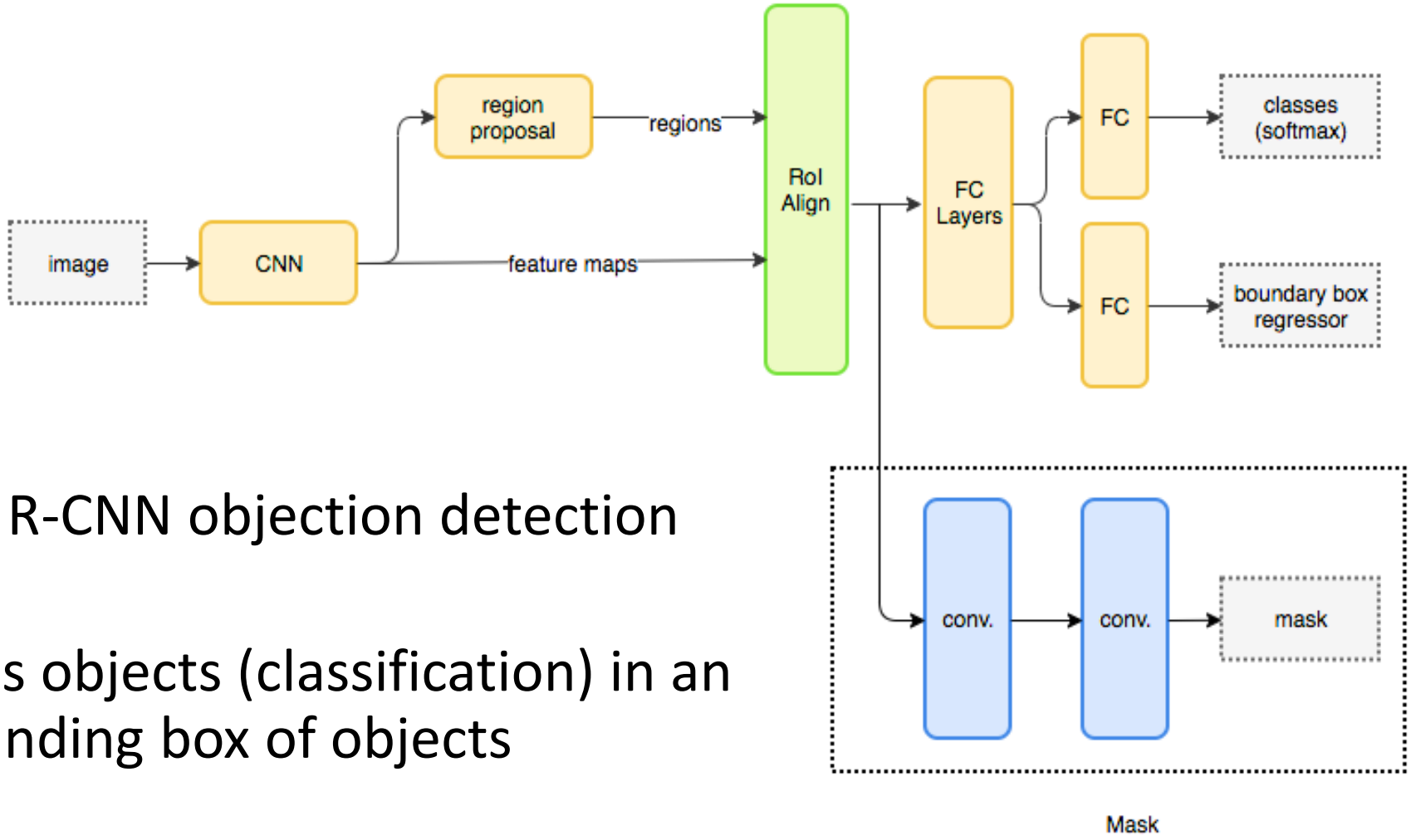
Mask R-CNN vs DeepLab

Mask R-CNN

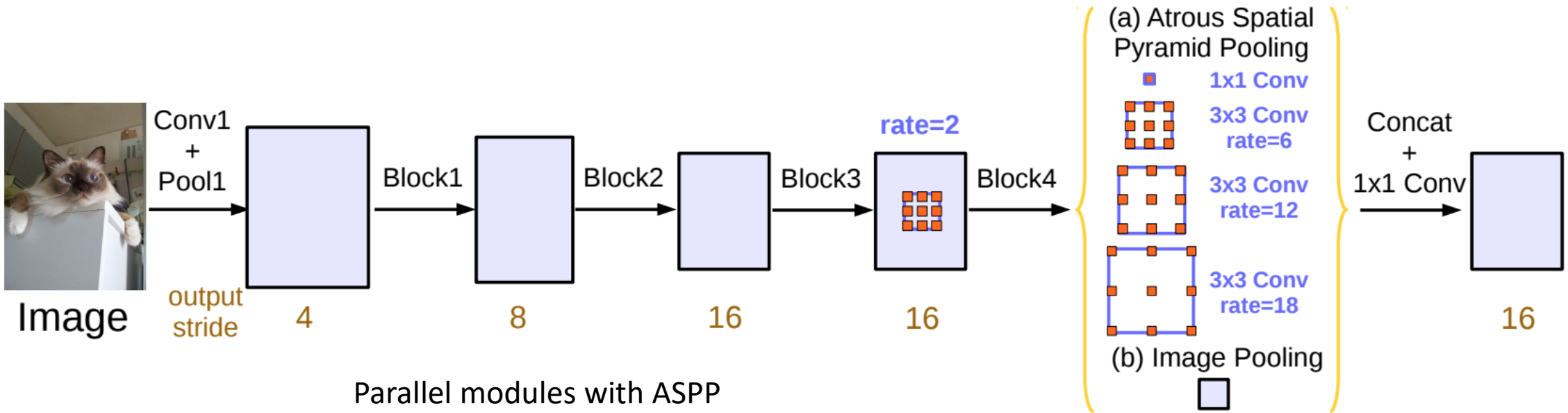


- It is based on Faster R-CNN objection detection model.
- Faster R-CNN detects objects (classification) in an image and finds bounding box of objects (regression).

- Extending Faster R-CNN, Mask R-CNN predicts a mask for each region of interest. So besides object class and bounding box location, Mask R-CNN also predicts the pixels of each object detected.

DeepLab

- It presents an architecture for controlling signal decimation and learning multi-scale contextual features.
- ResNet block uses atrous convolutions, uses different dilation rates to capture multi-scale context.
- On top of this new block, it uses Atrous Spatial Pyramid Pooling (ASPP). ASPP uses dilated convolutions with different rates as an attempt of classifying regions of an arbitrary scale.



Parallel modules with ASPP

(https://arxiv.org/abs/1706.05587)

DeepLab	Mask R-CNN	3D Conv-LSTM
Multi-class Semantic Segmentation	Originally designed for instance segmentation (Class-independent). However it can be used for Multi-class semantic Segmentaiton. Note: Mask is created for FG objects, and then class for the each FG region is determined from the object detection branch.	Binary-class semantic segmentation.
ImageNet pre-trained ResNet as its main feature extractor network.	Region proposal CNN is the backbone.	U-net style architecture.
High complex	High complex	Complex
Atrous Conv	Regular Conv	3D Conv + LSTM2D
DeepLab 3 (xception) 41M parameters		0.2M parameters

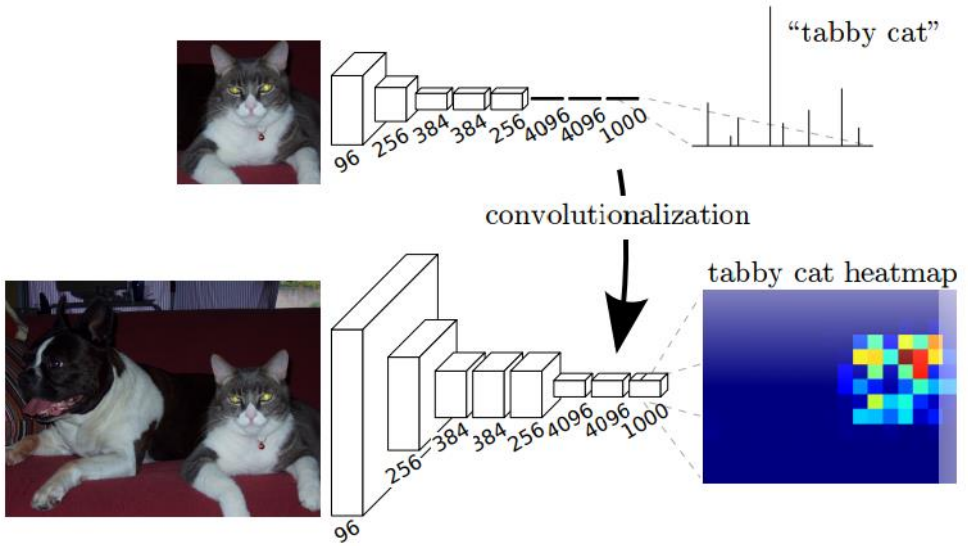
Deep learning for semantic segmentation

Semantic segmentation after 2014

- **Objective**: Label each pixel of an image by the object class that it belongs to, such as vehicle, human, sheep, and grass.
- **Breakthrough**: Major breakthrough came when **fully convolutional neural networks (FCNs)*** were first used by 2014 Long et. al.

(*A renewed paper from the same authors came in TPAMI, 2017 April)

FCN (2014) by Long et al. from Berkeley



- **Extensions:**
  - **U-Net**, SegNet, RefineNet, PSPNet, Mask-RCNN, and
  - Semi-supervised approaches like DecoupledNet[2015] and SS-GAN[2016]

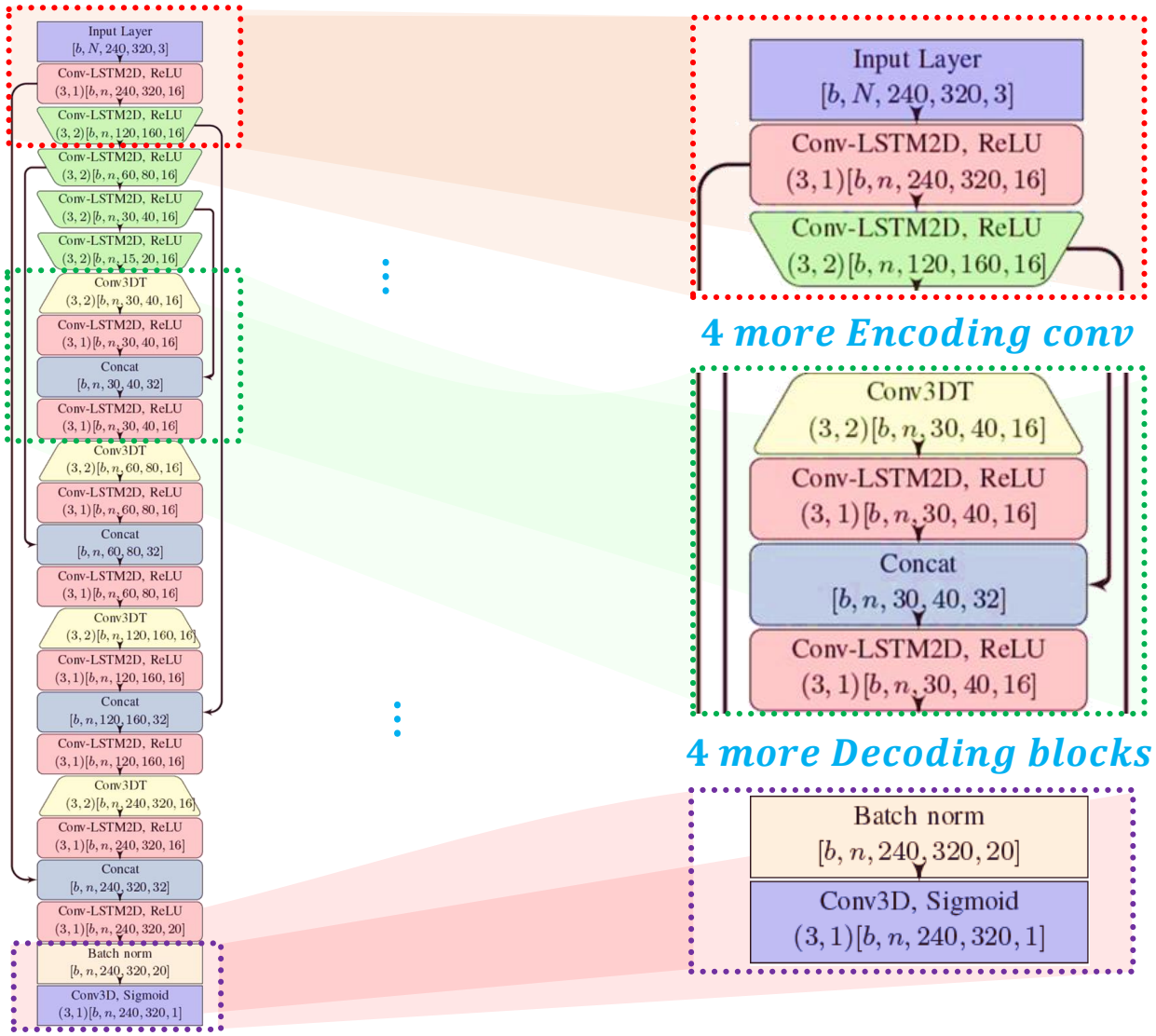
Base of our Model: A Vanilla Conv-LSTM2D model

- There is no direct architecture as the Vanilla model we used in the paper. However, it is inspired by the famous U-net [4] introduced by Ronneberger et al. for MRI segmentation.

- A straightforward option to handle the spatio-temporal features is using conv-LSTM2D units.

- So the Vanilla model uses conv-LSTM2D units instead of standard Conv units.

Vanila model to 3D Conv-LSTM2D model



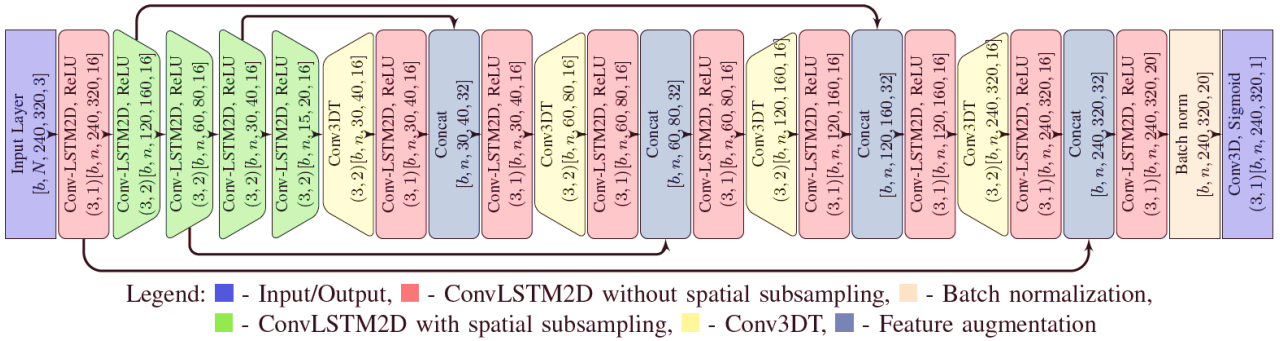
- The size of the kernel k, stride rate s, and dimension of the output are denoted in the following order and enclosing braces $(k, s)[b, n, H, W, D]$ on each layer. Where, b, n, H, W, and D represents the size, number of samples taken by the Conv-LSTM modules to capture the temporal information, hight and width of the frame, and number of output feature maps, respectively.

Legend

- Input/Output,
- ConvLSTM2D w/t subsampling
- ConvLSTM2D w/o subsampling
- Batch normalization,
- Conv3DT, - Feature augmentation

A layer-wise schematic of the Vanilla model inspired by the the U-net [4]. It exploits Conv-LSTM2D modules in an encoder-decoder fashion with residual feature concatenations.

Base of our Model:
A Vanilla Conv-LSTM2D model



- It is inspired by the U-net [4] architecture, but instead of the standard Convolutional and Transpose Convolutional layers; it employs Conv-LSTM2D and Conv3DT layers. We take advantage of a bottom-up implementation strategy. As we want to handle the sequence learning using LSTM modules, we start from a scratch model by modifying the U-net as shown above.
- Besides, our model requires a 5D input data (for time-domain: [bath size, time steps, hight, width, number of channels]), while the U-net takes 4D data ([bath size, height, width, number of channels]) without a consideration of temporal information as it originally targeted for biomedical image segmentation.

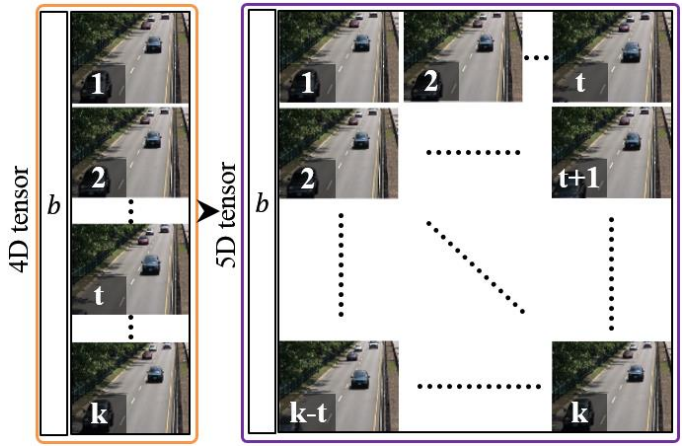


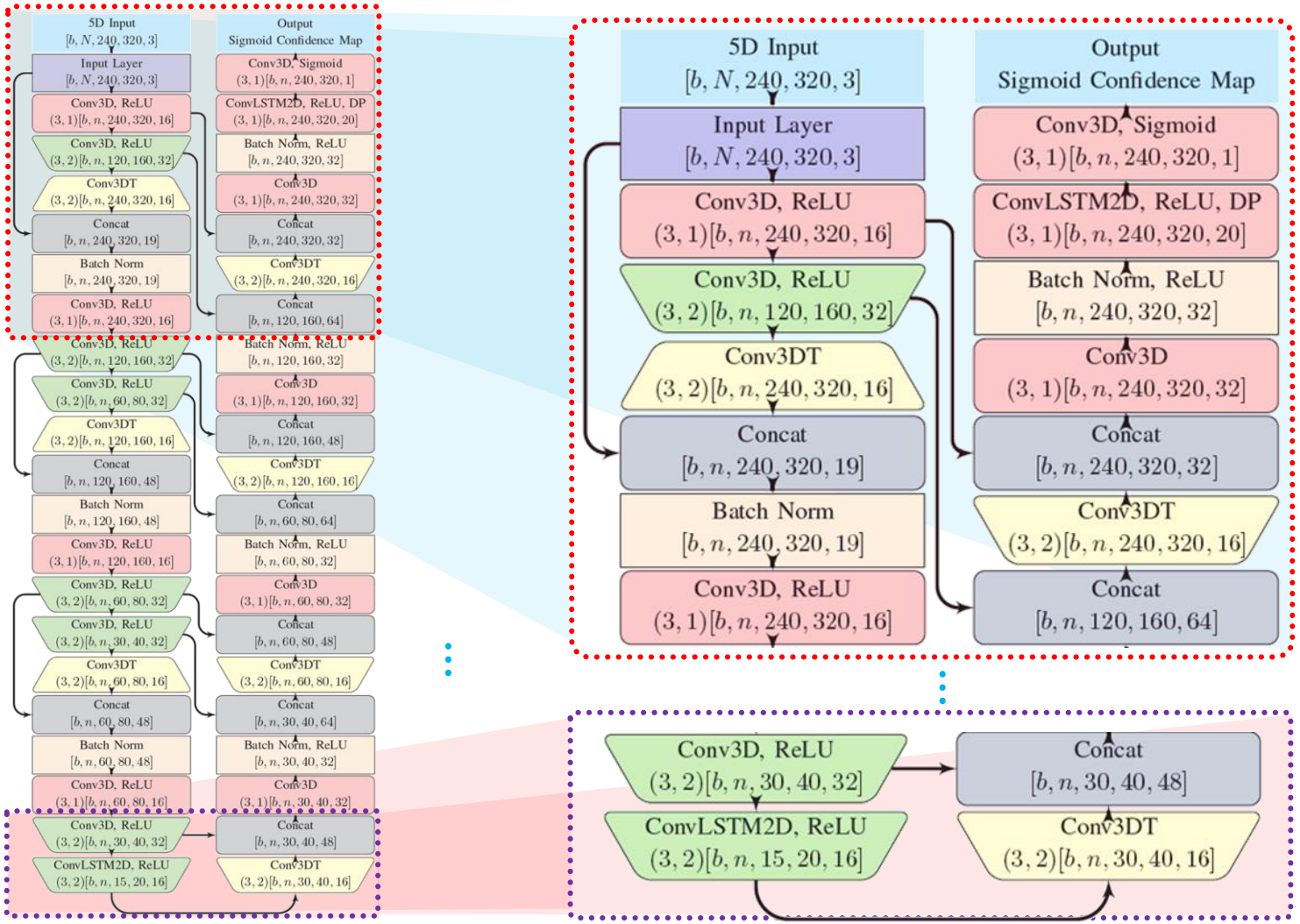
Fig. 7: Sequence generation for the 3D CNN-LSTM model: $4D \in \mathbb{R}^{b \times H \times W \times D}$, $5D \in \mathbb{R}^{b \times t \times H \times W \times D}$, where b, H, W, D, and t stand for input batch size, hight, width, number of channels and number of look back frames respectively.

- Then, we improve the scratch to the proposed 3D CNN-LSTM model through trial-and-error and empirical analysis.
- We call the first model as Vanilla structure since it sequentially reduces the spatial dimension without any feature-level augmentation in the encoding phase and does the up-sampling process with residual feature flows similar to the U-net and many of the segmentation DNNs in the literature, like DeepBS [5] and DBFCN [6].

Vanila model to 3D Conv-LSTM2D model

- From the Vanilla model, we retain the skeleton of the model, which is essentially an encoder-decoder CNN with the spatiotemporal learning system and residual feature flows from encoder to decoder. Then, we introduce the following changes to the model.
  i. It introduces a novel technique named **double-encoding using autoencoder-like micromodules and slow-decoding using feature passing residual connections**. Here, an input feature at a stage during the down-sampling process is encoded twice before it reaches completely to the next level of the dimension-reduced feature map. While the up-sampling process decodes the feature maps with two sets of residual feature flows from down-sampling stages for every new spatial dimension of the feature space.
  ii. In the base model, the time-dependent video cues are handled by Conv-LSTM2D layers stacked sequentially. Such, sequential LSTM modules consume a total of 298,529 trainable parameters and higher computational time. Besides, they handle the long-short term spatiotemporal cues at a global level. The improved model, however, overcomes these shortcomings by using 3D convolutions to capture the short spatiotemporal motions in local level while the long-short spatiotemporal motions are captured by LSTM modules in the down-sampling and up-sampling stages. In this way, the heavy computation of stacked Conv-LSTM2D layers is alleviated with better feature learning and less trainable parameters of 221,367

Vanila model to 3D Conv-LSTM2D model



Legend

- Input/Output,
- Conv3DT,
- Input layer,
- Conv3D w/o subsampling
- Conv3D w/t subsampling
- Batch normalization,
- Feature augmentation

A layer-wise schematic of the 3D CNN-LSTM model. It exploits autoencoder-like micro modules and slow-decoding strategy.

Vanila model to 3D Conv-LSTM2D model

- How the 3D CNN-LSTM lessens the number of trainable parameters?

- It is noticed that although the 3D CNN-LSTM model is deeper than the Vanilla model, it consumes fewer trainable parameters (221,367) than the ConvLSTM2D (298,529). It is understandable since a ConvLSTM2D layer has more nodes than a Conv3D layer.

Case	Layer type	Output shape	# Param
A	InputLayer	$b \times 240 \times 320 \times 3$	0
	ConvLSTM2D	$b \times 240 \times 320 \times 16$	11,008
B	InputLayer	$b \times 240 \times 320 \times 3$	0
	Conv3D	$b \times 240 \times 320 \times 16$	1,312

TABLE I: Trainable parameter comparison: ConvLSTM2D vs. Conv3D based on Keras 2.1.5-tf API.

- For instance, a ConvLSTM2D with a kernel size of 3 and 16 output feature maps for a four-frame look back input sequence with a dimension of 240×320×3 subsumes 11,008 trainable parameters while a Conv3D has only 1,312 trainable parameters for the same setting as shown in Table 1.

- So replacing a ConvLSTM2D with a Conv3D, we can achieve ~ 84% reduction in the trainable parameters that allow us to design a deeper architecture. Thus, our 3D CNN-LSTM architecture becomes 1.75 times deeper with ~ 25% lesser parameters than the Vanilla model.

Vanila model to 3D Conv-LSTM2D model

- Proof of Concept

- The initial performance evaluation is made between the base model (Vanilla) and the proposed 3D CNN-LSTM is to analyze how well our ideas pay off in terms of FG detection and processing time. In other words, it is a sanity test and proof of concept (PoC).

- Then, based on the empirical manifest we finalize the model and carry out extensive experiments on a set of sixteen benchmark video sequences from the change detection 2014 [7] (CDnet).

Dataset	Model	G-th	O-th	Average
Highway	Vanilla	95.20	95.58	95.39
	3D CNN-LSTM	97.71	97.69	97.70
Office	Vanilla	95.19	95.14	95.17
	3D CNN-LSTM	96.86	96.74	96.80
Canoe	Vanilla	93.83	88.23	91.03
	3D CNN-LSTM	95.39	95.34	95.37
Boats	Vanilla	90.88	91.14	91.01
	3D CNN-LSTM	92.60	92.47	92.54
Overpass	Vanilla	90.21	88.02	89.12
	3D CNN-LSTM	95.78	95.46	95.62
Boulevard	Vanilla	87.07	81.35	84.35
	3D CNN-LSTM	95.70	95.16	95.43
CopyMachine	Vanilla	95.53	94.58	95.06
	3D CNN-LSTM	96.83	96.67	96.75
Overall average	Vanilla	92.56	90.58	91.57
	3D CNN-LSTM	95.84	95.65	95.74

TABLE II: Sanity check results of the proposed 3D CNNLSTM compared to the Vanilla model in terms of FoM.

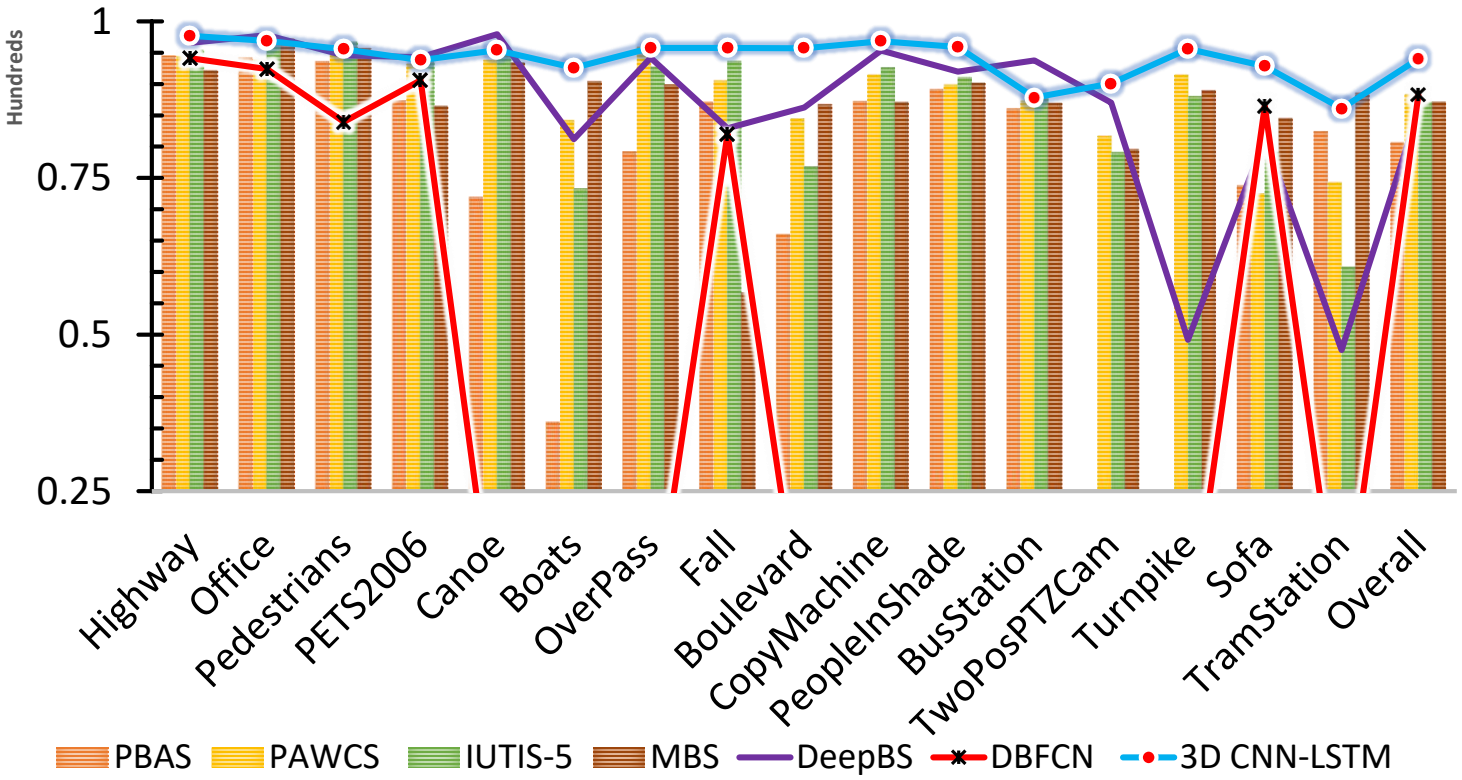
Quantitative analysis:

$$FoM = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \times 100\%$$

	PBAS [42]	PAWCS [13]	IUTIS-5 [58]	MBS [59]	DeepBS [31]	DBFCN [1]	3D CNN-LSTM
Highway	94.5	94.4	95.4	92.2	96.6	94.1	97.7
Office	94.2	93.7	96.9	97.2	97.8	92.4	96.7
Pedestrians	93.6	94.6	96.7	95.7	94.6	83.9	95.5
PETS2006	87.4	93.2	93.5	86.5	94.3	90.6	93.8
Canoe	72.0	93.8	94.6	93.5	97.9	na	95.4
Boats	36.1	84.2	73.3	90.4	81.2	na	92.6
OverPass	79.3	95.9	92.7	89.9	94.2	na	95.8
Fall	87.1	90.5	93.6	56.7	82.9	82.0	95.7
Boulevard	66.0	84.4	76.8	86.7	86.2	na	95.7
CopyMachine	87.3	91.4	92.6	87.1	95.3	na	96.8
PeopleInShade	89.2	89.9	91.0	90.2	92.0	na	95.9
BusStation	86.1	87.3	88.3	87.0	93.1	na	87.9
TwoPosPTZCam	na	81.7	79.1	79.6	87.0	na	90.1
Turnpike	na	91.5	88.0	89.0	49.2	na	95.6
Sofa	73.8	72.5	79.1	84.6	81.3	86.5	92.9
TramStation	82.4	74.3	60.8	88.6	47.5	na	86.1
Overall	80.6	88.3	87.0	87.2	85.7	88.3	94.0

TABLE III:Performance comparison in terms of FoM: Global-th and Otsu-th stand for the two methods applied to transform Sigmoid scores to binary mask. Values in red are the best while the ones in blue are the second best FoM (na - not available).

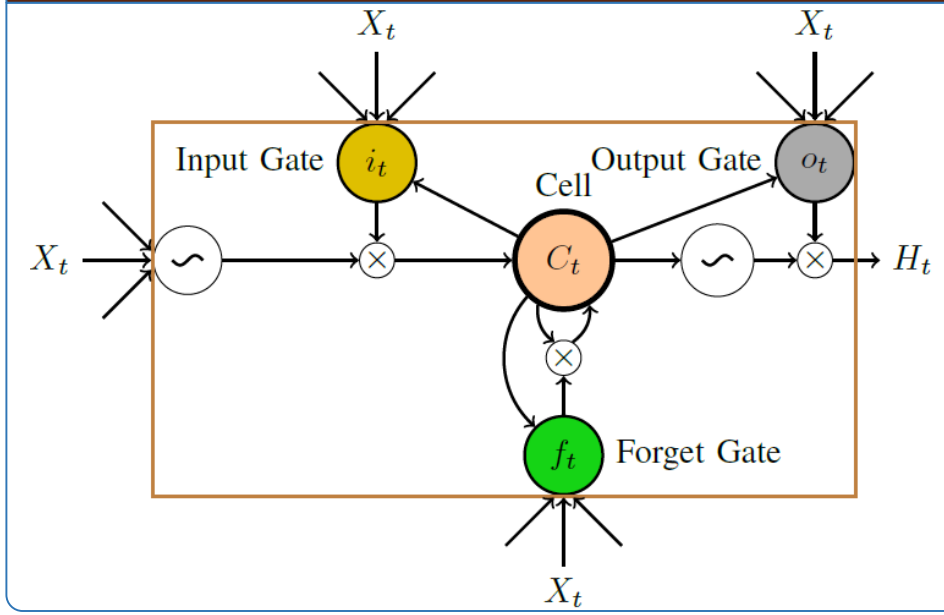
Quantitative analysis: FoM



- The prosed 3D CNN-LSTM model exhibits a robustness with average FoM of ~ 94%.
- The overall improvements of the proposed model are 14%, 6%, 7%, 7%, 8%, and 6% compared to PBAS, PAWCS, IUTIS-5, MBS, DeepBS, and DBFCN respectively.

Performance comparison in terms of FoM: PBAS (2012), PAWCS (2015), IUTIS-5 (2017), MBS (2017), DeepBS (2018), DBFCN (2018)

Long-short-term-memory (LSTM)



$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + b_i), \quad (5)$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f), \quad (6)$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + b_o), \quad (7)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c), \quad (8)$$

$$H_t = o_t \circ \tanh(C_t), \quad (9)$$

- where σ is the recurrent activator, $W_{x\_}$ and $W_{h\_}$ are the spatial dimension of conv kernels. In this work, σ is a hard sigmoid function.

Behavior

- It's useful to summarize various behaviors the memory cell can achieve depending on the values of the input and forget gates:

input gate	forget gate	behavior
0	1	remember the previous value
1	1	add to the previous value
0	0	erase the value
1	0	overwrite the value

Component description

- At the center is a **memory cell**, which is the thing that's able to remember information over time. It has a linear activation function, and a self-loop which is **modulated** by a **forget gate**, which takes values between 0 and 1; this means that the weight of the self-loop is equal to the value of the forget gate.
- The forget gate is a unit similar to the ones we've covered previously; it computes a linear function of its inputs, followed by a logistic activation function (which means its output is between 0 and 1). The forget gate would probably be better called a "remember gate", since if it is on (takes the value 1), the memory cell remembers its previous value, whereas if the forget gate is off, the cell forgets it.
- The block also receives inputs from other blocks in the network; these are summed together and passed through a tanh activation function (which squashes the values to be between -1 and 1). The connection from the input unit to the memory cell is gated by an **input gate**, which has the same functional form as the forget gate (i.e., linearthen-logistic).
- The block produces an output, which is the value of the memory cell, passed through a tanh activation function. It may or may not pass this on to the rest of the network; this connection is modulated by the **output gate**, which has the same form as the input and forget gates.

References

1. Vincent C. and Edward C., Practical Object Detection and Segmentation, http://cs231n.stanford.edu/slides/2018/cs231n_2018_ds06.pdf
2. Jonathan H., Image segmentation with Mask R-CNN.
3. Liang-Chieh C., Alexander H., George P., Florian S., Peng W., Hartwig A., MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features, 2018.
4. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Inter.Conf. Medical Image Computing and Comput.-Assisted Intervention, pp. 234–241, Springer, 2015
5. M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," Pattern Recognition, vol. 76, pp. 635 – 649, 2018.
6. L. Yang, J. Li, Y. Luo, Y. Zhao, H. Cheng, and J. Li, "Deep background modeling using fully convolutional network," IEEE Trans. Intelli. Transportation Sys., vol. 19, no. 1, pp. 254–262, 2018.
7. Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "Cdnet 2014: An expanded change detection benchmark dataset," in IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 393–400, June 2014.
8. F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 724–732, 2016.