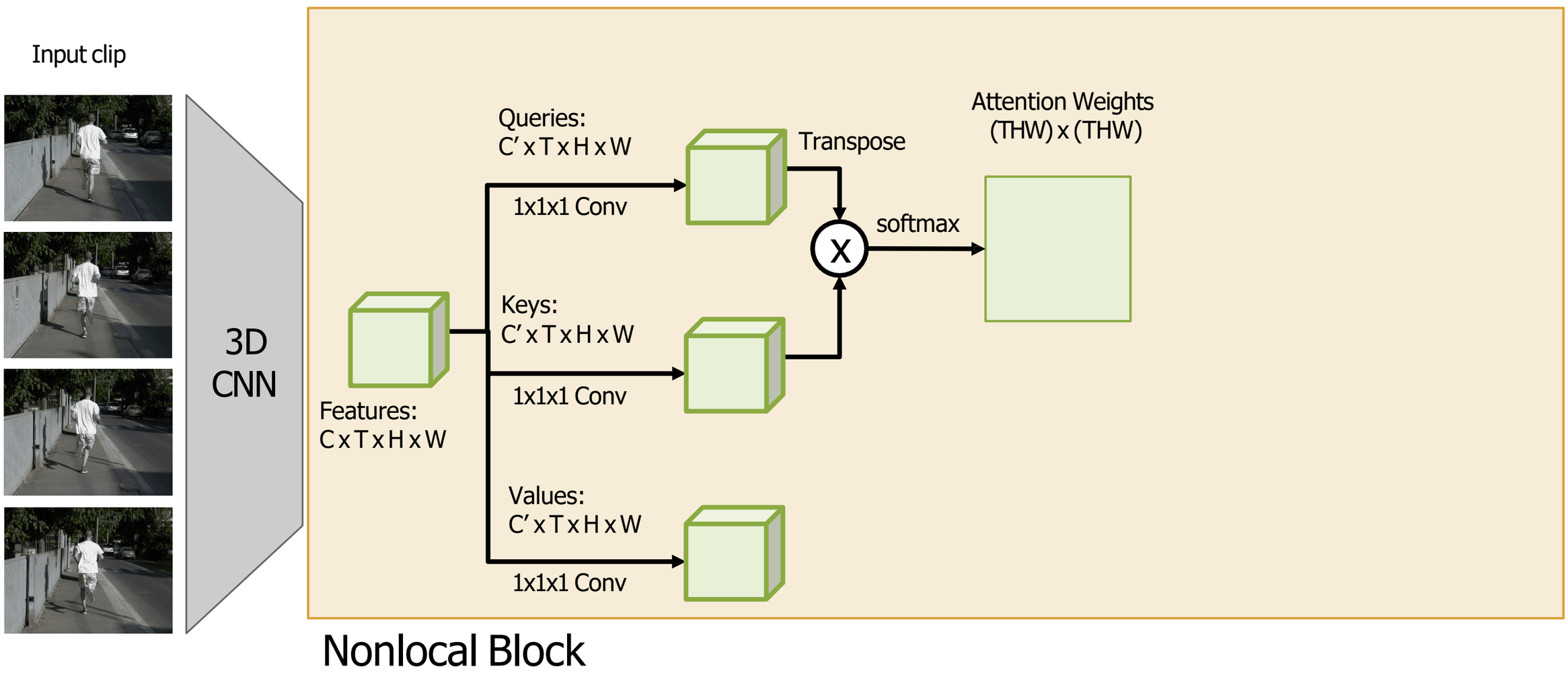
Spatio-Temporal Self-Attention (Nonlocal Block)



Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

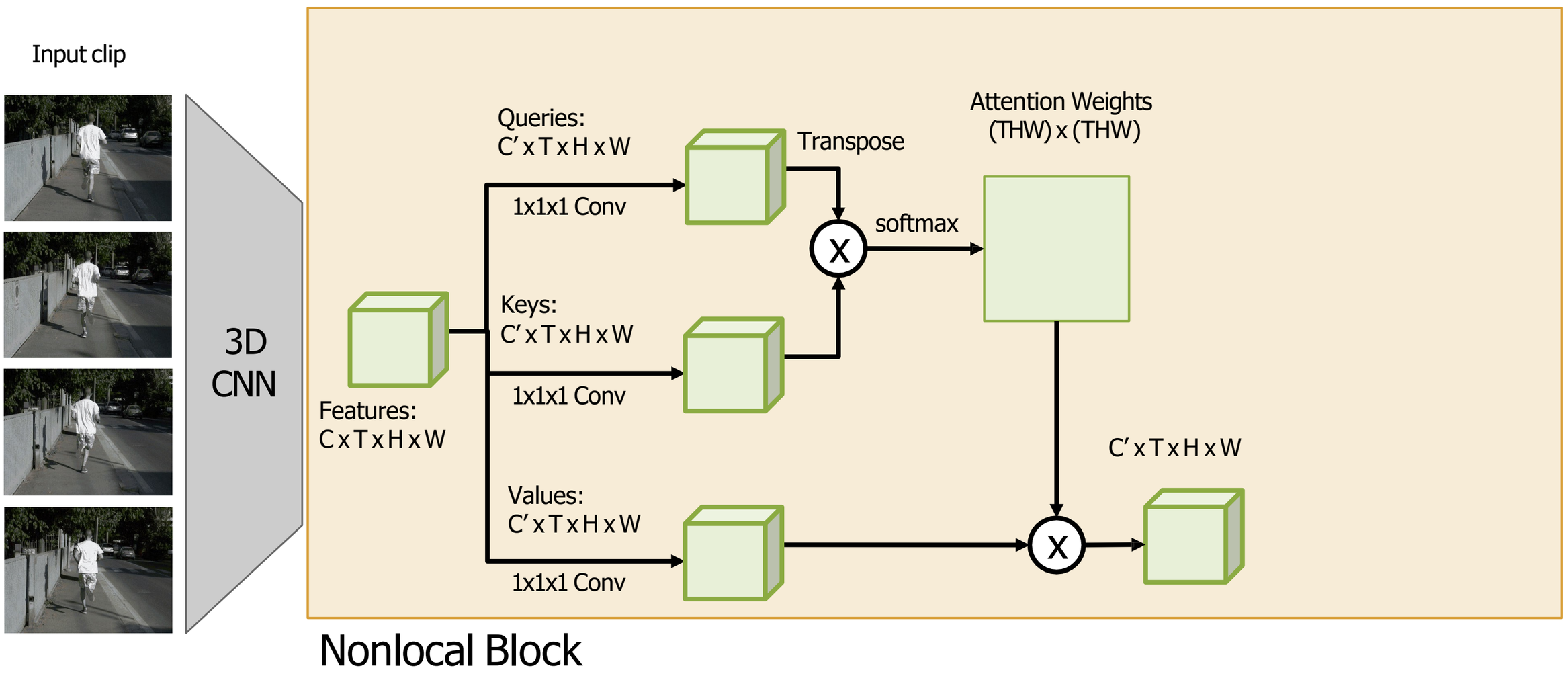
Spatio-Temporal Self-Attention (Nonlocal Block)



Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

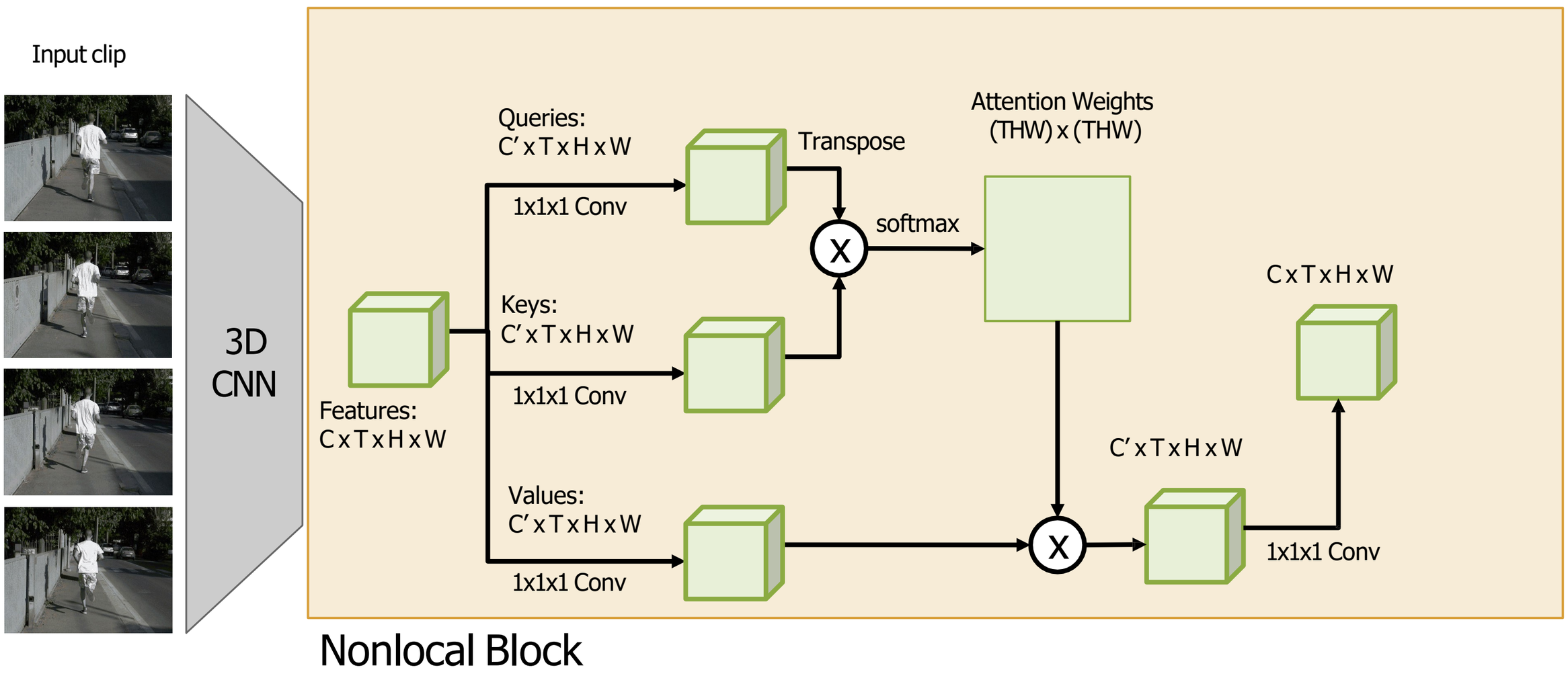
Spatio-Temporal Self-Attention (Nonlocal Block)



Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

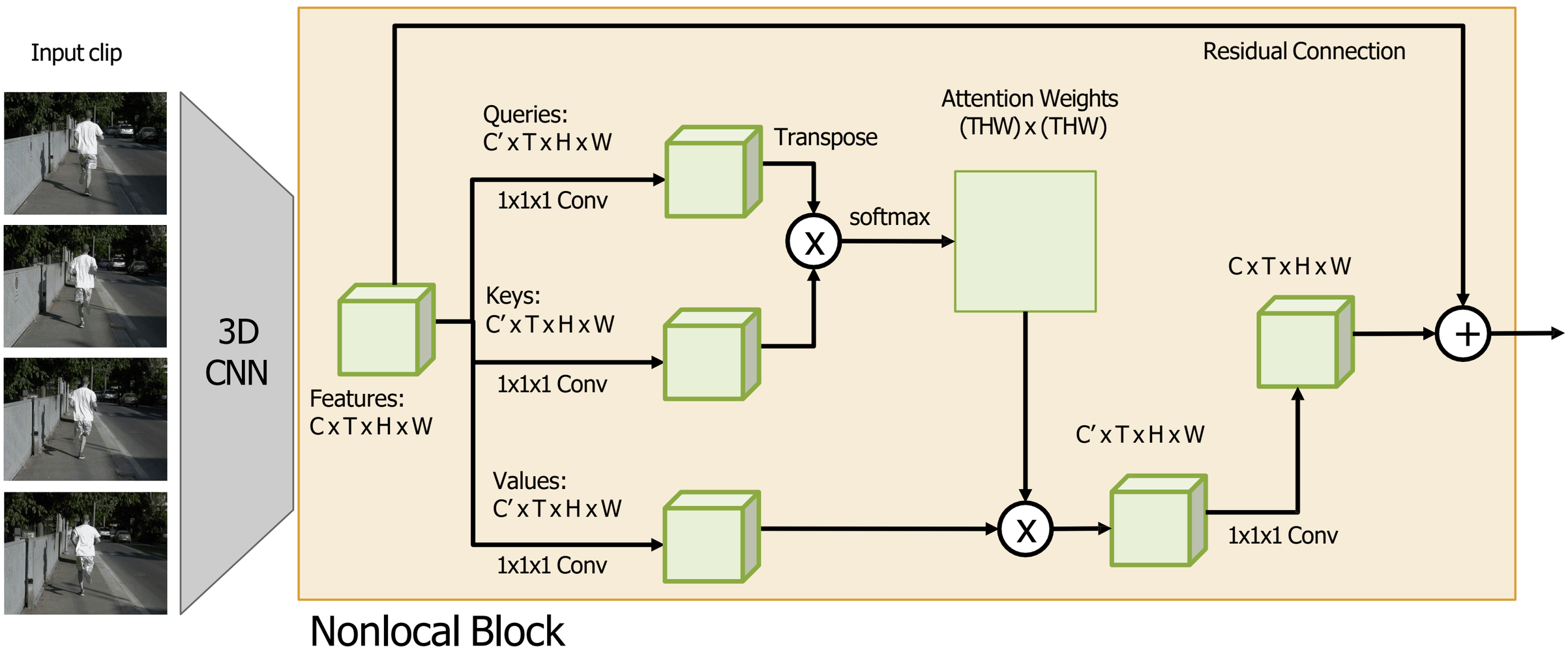
Spatio-Temporal Self-Attention (Nonlocal Block)



Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

Spatio-Temporal Self-Attention (Nonlocal Block)



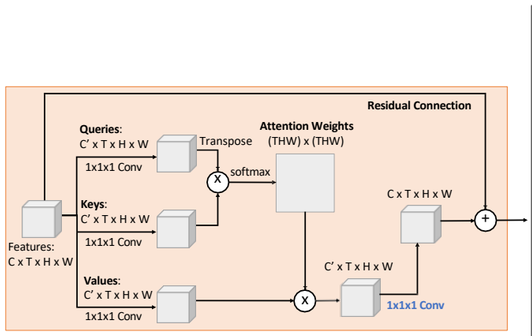
Wang et al, "Non-local neural networks", CVPR 2018

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

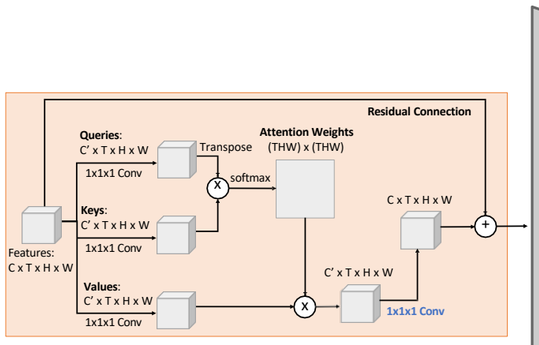
3D CNN

We can add nonlocal blocks into existing 3D CNN architectures.
But what is the best 3D CNN architecture?



Nonlocal Block

3D CNN



Nonlocal Block

3D CNN

Running

Wang et al, "Non-local neural networks", CVPR 2018

Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

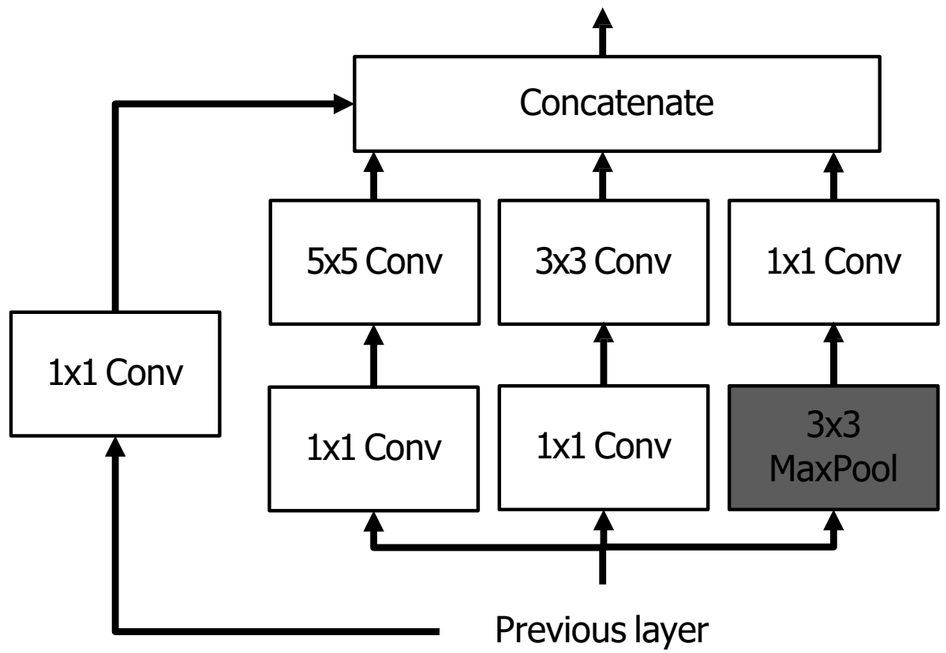
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Inception Block: Original



Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

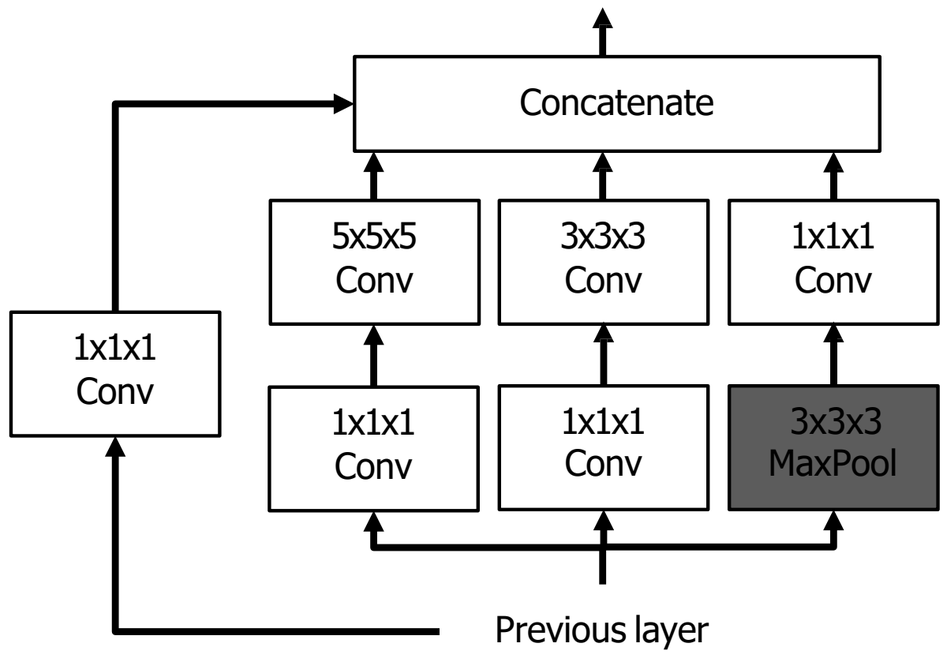
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Inception Block: Inflated



Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

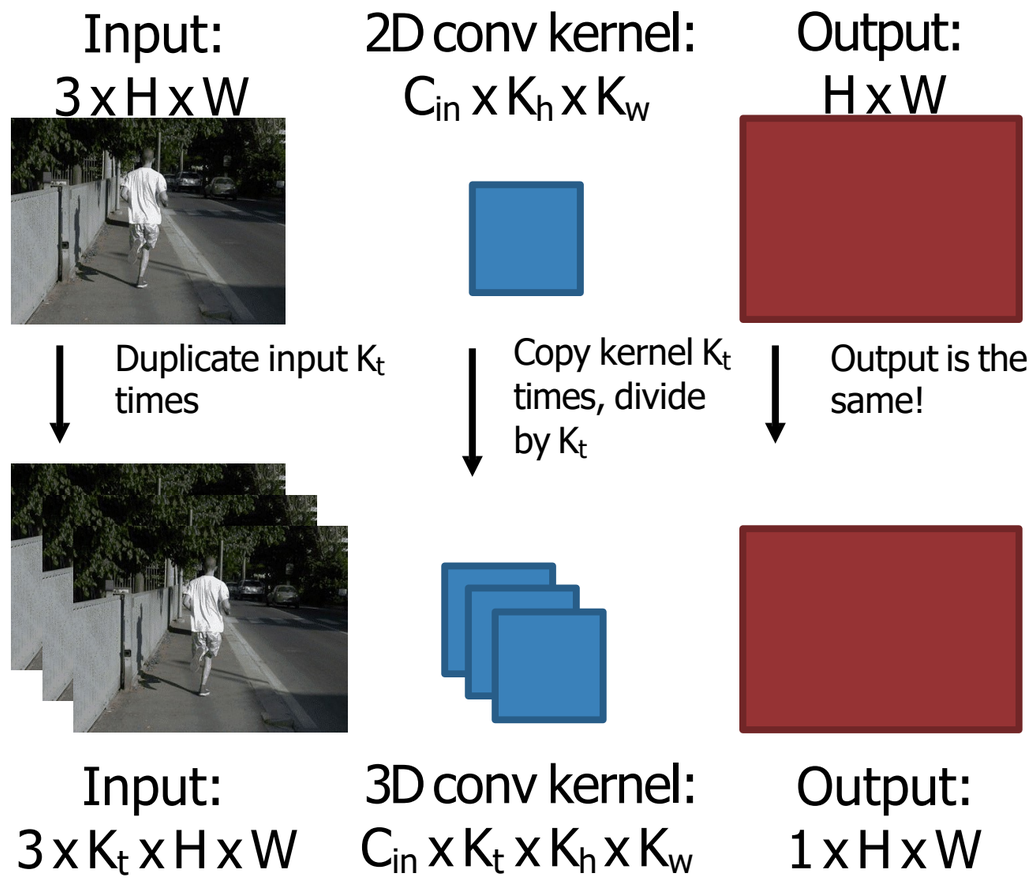
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input



Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

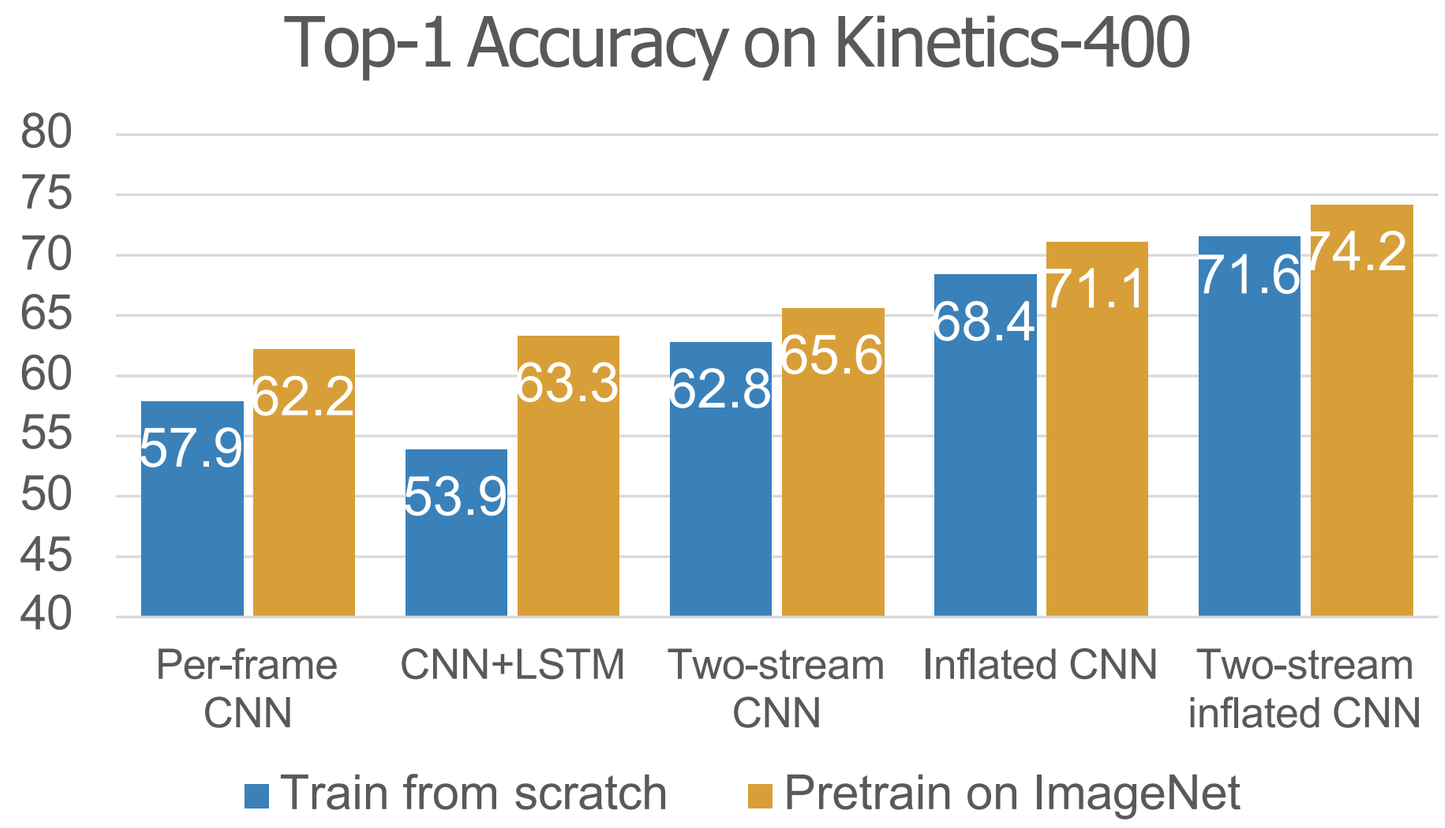
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input

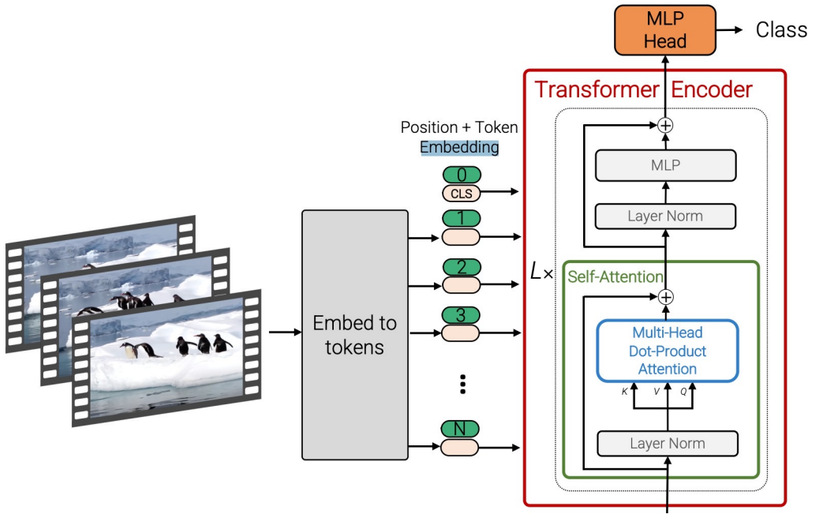


Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

All using Inception CNN

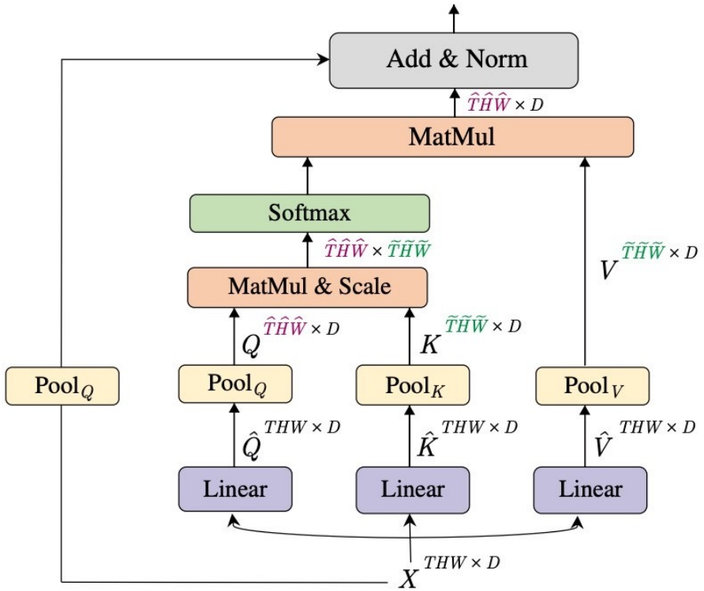
Vision Transformers for Video

Factorized attention:
Attend over space / time



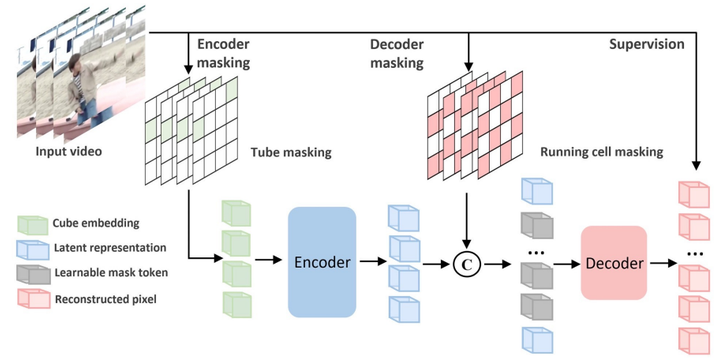
Bertasius et al, "Is Space-Time Attention All You Need for Video Understanding?", ICML 2021
Arnab et al, "ViViT: A Video Vision Transformer", ICCV 2021
Neimark et al, "Video Transformer Network", ICCV 2021

Pooling module:
Reduce number of tokens



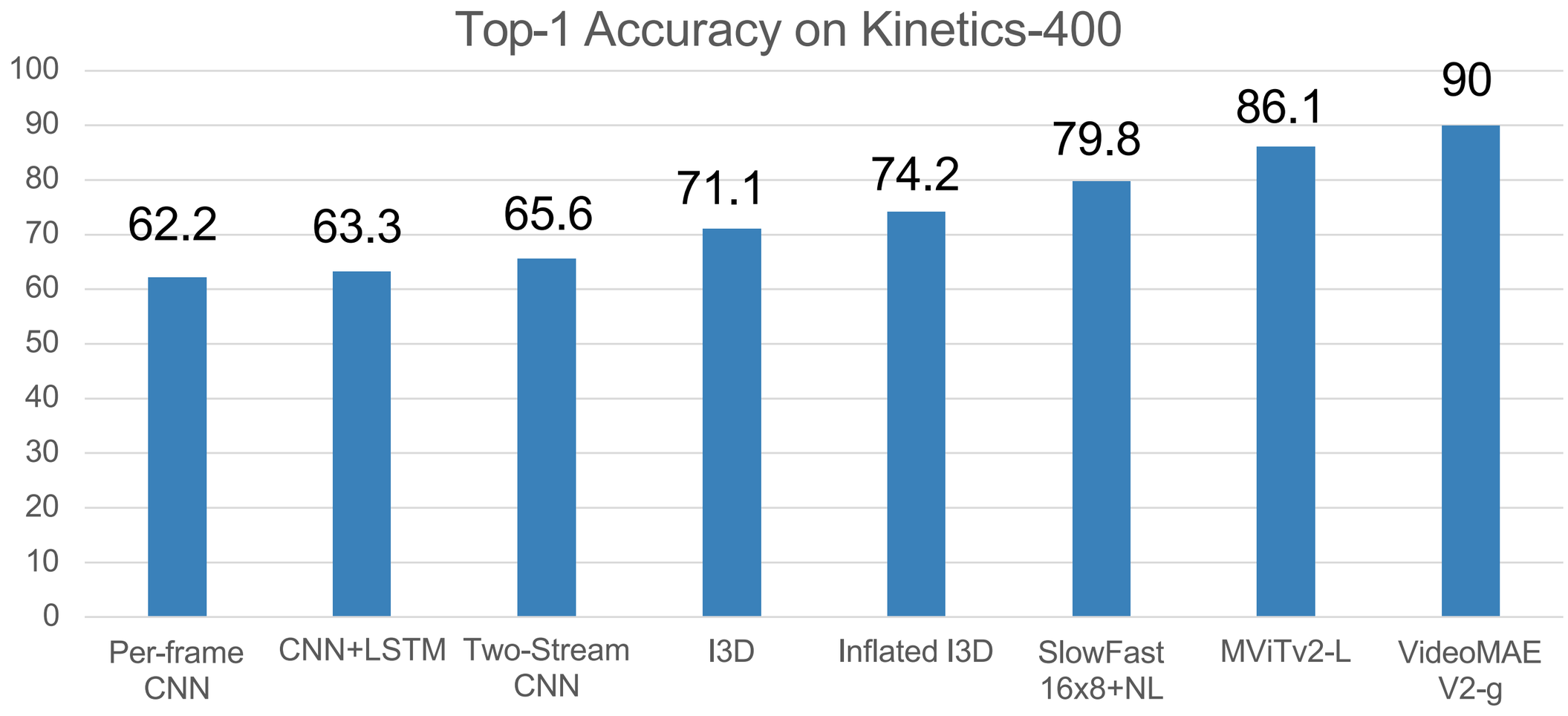
Fan et al, "Multiscale Vision Transformers", ICCV 2021
Li et al, "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection", CVPR 2022

Video masked autoencoders:
Efficient scalable pretraining

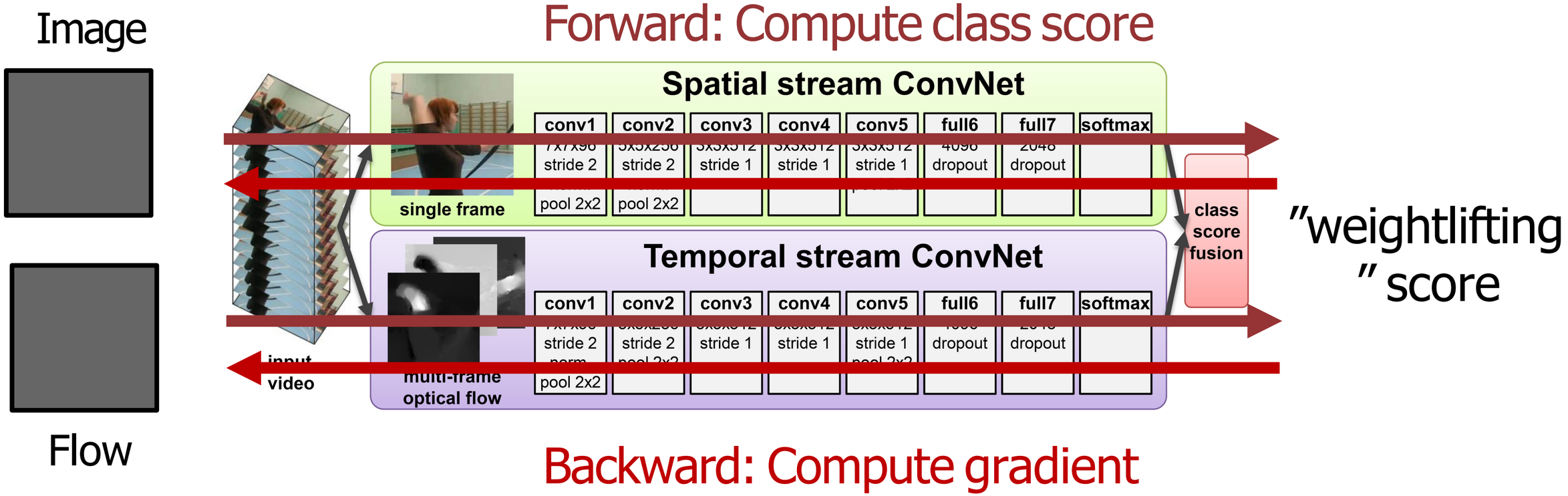


Wang et al. VideoMAE V2: Scaling Video Masked Autoencoders with Dual Making. CVPR 2023.
Tong et al. Video MAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. NeurIPS 2022.
Feichtenhofer et al. Masked autoencoders as spatiotemporal learners. NeurIPS 2022.

Vision Transformers for Video



Visualizing Video Models



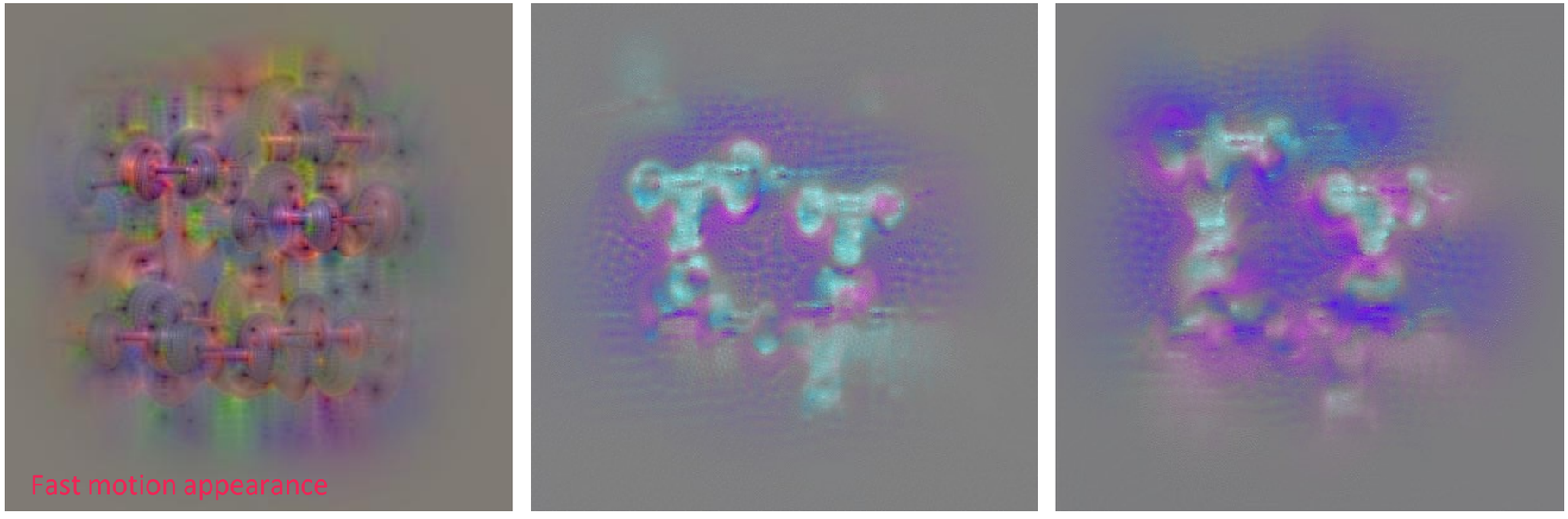
Add a term to encourage spatially smooth flow; tune penalty to pick out "slow" vs "fast" motion

Figure credit: Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014
Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018
Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.

Can you guess the action?

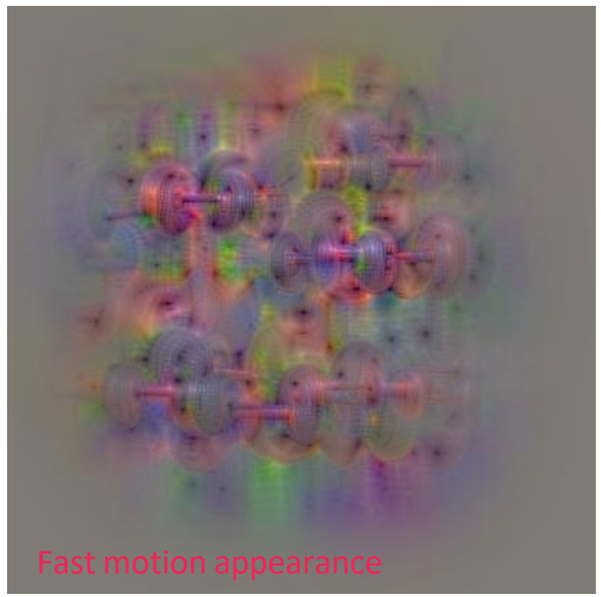
Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018
Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.
Slide credit: Christoph Feichtenhofers

Appearance | "Slow" motion | "Fast" motion



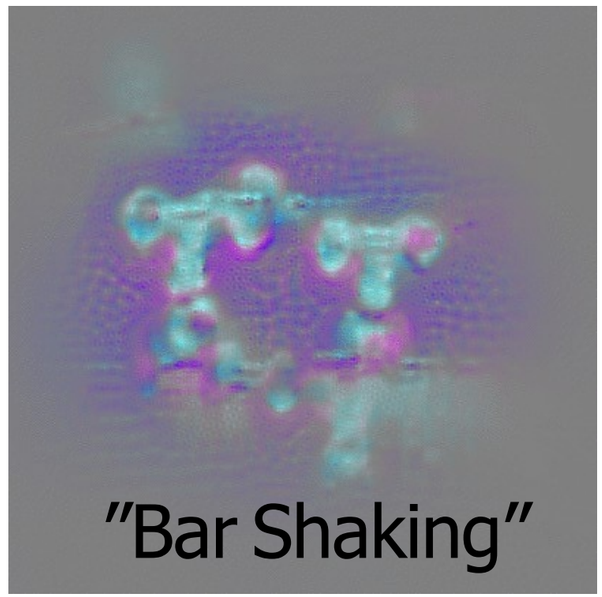
Can you guess the action? Weightlifting

Appearance



Fast motion appearance

"Slow" motion



"Bar Shaking"

"Fast" motion

"Push overhead"

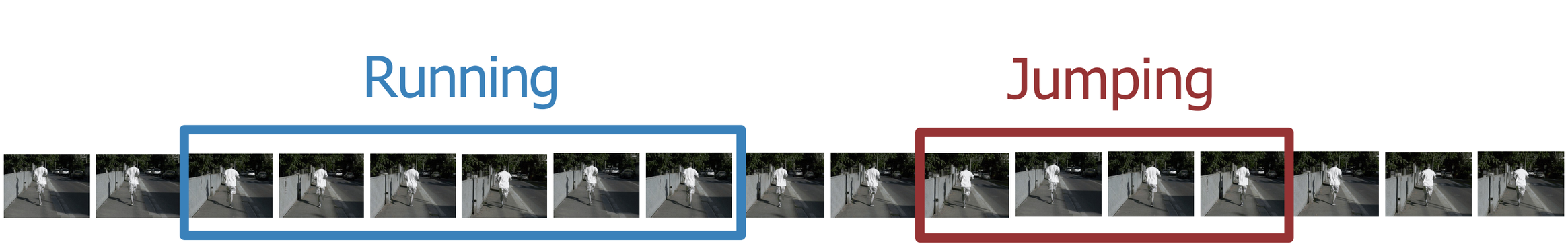
So far: Classify short clips

Videos: Recognize actions

Swimming
Running
Jumping
Eating
Standing

Temporal Action Localization

Given a long untrimmed video sequence, identify frames corresponding to different actions

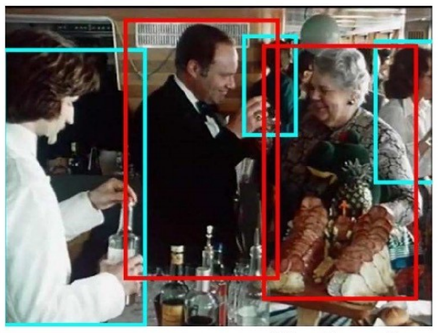


Can use architecture similar to Faster R-CNN: first generate temporal proposals then classify

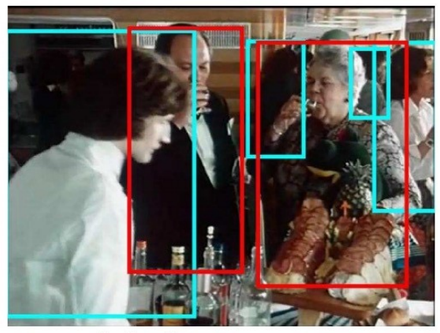
Chao et al, ” Rethinking the Faster R-CNN Architecture for Temporal Action Localization”, CVPR 2018

Spatio-Temporal Detection

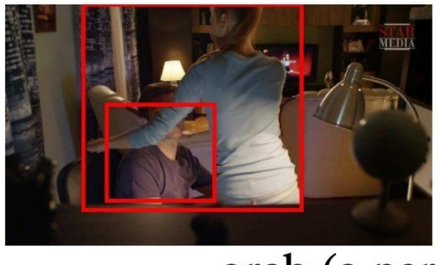
Given a long untrimmed video, detect all the people in both space and time and classify the activities they are performing. Some examples from AVA Dataset:



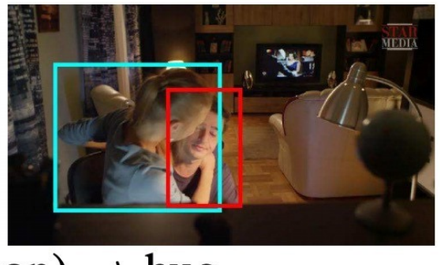
clink glass → drink



open → close



grab (a person) → hug



look at phone → answer phone

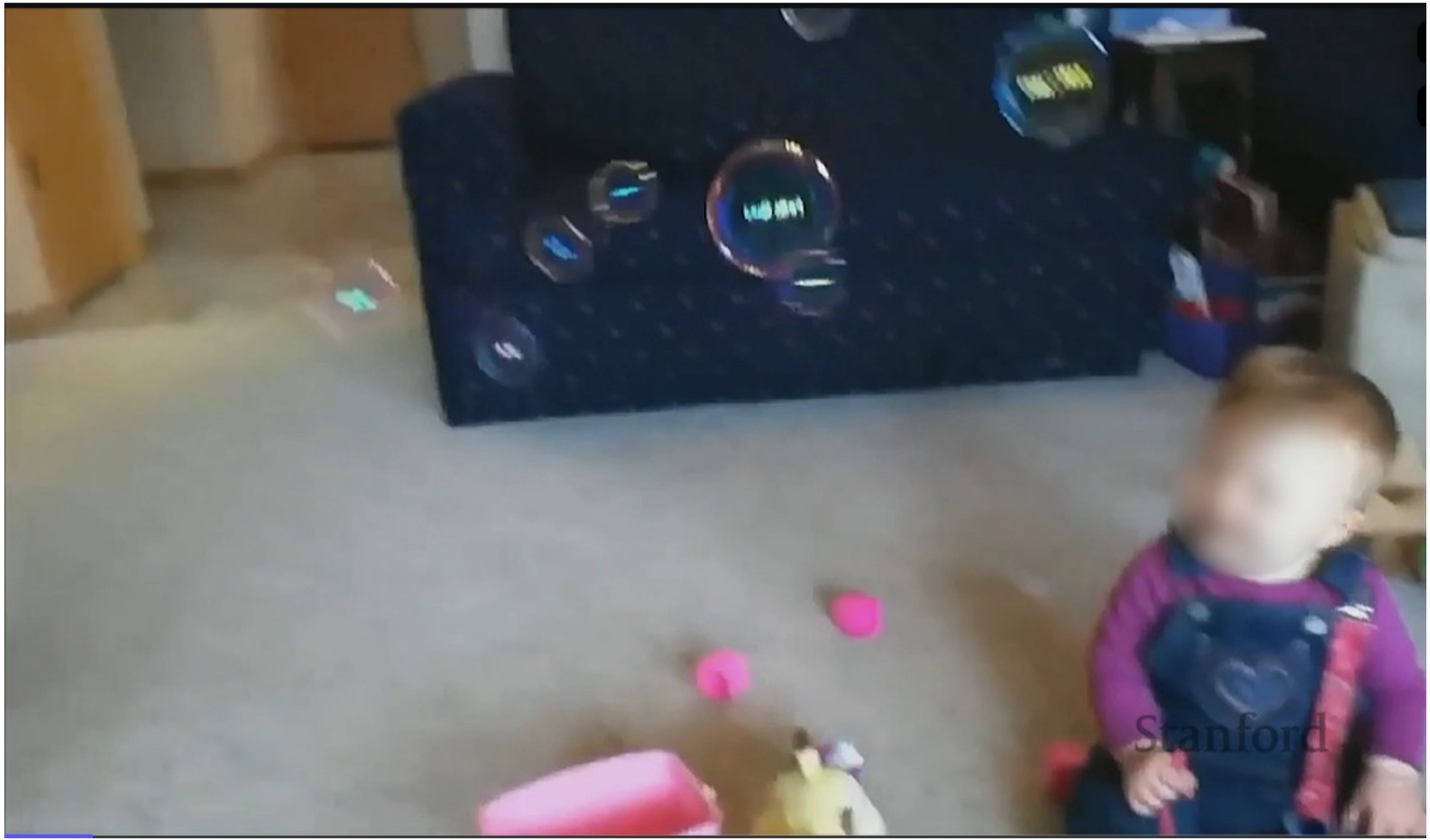
Gu et al, "AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions", CVPR 2018

Today: Temporal Stream

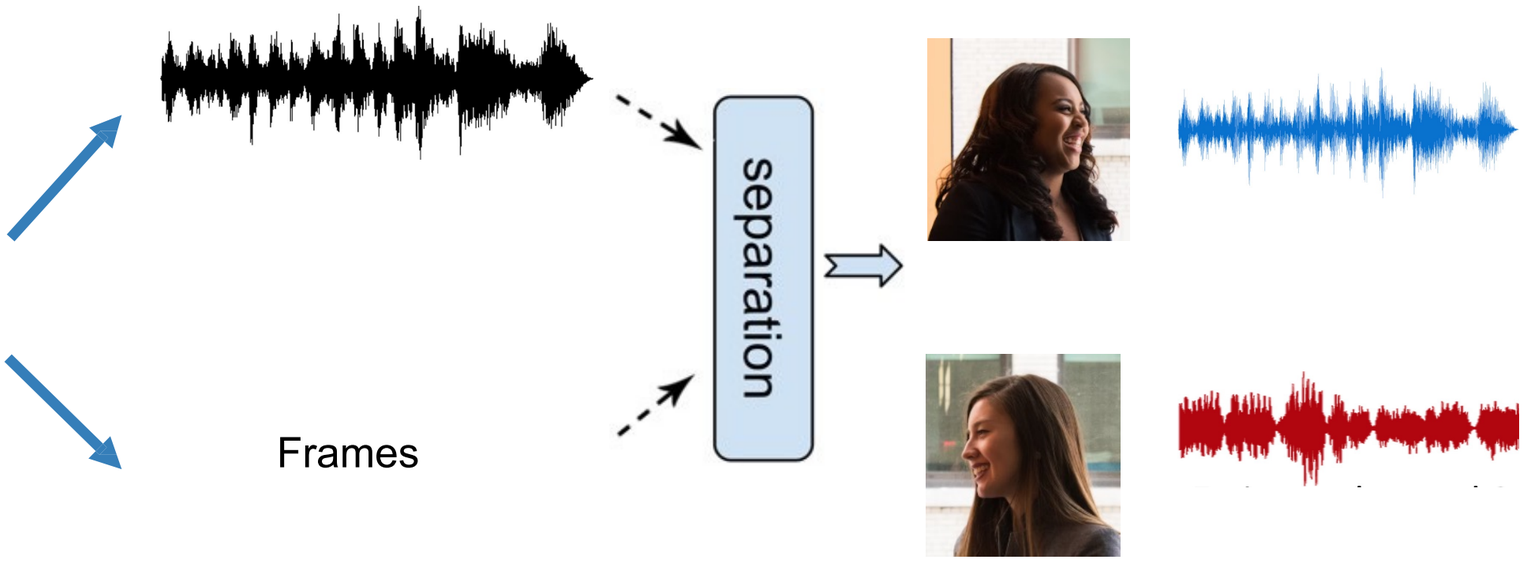
...

...

3D CNN, Two-Stream Neural Network, Spatial-Temporal Self-Attention……


Stanford

Visually-guided audio source separation



[Gao et al. ECCV 2018, Afouras et al. Interspeech'18, Gabby et al. Interspeech'18, Owens & Efros ECCV'18, Ephrat et al. SIGGRAPH'18, Zhao et al. ECCV 2018, Gao & Grauman ICCV 2019, Zhao et al. ICCV 2019, Xu et al. ICCV 2019, Gan et al. CVPR 2020, Gao et al. CVPR 2021, Tzinis et al. ECCV 2022, Chen et al. CVPR 2023] Images: Public Domain- Unsplash

Speech Mixture Video:

https://www.youtube.com/watch?v=Kxb9AUmSrIA

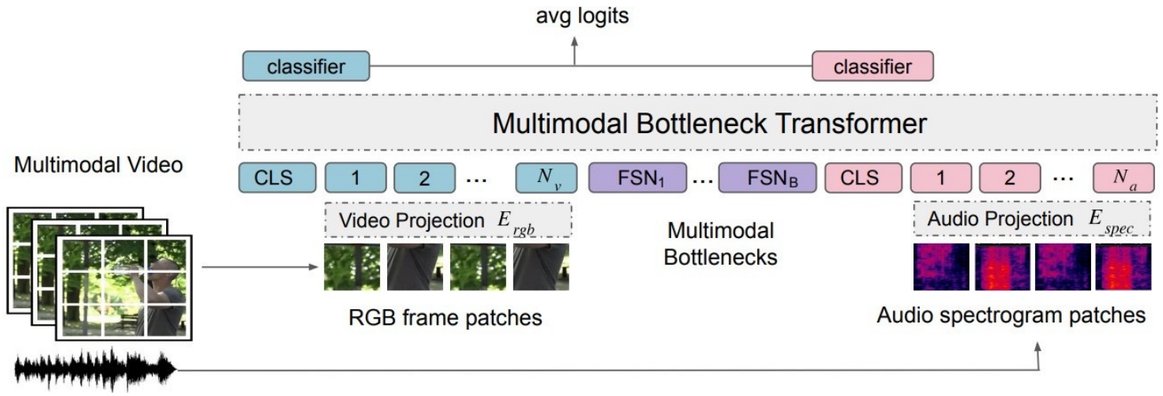
Musical instruments source separation

Train on 100,000 unlabeled multi-source video clips,
then separate audio for novel video.

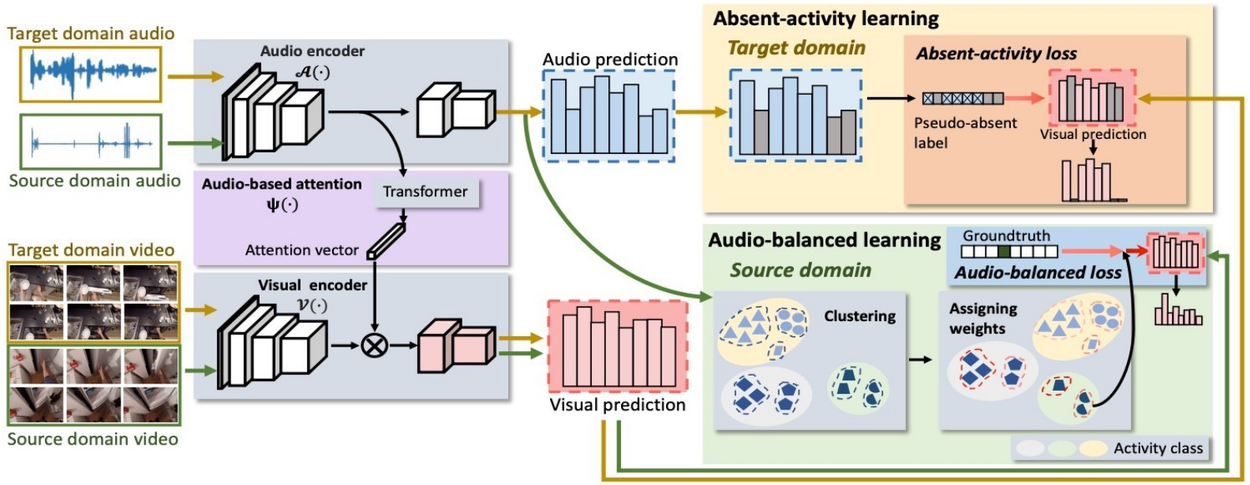
original video
(before separation)

object detections:
violin & flute

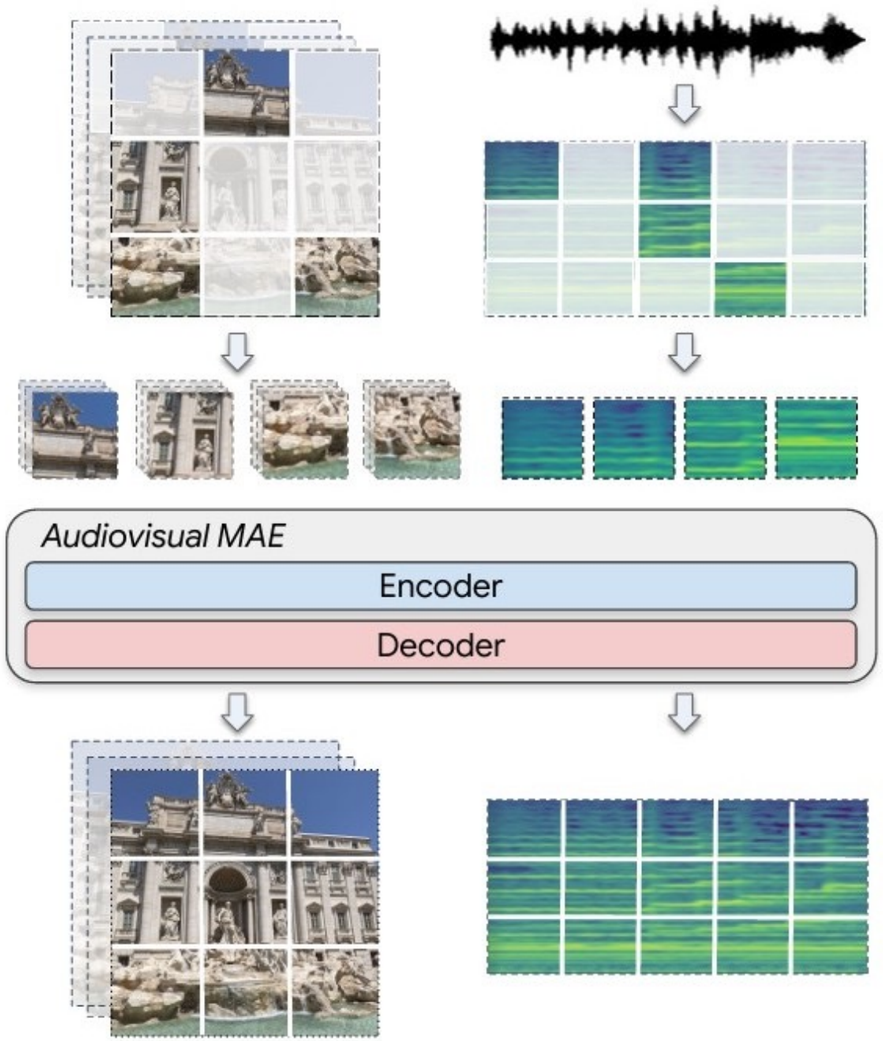
Audio-Visual Video Understanding



Attention Bottlenecks for Multimodal Fusion, Nagrani et al. NeurIPS 2021



Audio-Adaptive Activity Recognition Across Video Domains, Zhang et al. CVPR 2022



Audio-Visual Masked Autoencoders. Georgescu et al. ICCV 2023.

Efficient Video Understanding

Action recognition in long videos

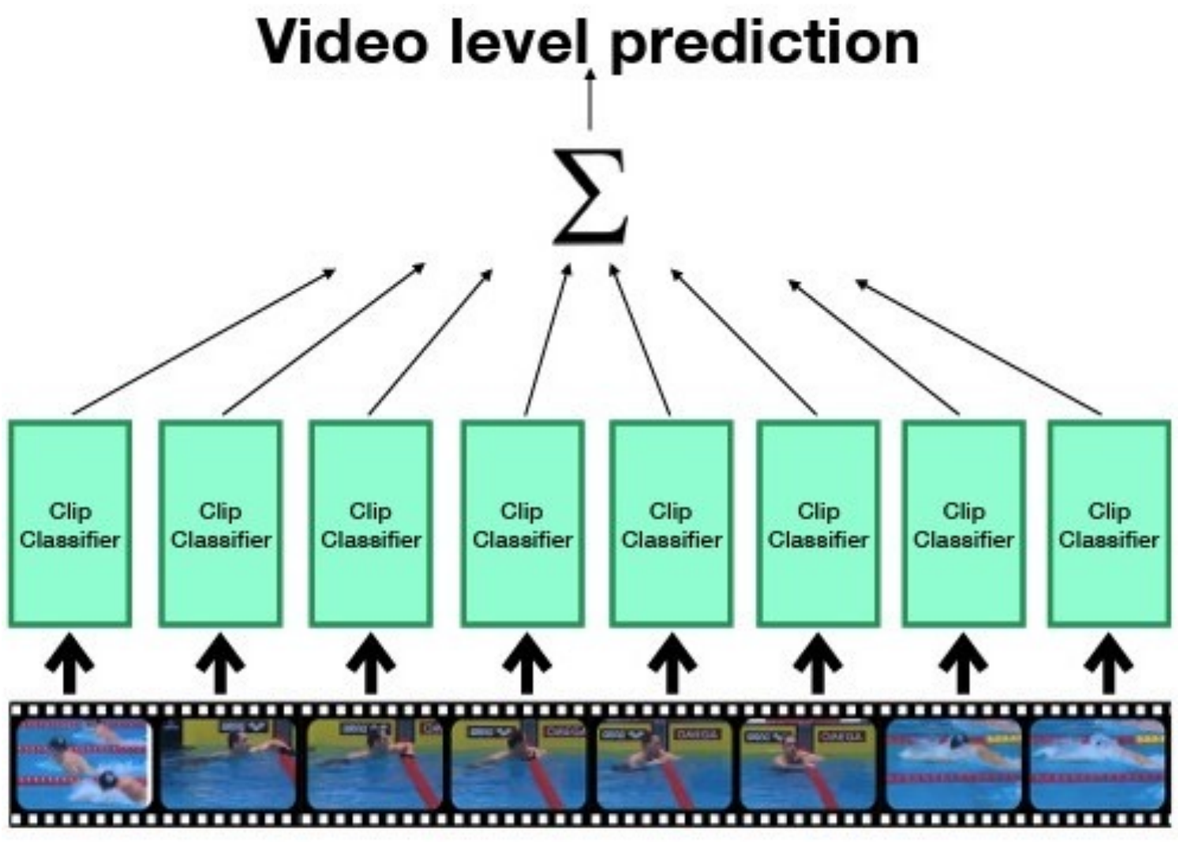
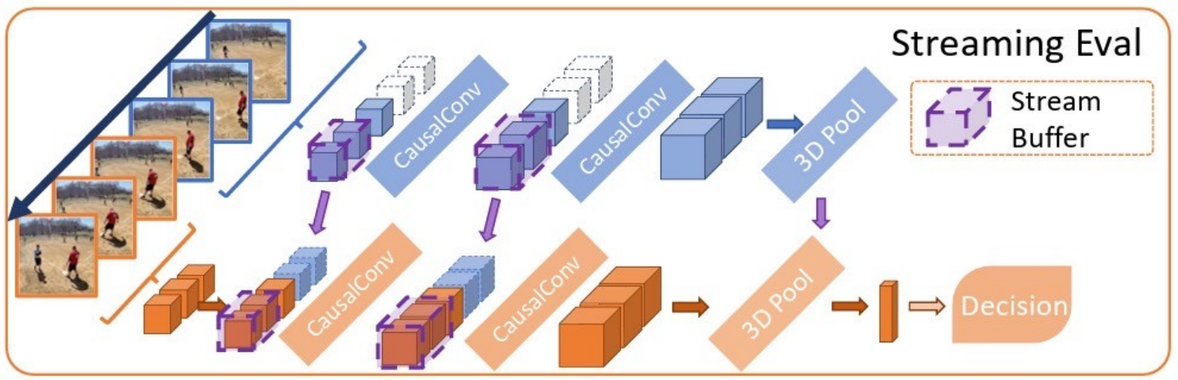
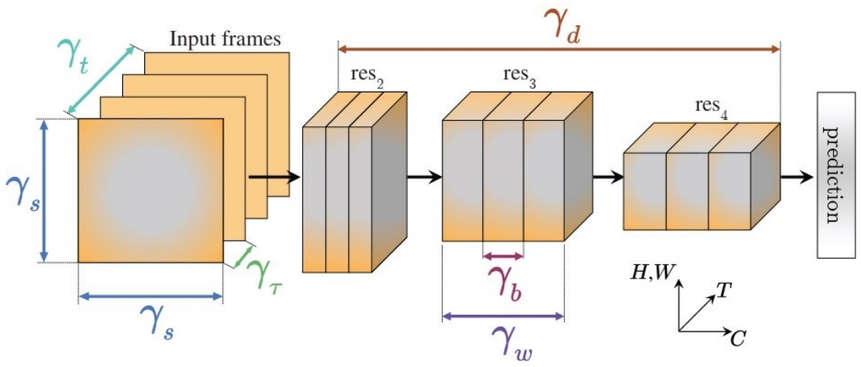


Image Credit: Korbar et al.

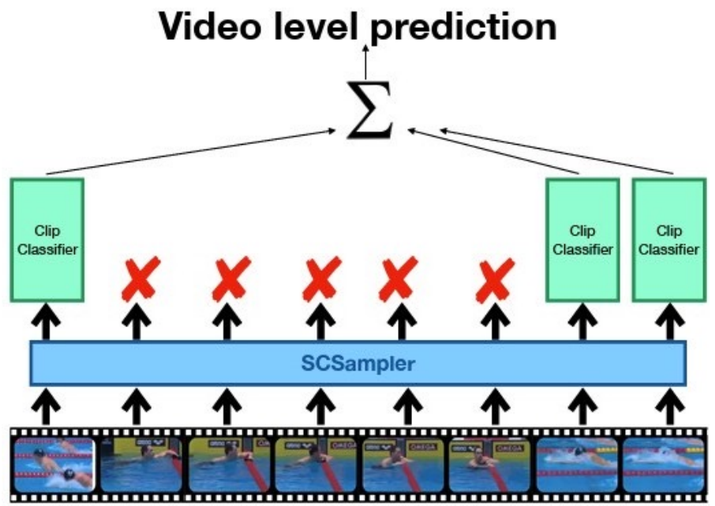
Efficient Video Understanding



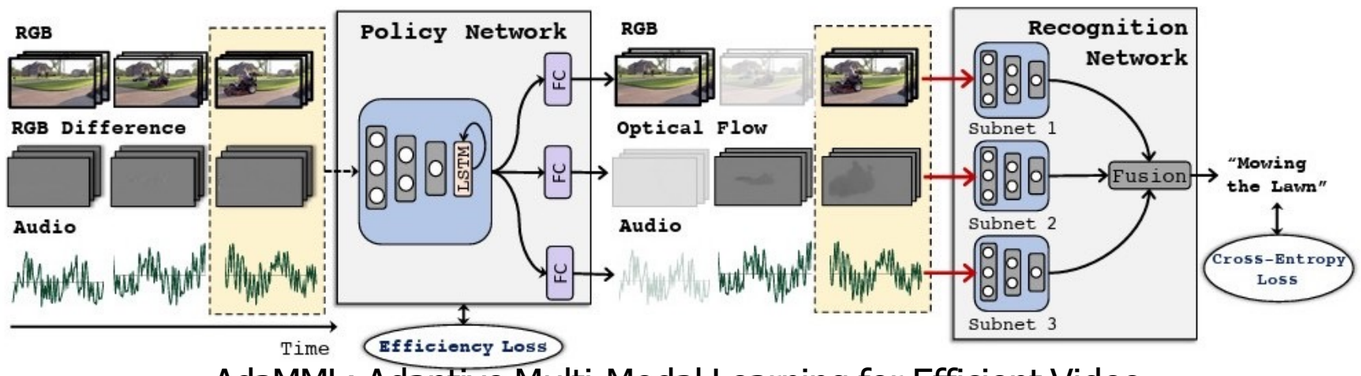
MoViNets: Mobile Video Networks for Efficient Video Recognition. Kondratyuk et al. CVPR 2021



X3D: Expanding Architecture for Efficient Video Recognition. Christoph Feichtenhofer. CVPR 2020.

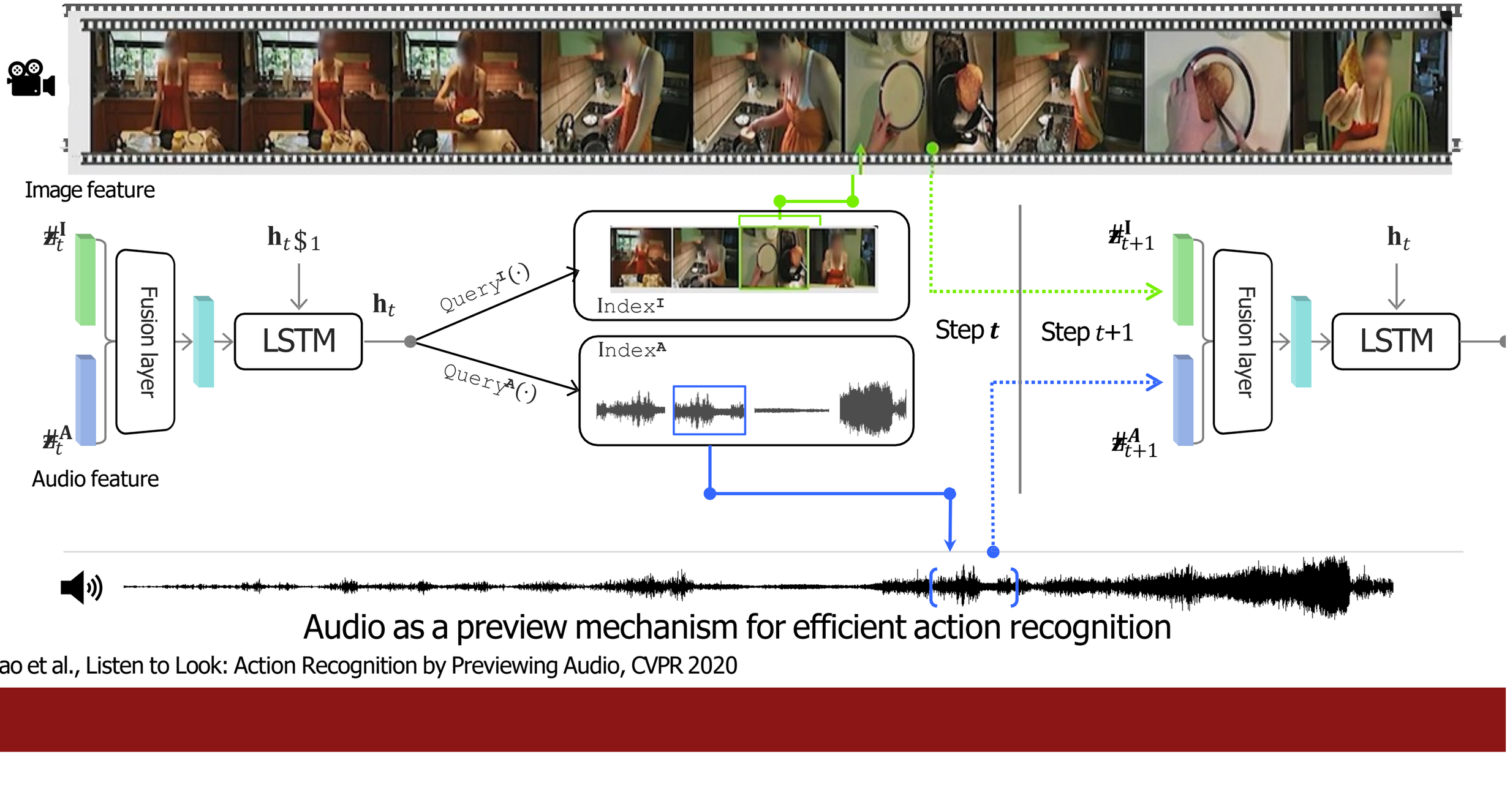


SCSampler: Sampling Salient Clips from Video for Efficient Action Recognition. Korbar et al. ICCV 2019



AdaMML: Adaptive Multi-Modal Learning for Efficient Video Recognition. Pandal et al. ICCV 2021

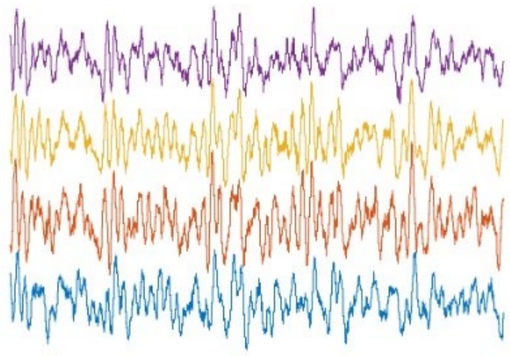
Efficient Video Understanding



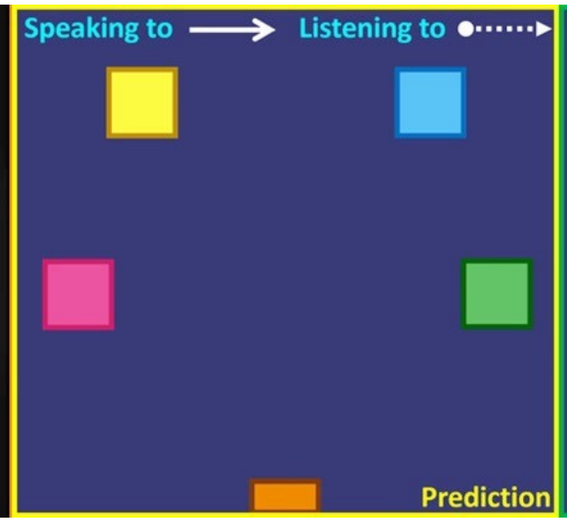
Audio as a preview mechanism for efficient action recognition

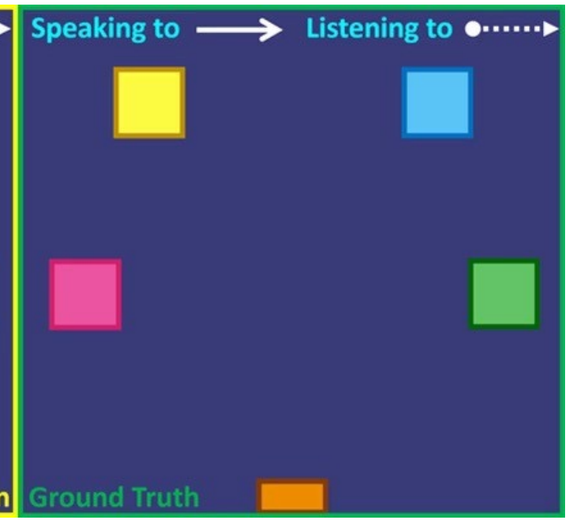
Gao et al., Listen to Look: Action Recognition by Previewing Audio, CVPR 2020

Multimodal Egocentric Video Understanding



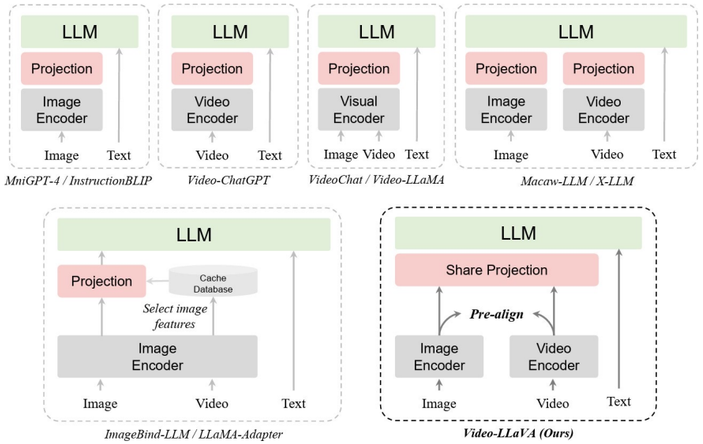
Ego-Exo Conversational Graph Prediction



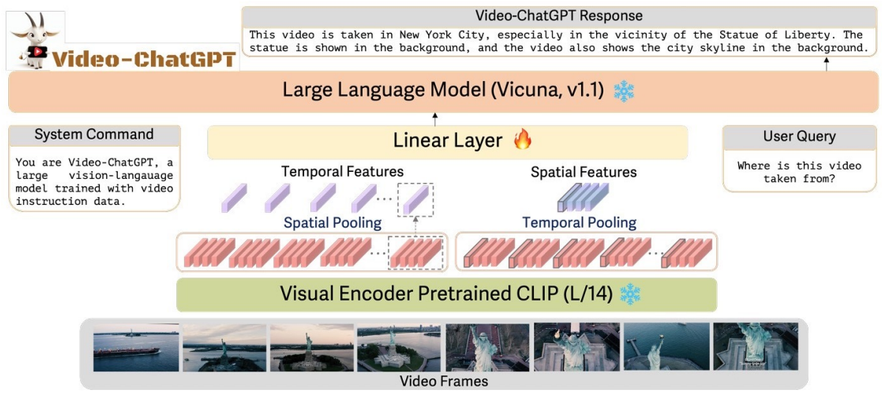


The Audio-Visual Conversational Graph: From and Egocentric-Exocentric Perspective. Jia et al. CVPR 2024

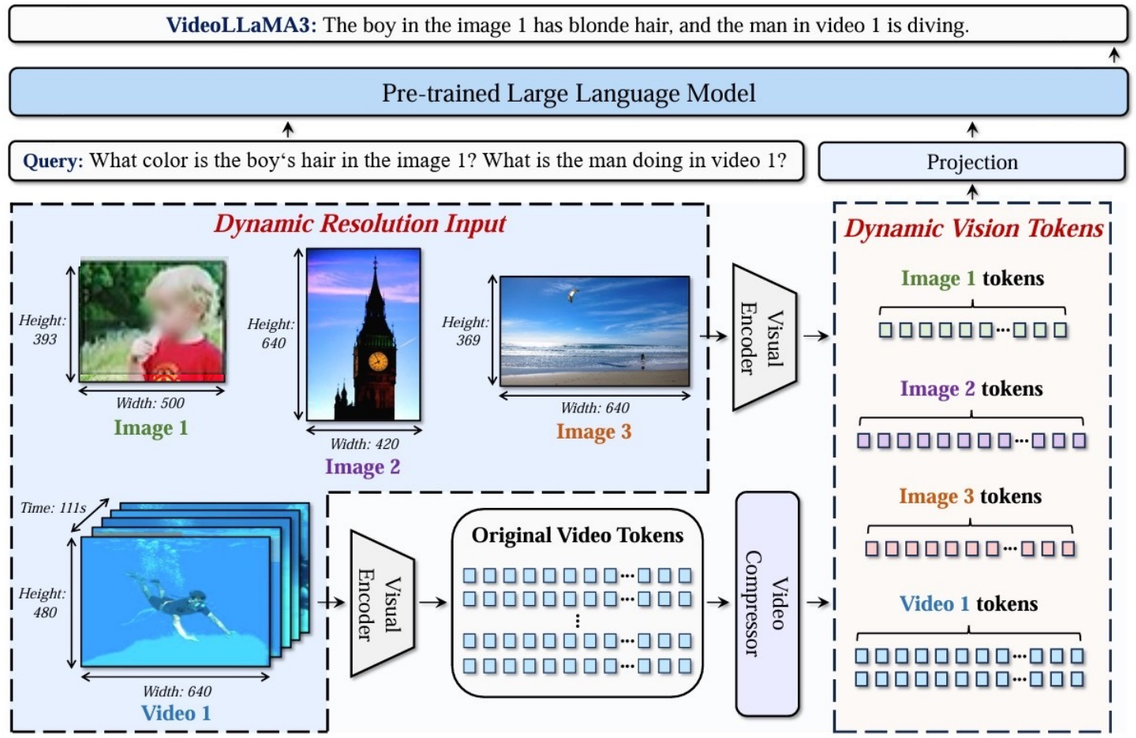
Video Understanding +LLMs



Video-LLaVA: Learning United Visual Representations by Alignment Before Projection. Lin et al. EMNLP 2024



Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language Models. Maaz et al. ACL 2024.



VideoLLaMA 3: Frontier Multimodal Foundation Models for Video Understanding. Zhang et al. arXiv 2025

Next time: Large Scale Distributed Training