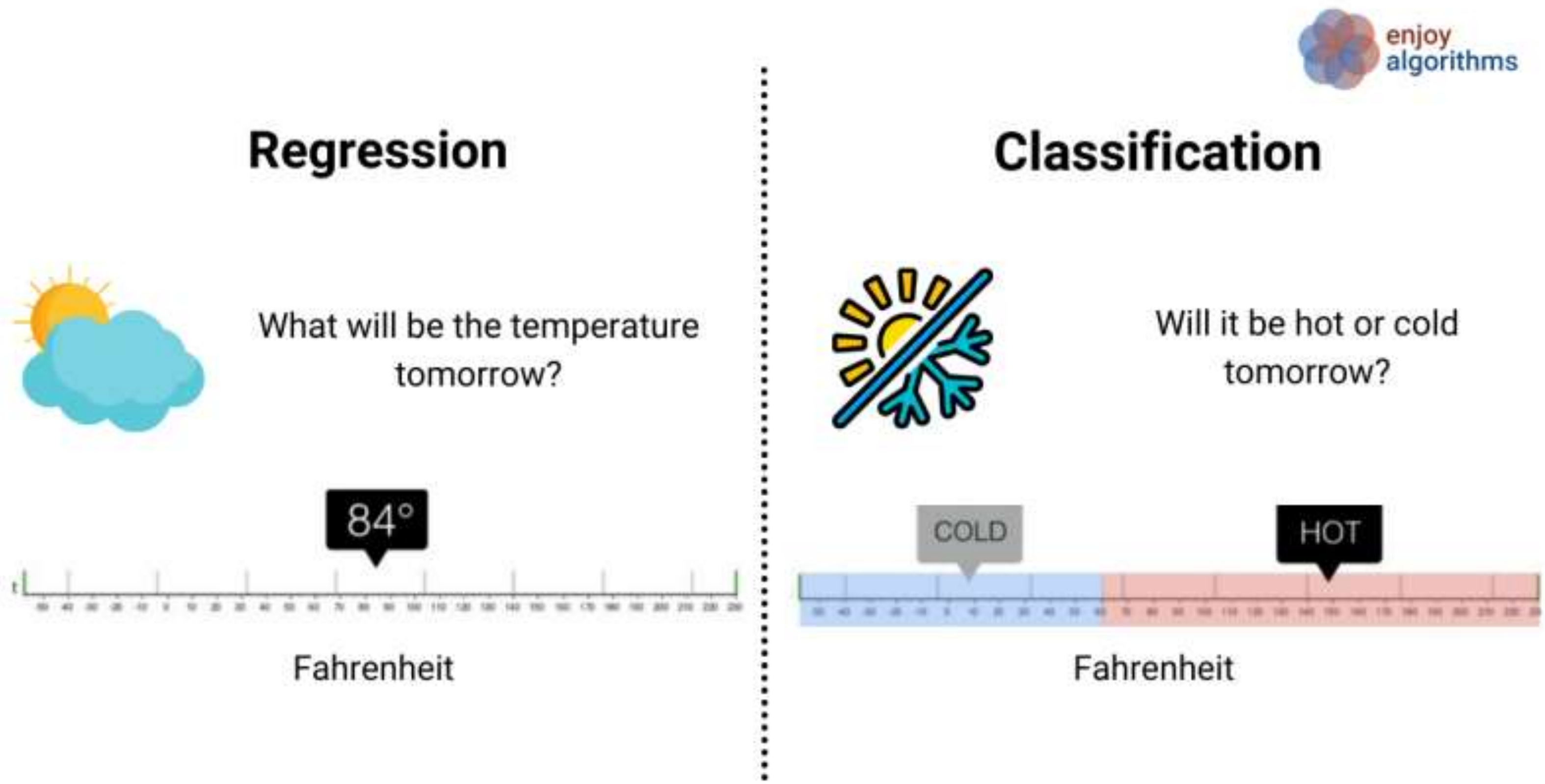
Discriminative models

Greg Tsagkatakis

CSD - UOC

ICS - FORTH

Types of problems

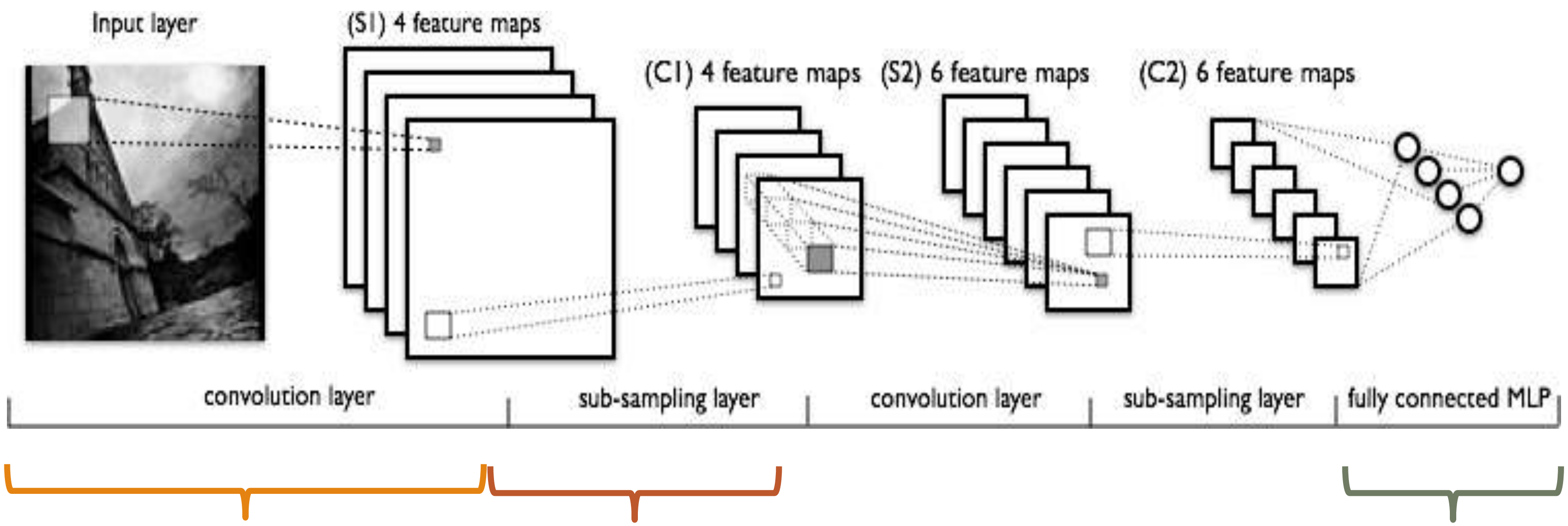


State-of-the-art (since 2015)

Deep Learning (DL)

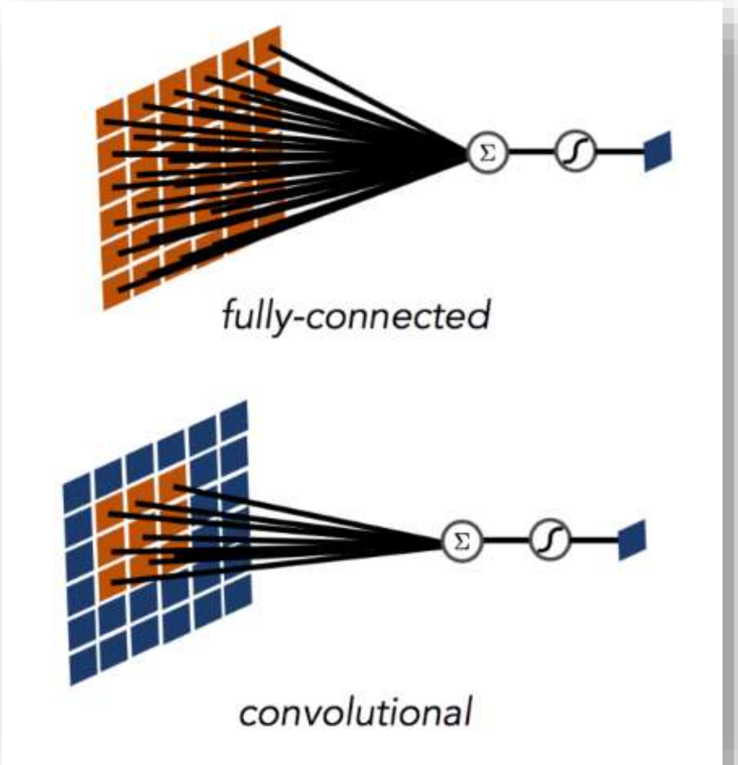
- Convolutional Neural Networks (CNN) <-> Images
- Recurrent Neural Networks (RNN) <-> Audio

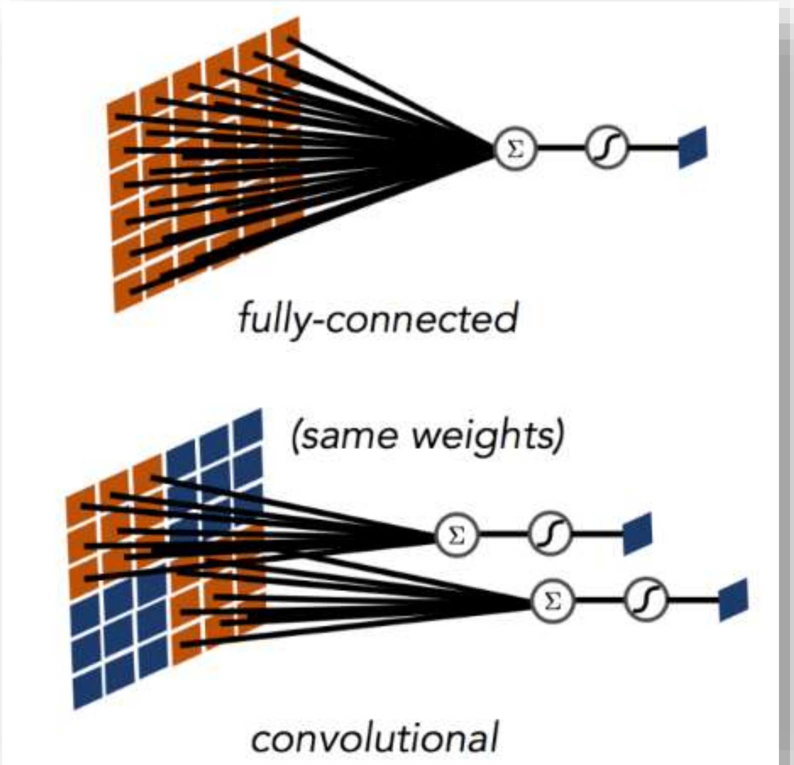
Convolutional Neural Networks



(Convolution + Subsampling) + () … + Fully Connected

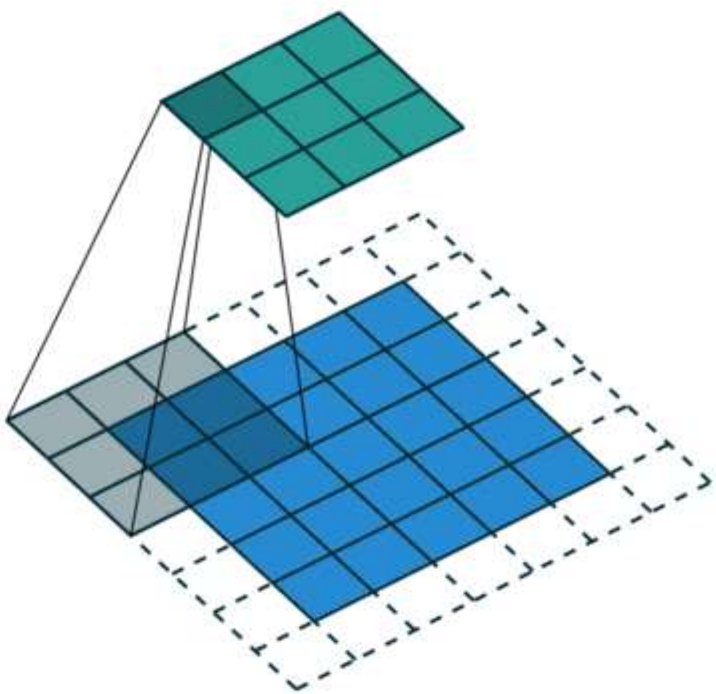
Convolutional Neural Networks

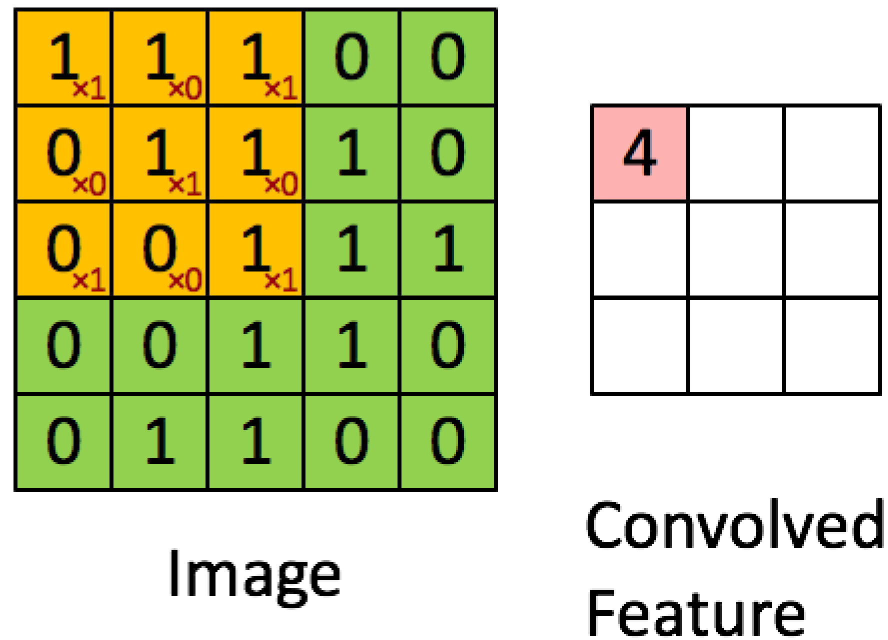




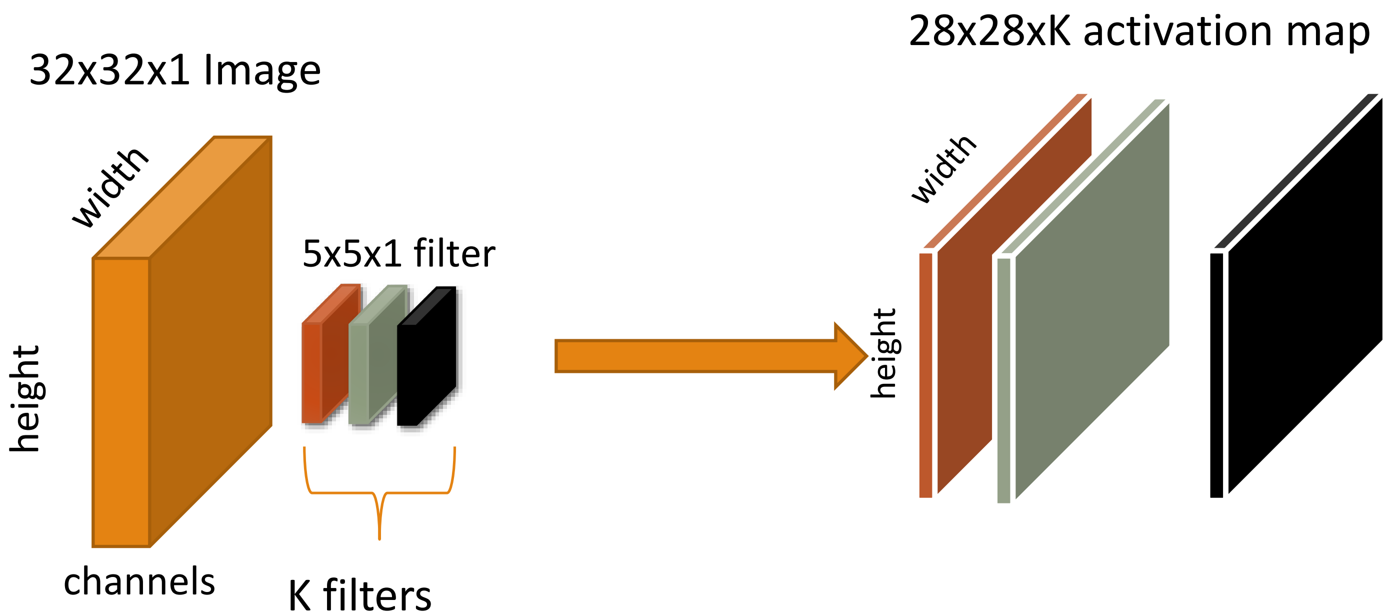
Tutorial: Introduction to convolutional neural networks (CNNs)
https://github.com/langnico/DL_tutorial_RS/tree/master

Convolution operator





Convolutional Layers



$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n) K(m,n)$$

$$= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n) K(-m,-n)$$

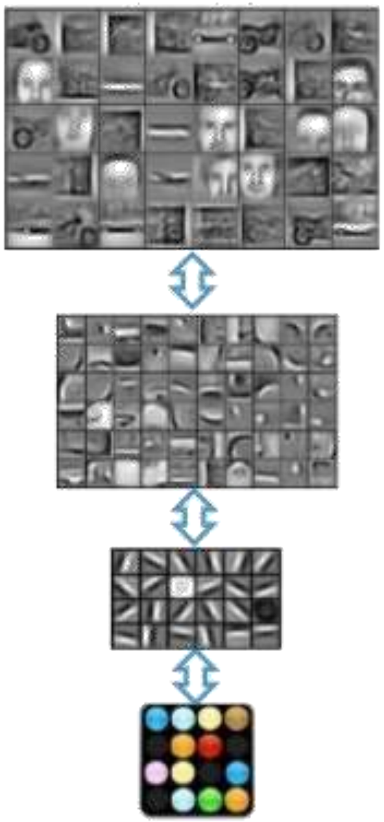
Convolutional Layers

Characteristics

- Hierarchical features
- Location invariance

Parameters

- Number of filters (32,64…)
- Filter size (3x3, 5x5)
- Stride (1)
- Padding (2,4)



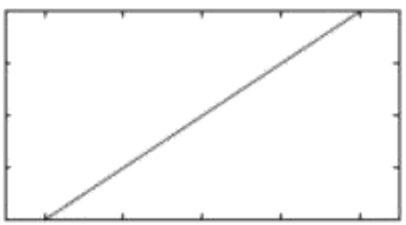
“Machine Learning and AI for Brain Simulations” – Andrew Ng Talk, UCLA, 2012

Activation Layer

Introduction of non-linearity

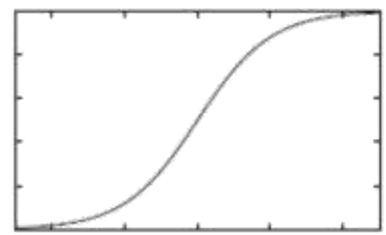
- Brain: thresholding -> spike trains

Identity (Linear)



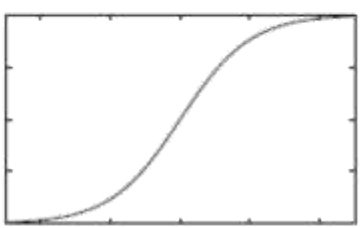
$$identity(x) = x$$

Sigmoid



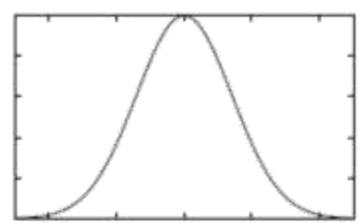
$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Tanh (Hypertangent)



$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Gaussian



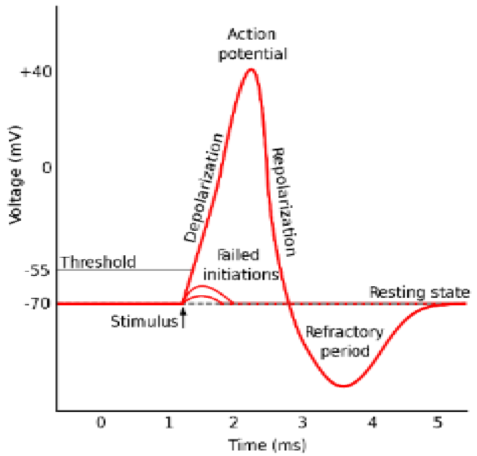
$$gaussian(x) = e^{-x^2/\sigma^2}$$

Activation Layer

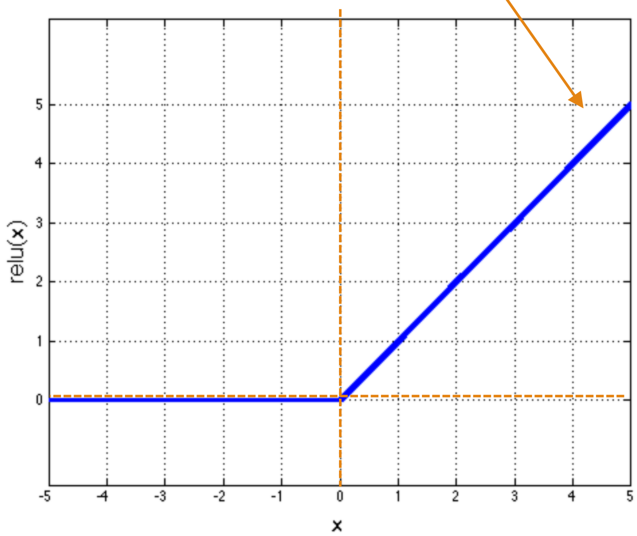
ReLU: x=max(0,x)

- ✓ Simplifies backprop
- ✓ Makes learning faster
- ✓ Avoids saturation issues
- ✓ ~ non-negativity constraint

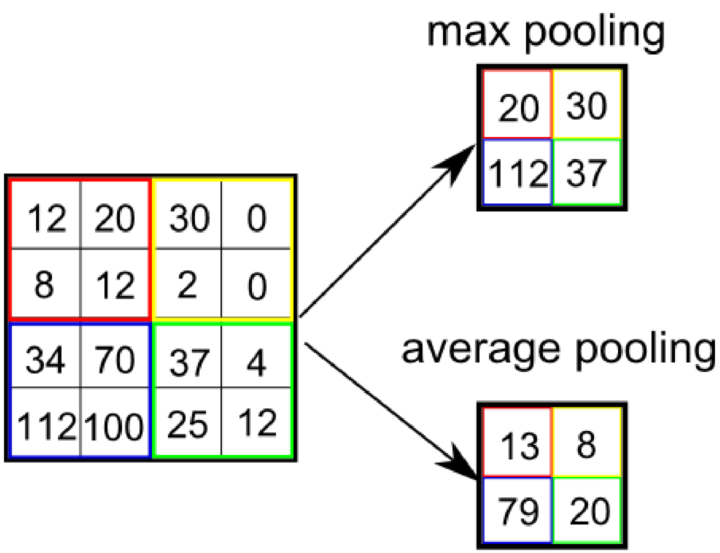
(Note: The brain)



No saturated gradients



Subsampling (pooling) Layers



<-> downsampling

➢ Scale invariance

Parameters

- Type
- Filter Size
- Stride

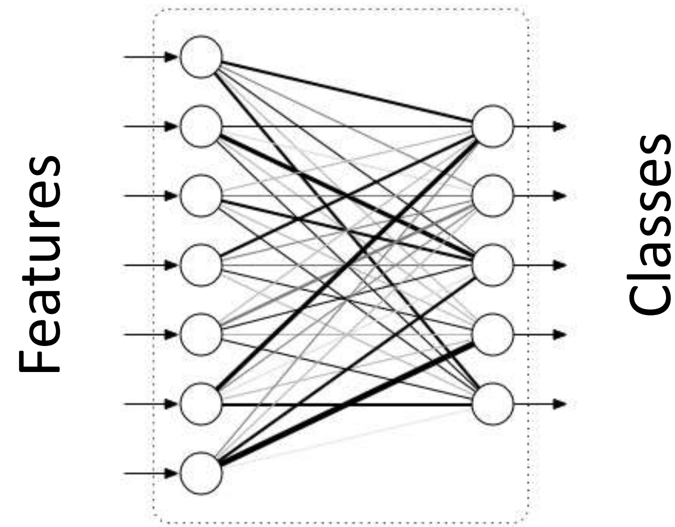
Fully Connected Layers

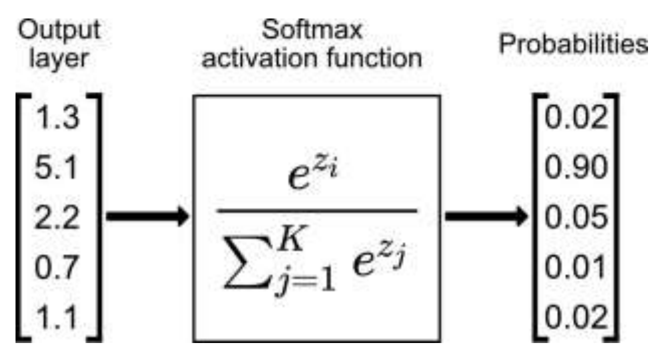
Full connections to all activations in the previous layer

Typically at the end

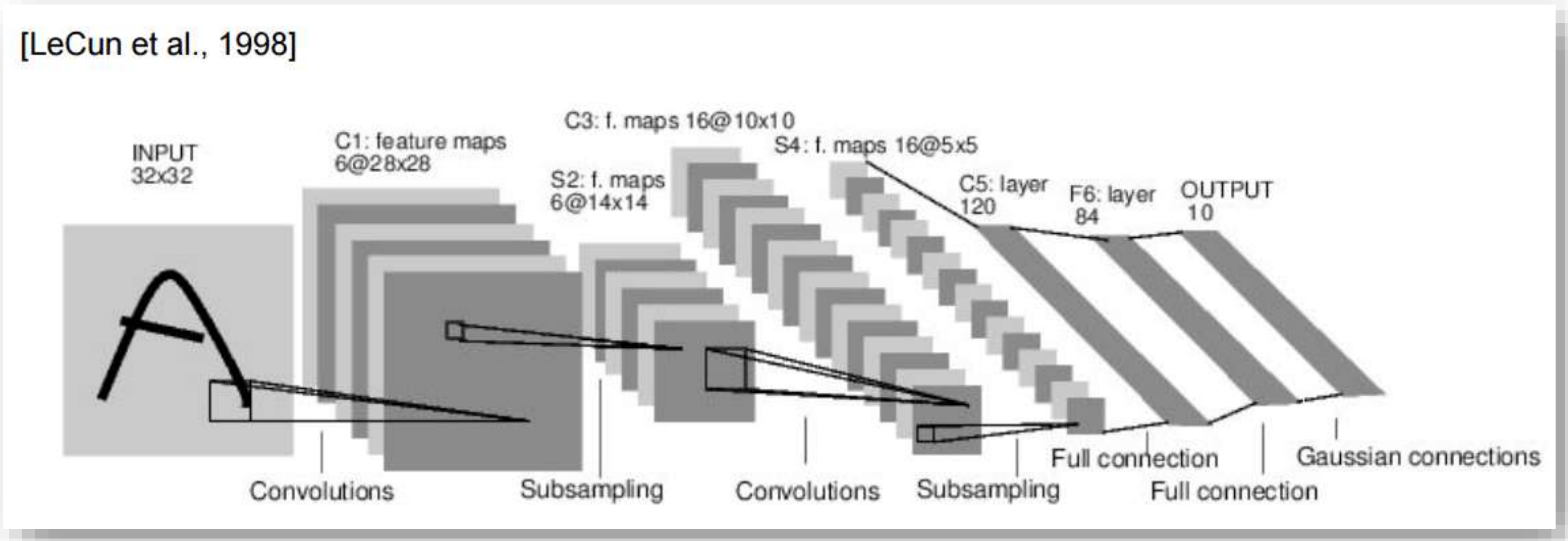
Can be replaced by conv

Softmax on logits

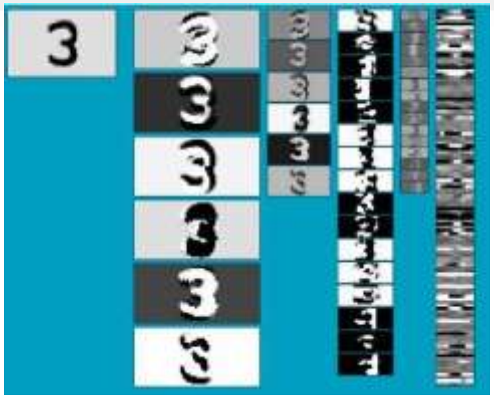




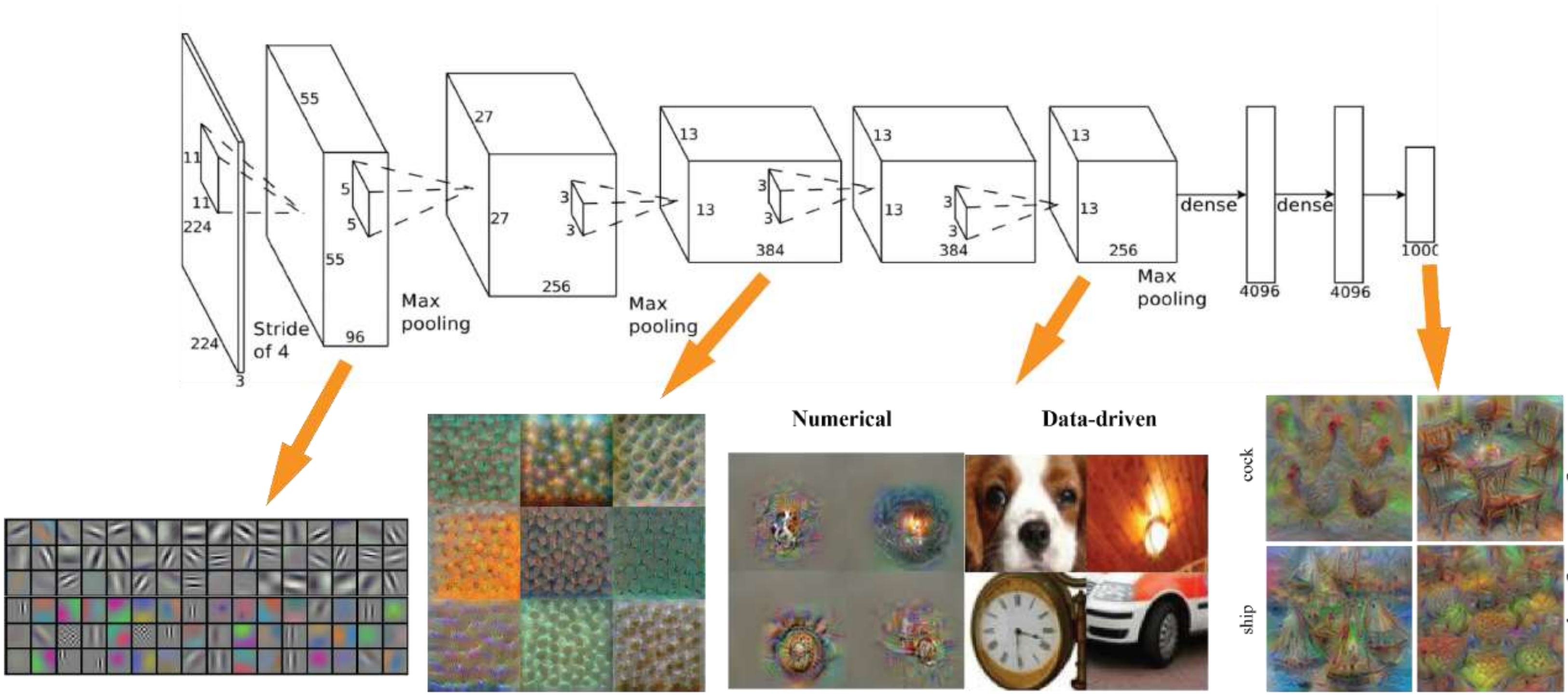
LeNet [1998]



[LeCun et al., 1998]
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections



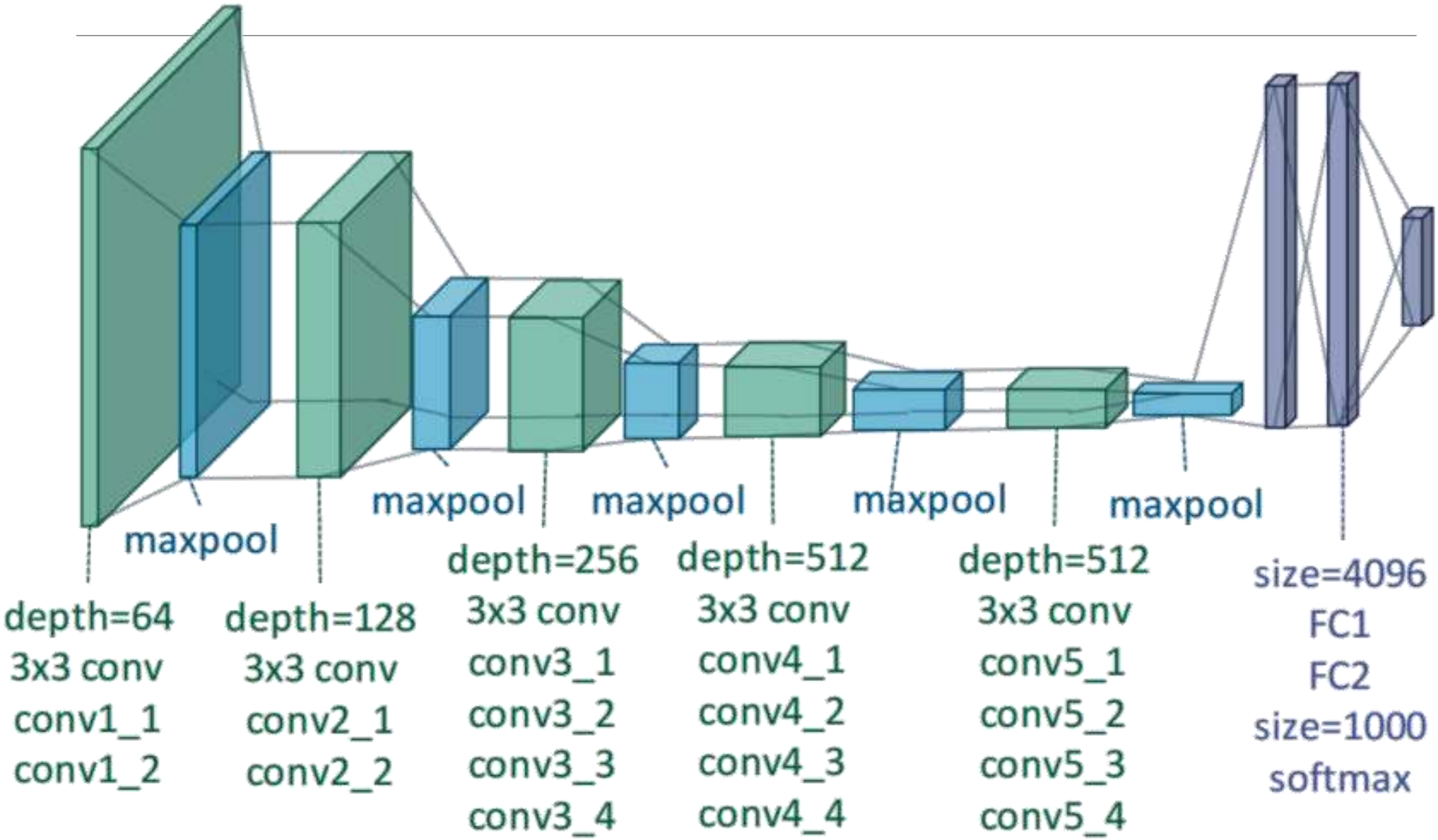
AlexNet [2012]



Conv 1: Edge+Blob Conv 3: Texture Conv 5: Object Parts Fc8: Object Classes

Alex Krizhevsky, Ilya Sutskever and Geoff Hinton, [ImageNet ILSVRC challenge]() in 2012
http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

VGGnet [2014]



K. Simonyan, A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv technical report, 2014

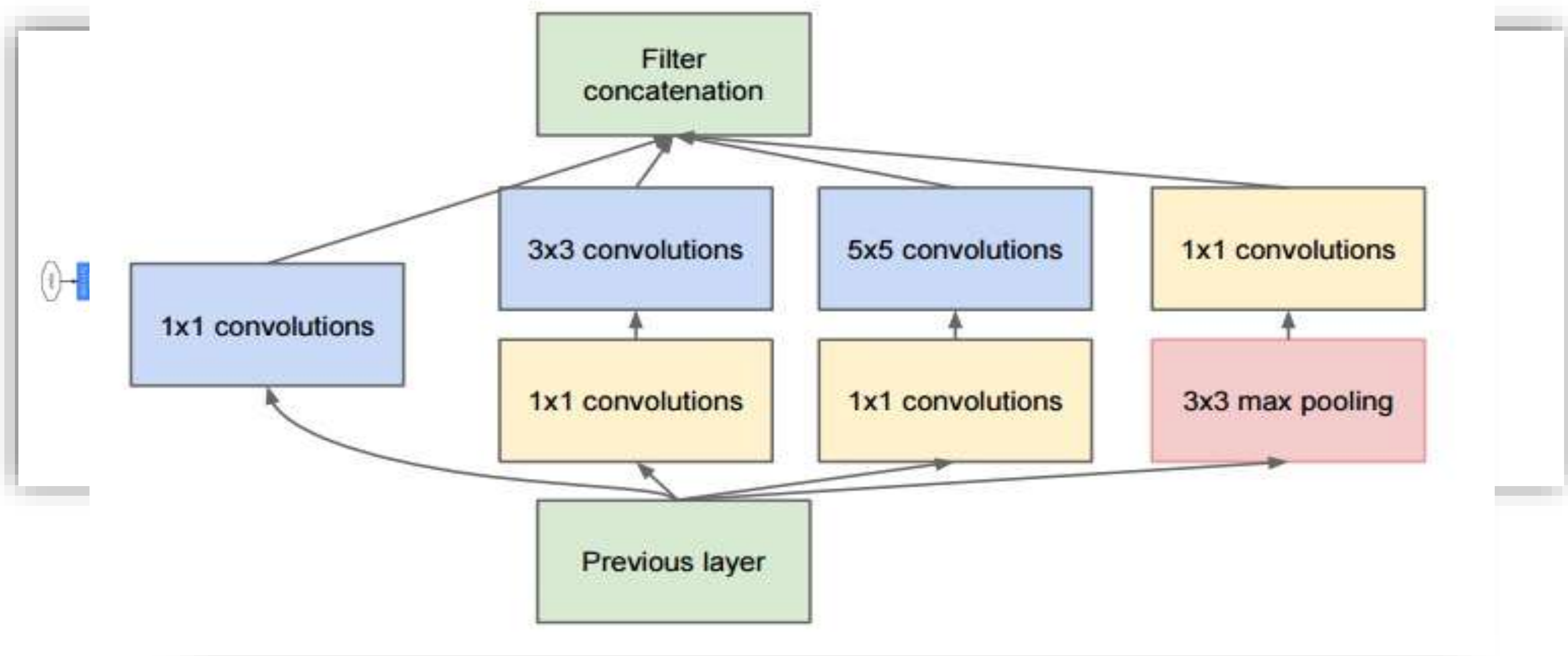
VGGnet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

D: VGG16
E: VGG19
All filters are 3x3

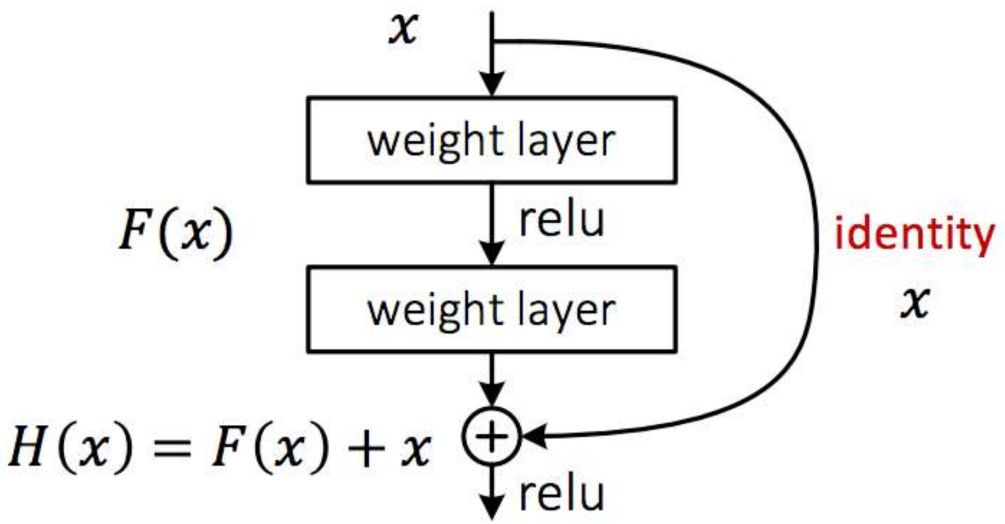
More layers
smaller filters

Inception (GoogLeNet, 2014)

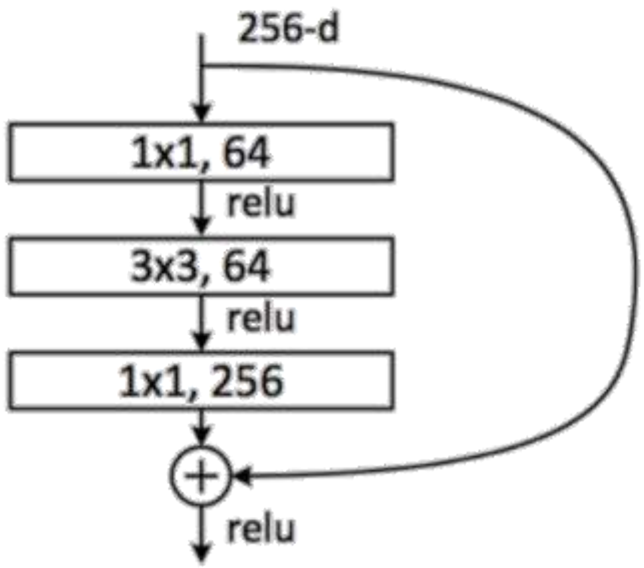


Inception module with dimensionality reduction

Residuals



x
weight layer
relu
weight layer
$F(x)$
identity
x
$H(x) = F(x) + x$
relu

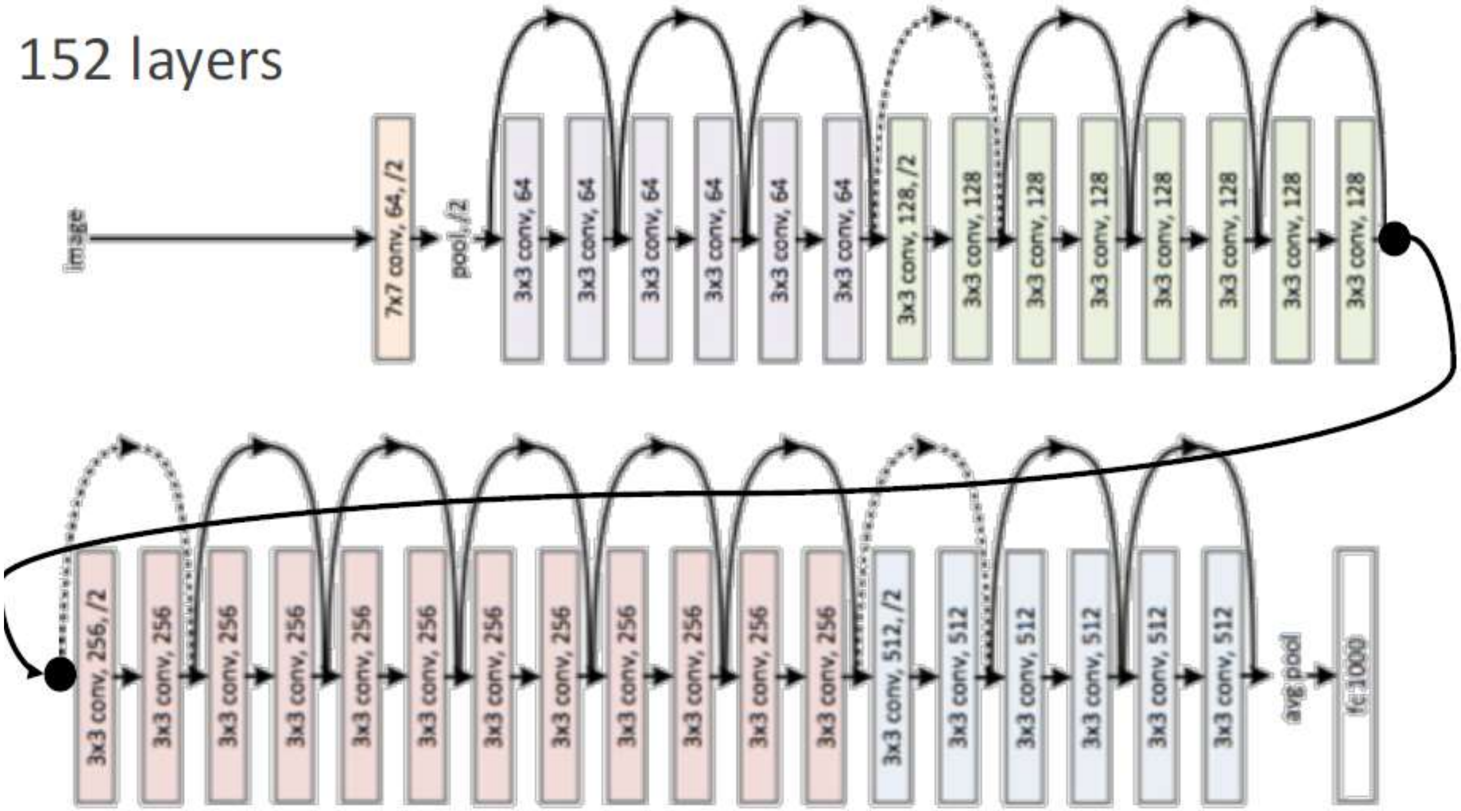


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

ResNet, 2015

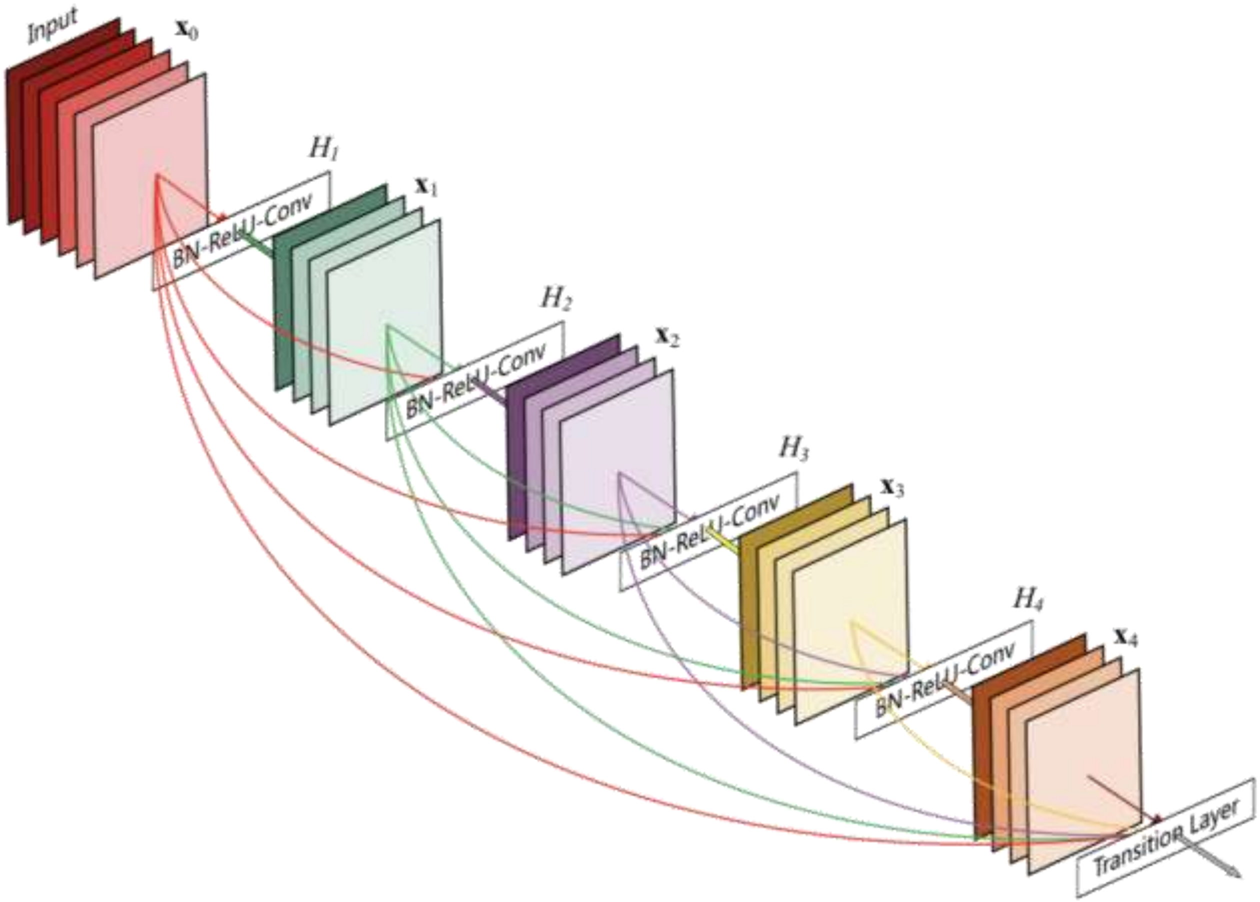
Residual Networks

152 layers



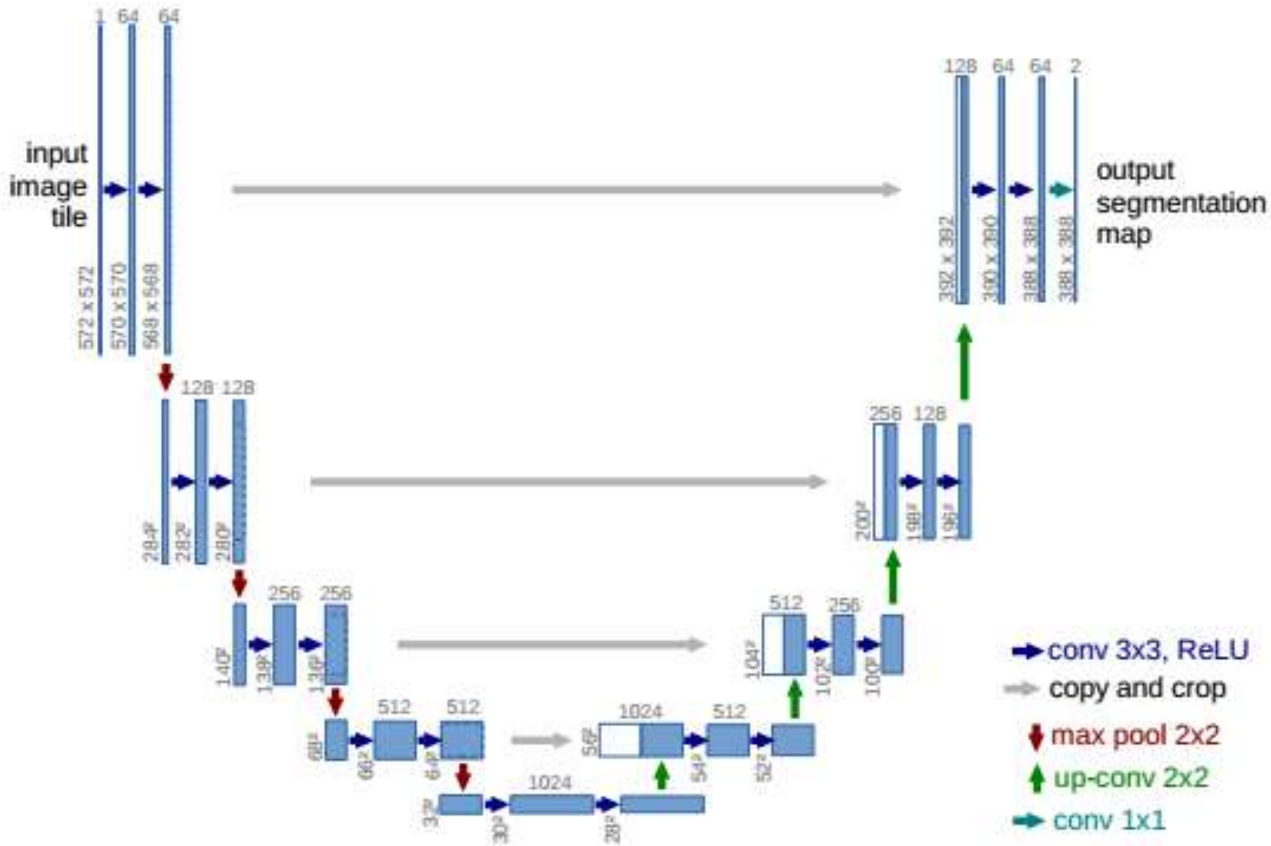
He, Kaiming, et al. "Deep residual learning for image recognition." *IEEE CVPR*. 2016.

DenseNet



Densely Connected Convolutional Networks, 2016

U-NET



Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015:*

Recurrent Neural Networks

Motivation

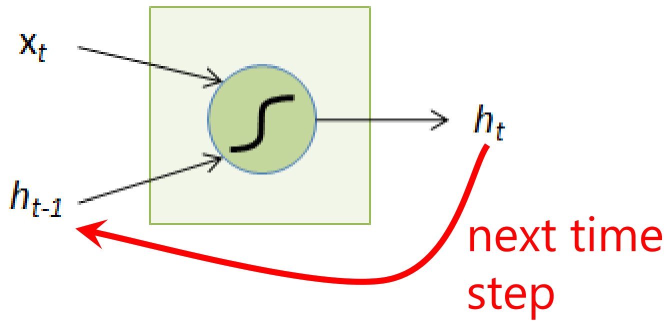
- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output
- fixed amount of computational steps
- recurrent nets allow us to operate over *sequences* of vectors

Use cases

- Video: sequence understanding
- Audio: speech transcription
- Text: natural language processing

Recurrent neuron

- x_t: Input at time t
- h_{t-1}: State at time t-1



$$h_t = f(W_h h_{t-1} + W_x x_t)$$

Recurrent Neural Networks

Feed-forward NN

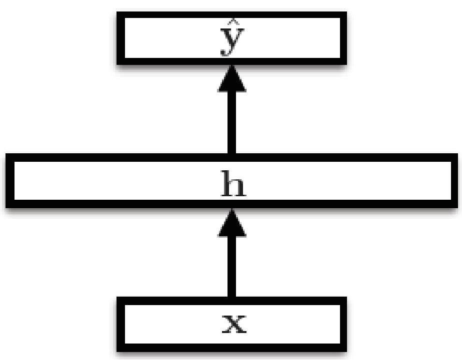
$$\mathbf{h} = g(\mathbf{V}\mathbf{x} + \mathbf{c})$$

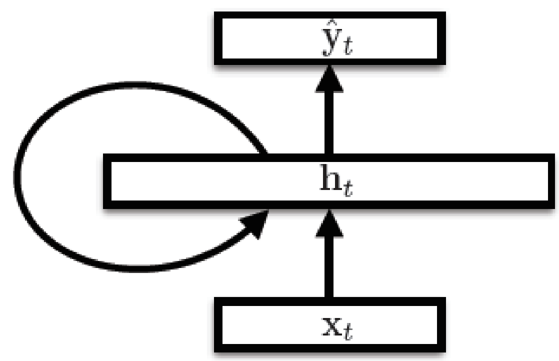
$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

Recurrent NN

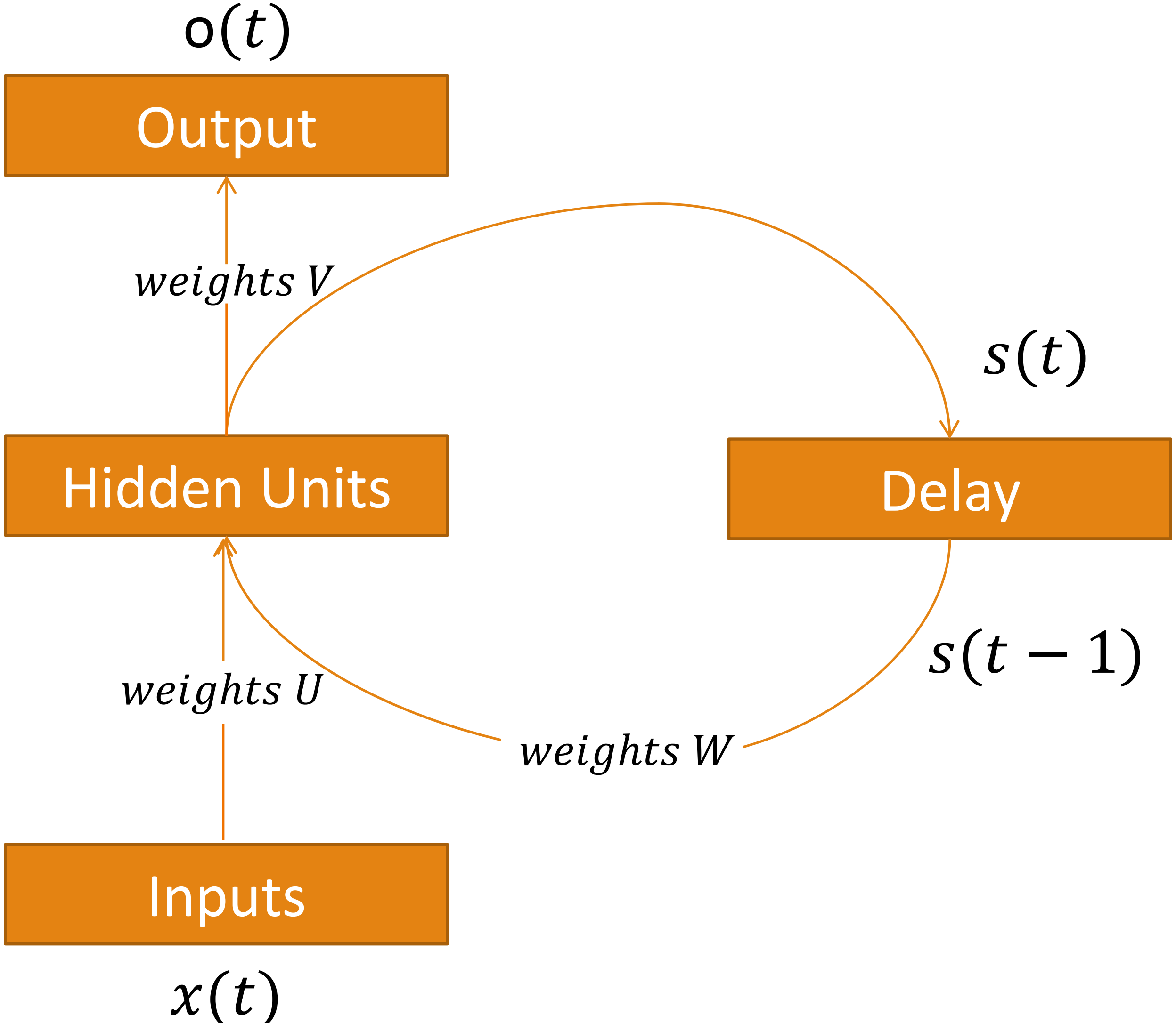
$$\mathbf{h}_t = g(\mathbf{V}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{c})$$

$$\hat{\mathbf{y}}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b}$$



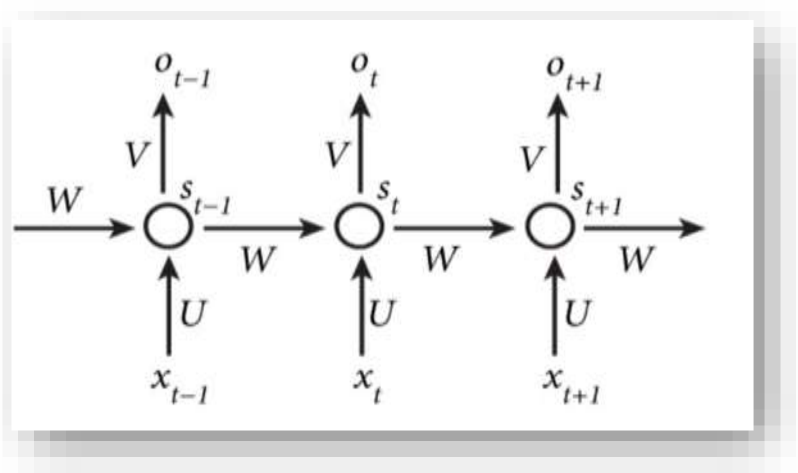


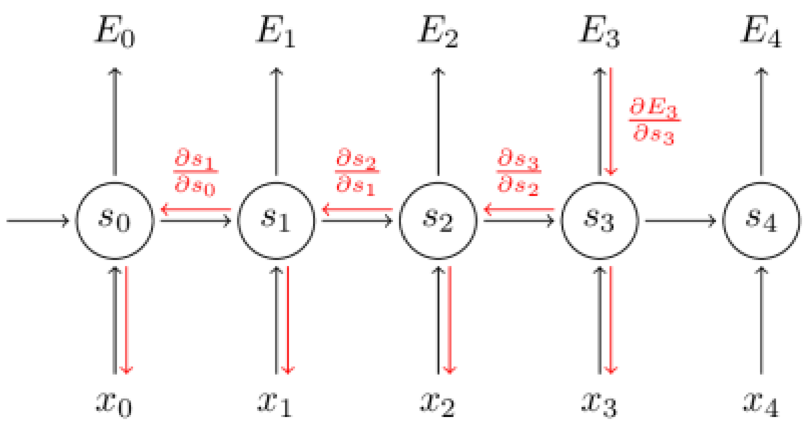
RNN Architecture



Unfolding RNNs

- Each node represents a layer of network units at a single time step.
- The same weights are reused at every time step.





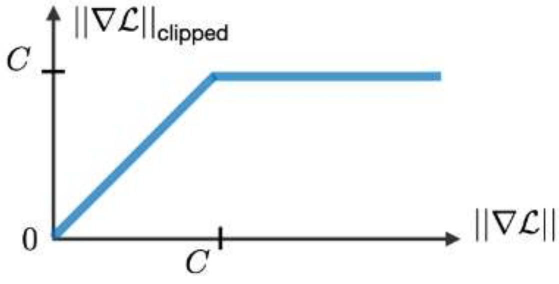
Training RNNs

Loss function: $\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$

Backpropagation through time $\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \left. \frac{\partial \mathcal{L}^{(T)}}{\partial W} \right|_{(t)}$

Vanishing/exploding gradient: failure to capture long-range dependencies

Gradient clipping



RNNs pros and cons

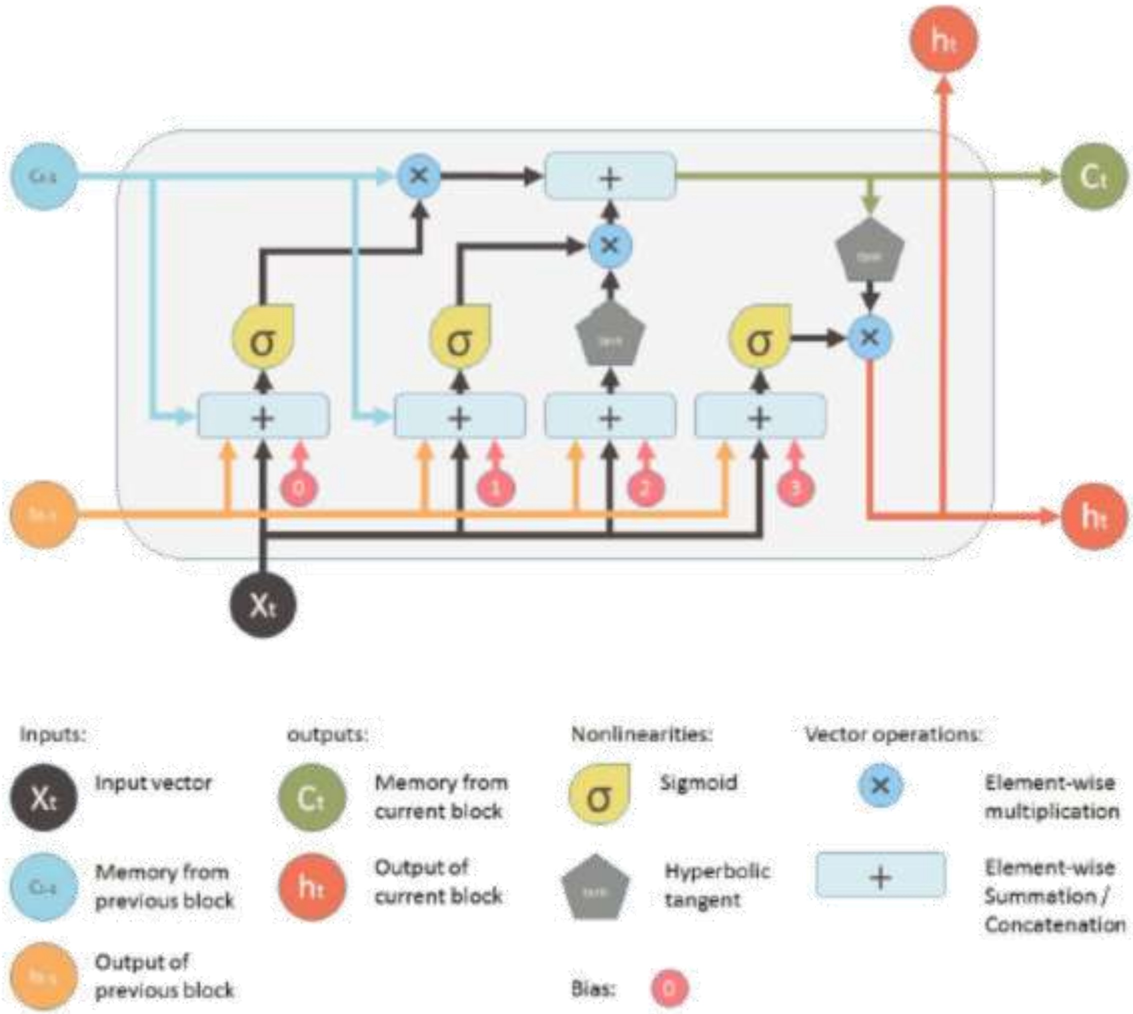
ADVANTAGES

• Possibility of processing input of any length
• Model size not increasing with size of input
• Computation takes into account historical information
• Weights are shared across time

DISADVANTAGES

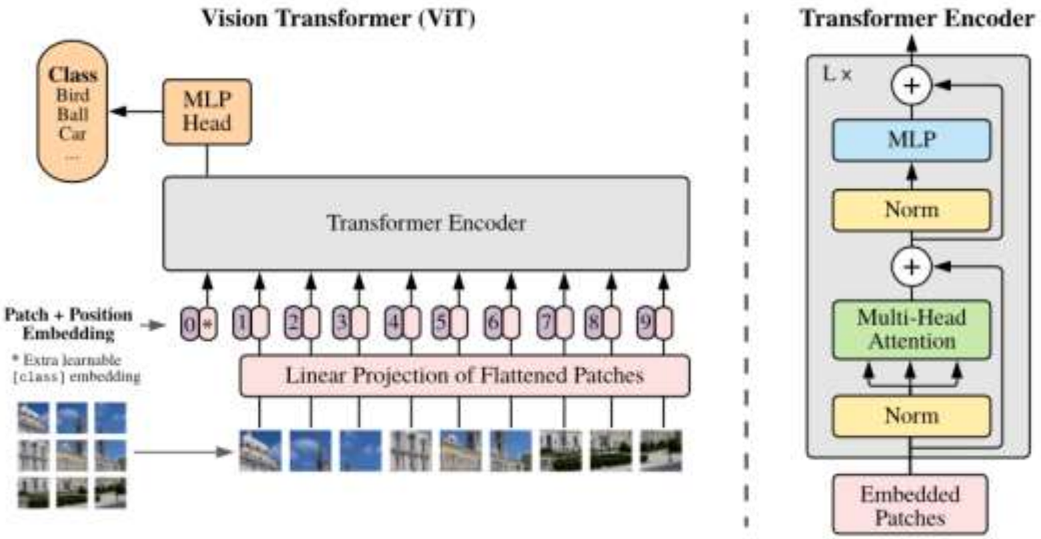
• Computation being slow
• Difficulty of accessing information from a long time ago
• Cannot consider any future input for the current state

Long Short-Term Memory Nets (LSTMs)



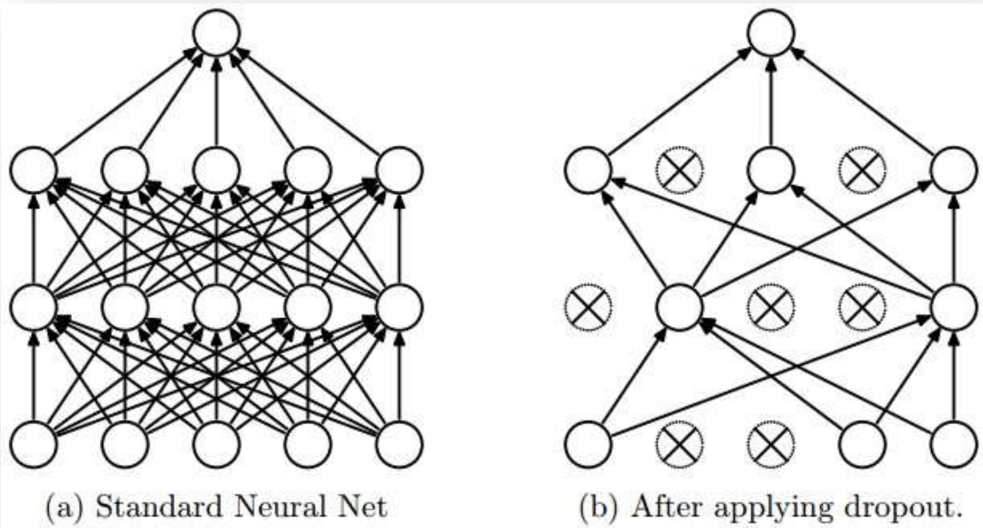
Vision Transformer (ViT)

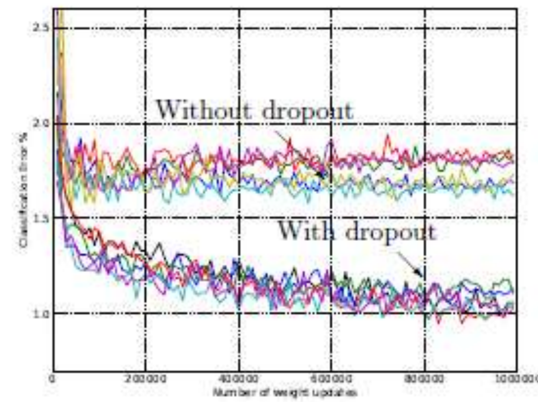
- Split an image into patches (fixed sizes)
- Flatten the image patches
- Create lower-dimensional linear embeddings from these flattened image patches
- Include positional embeddings
- Feed the sequence as an input to a state-of-the-art transformer encoder
- Pre-train the ViT model with image labels, which is then fully supervised on a big dataset
- Fine-tune the downstream dataset for image classification

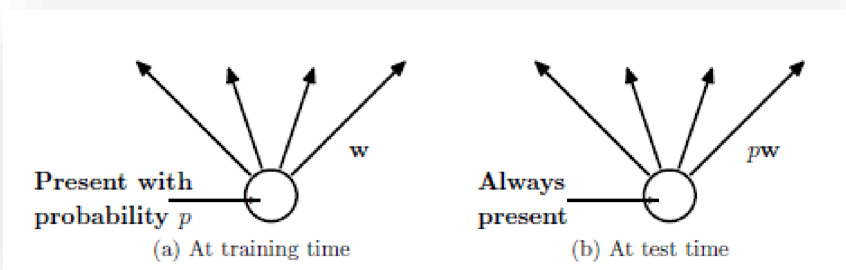


Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Dropout







Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958.

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

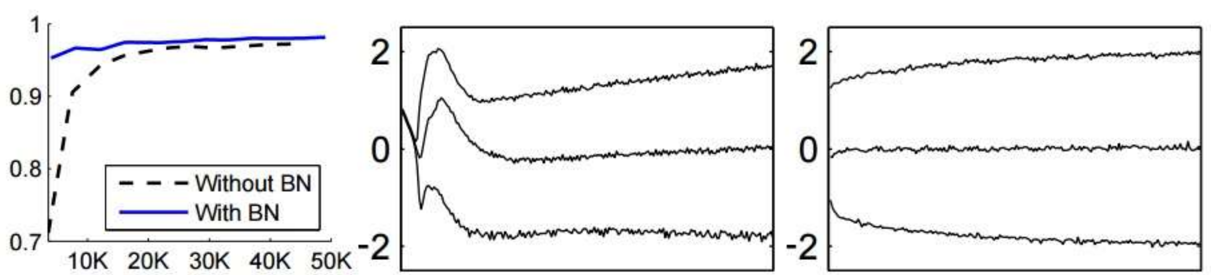
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

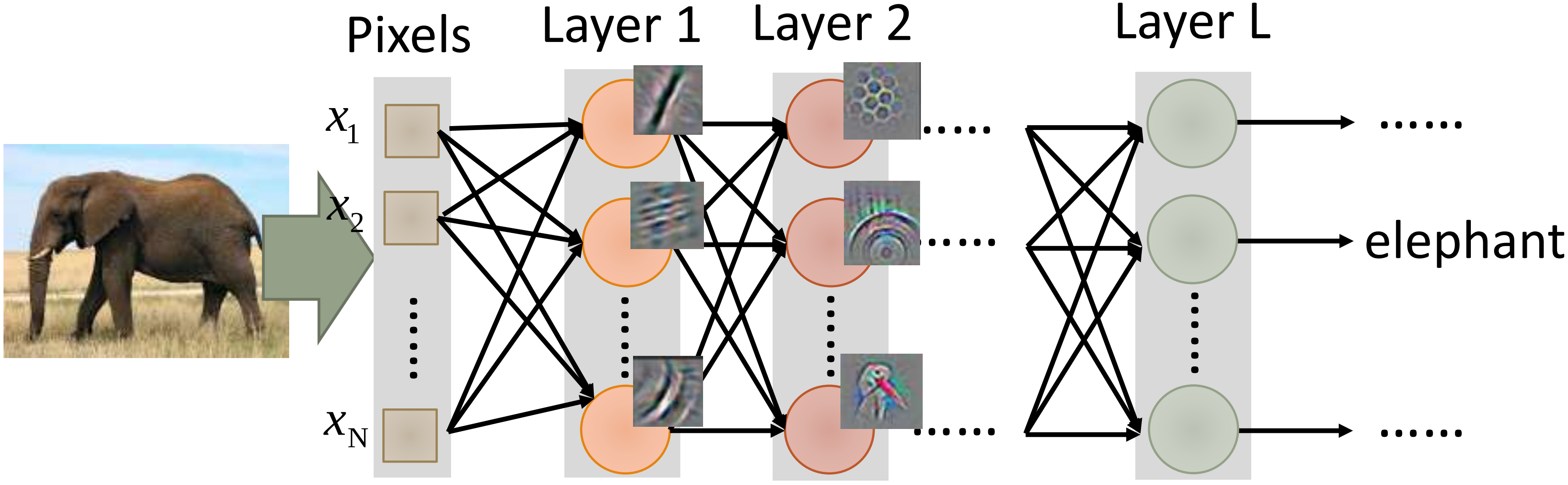
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



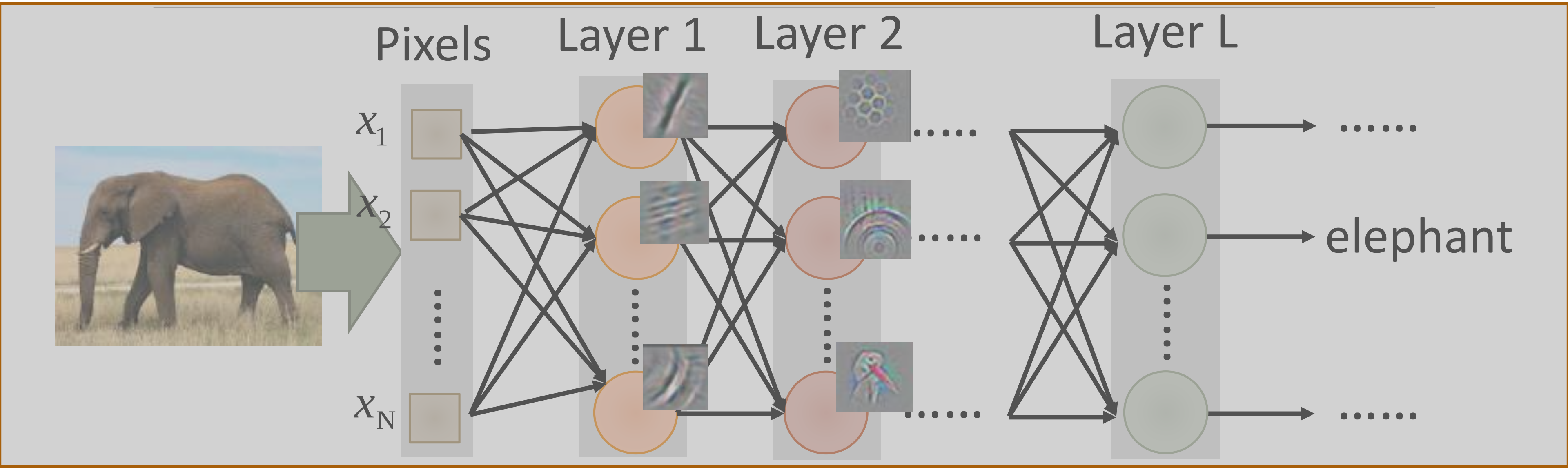
(a) (b) Without BN (c) With BN

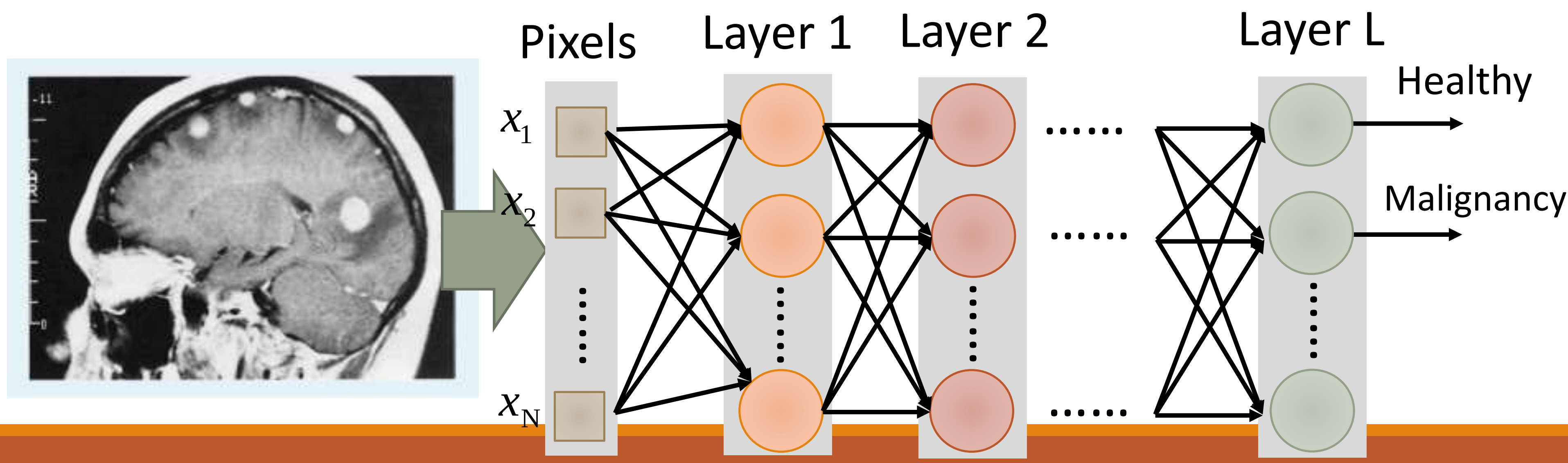
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Transfer Learning

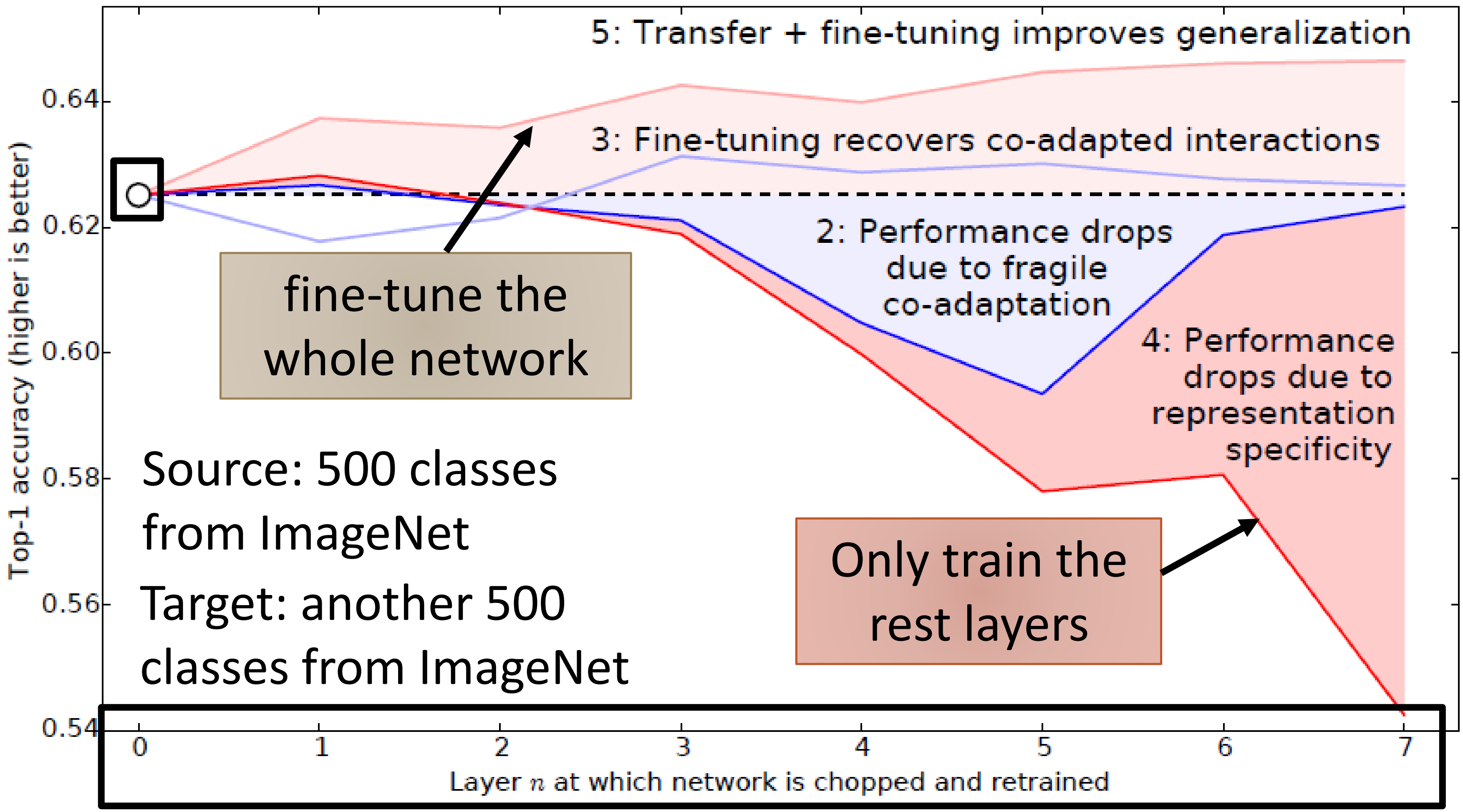


Transfer Learning



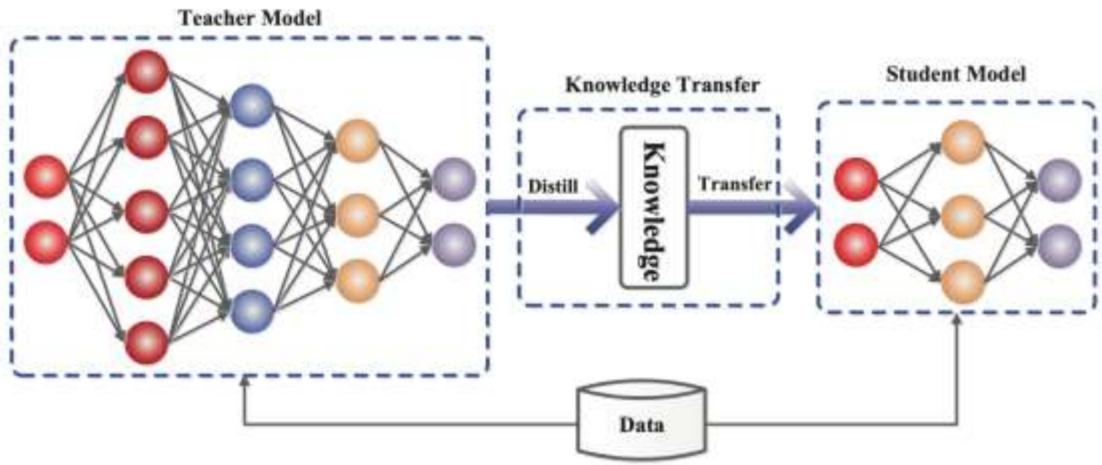


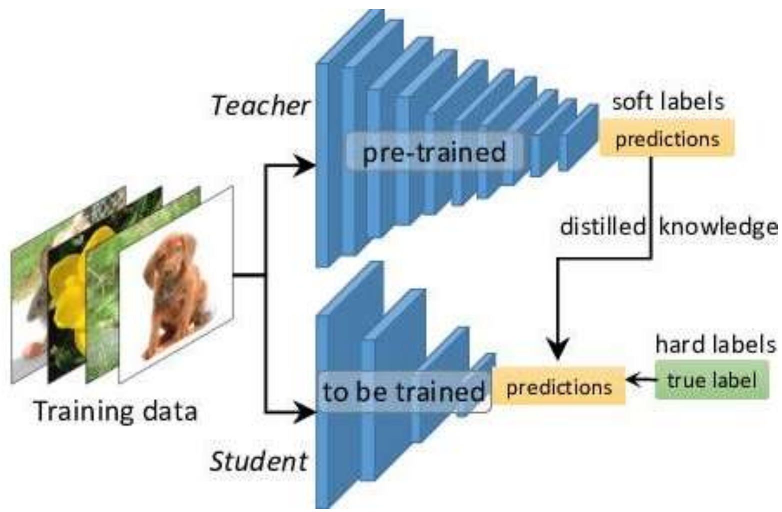
Layer Transfer - Image



J. Yosinski, J. Clune, Y. Bengio, H. Lipson, "How transferable are features in deep neural networks?", NIPS, 2014

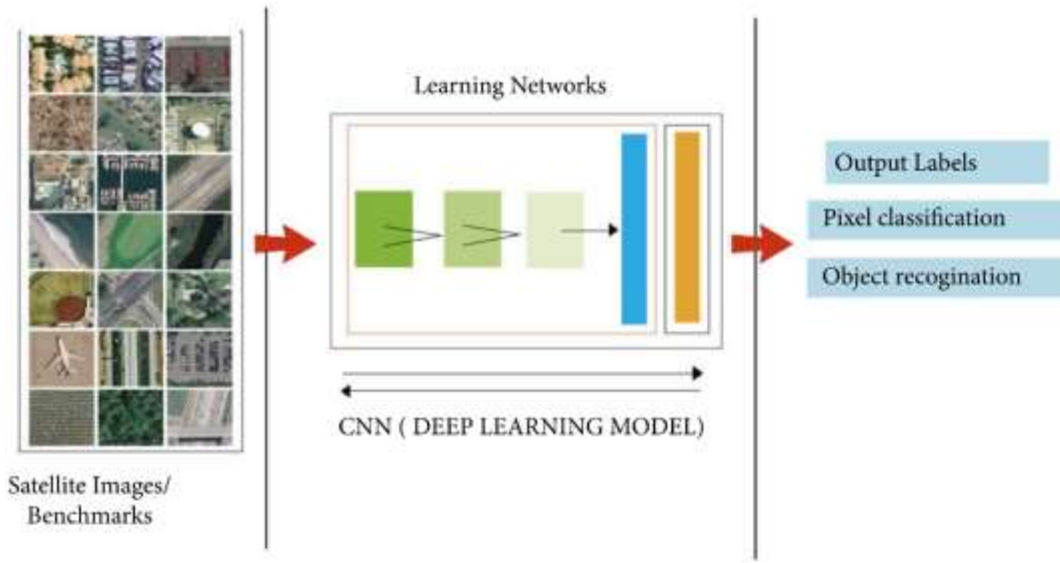
Knowledge distillation

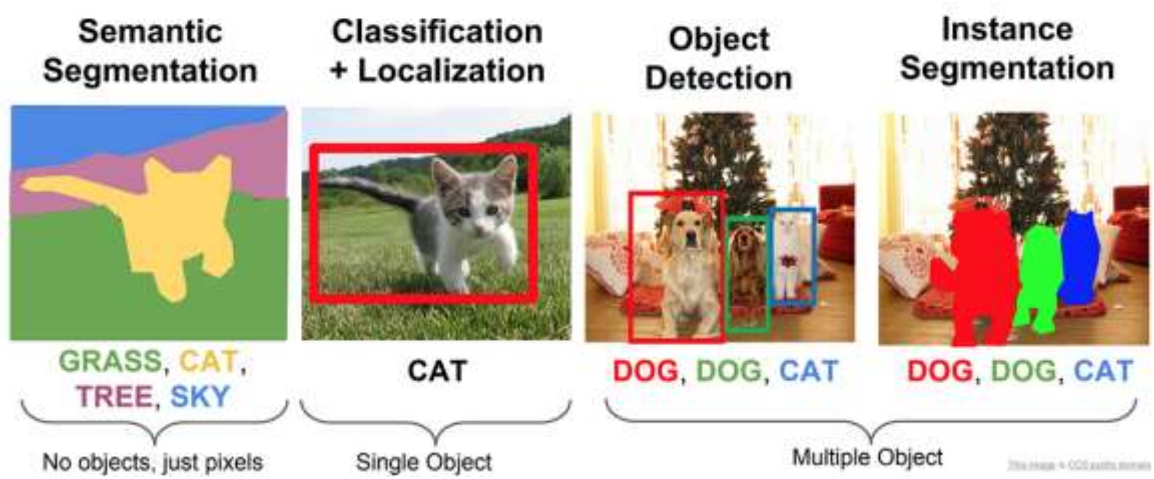




https://neptune.ai/blog/knowledge-distillation

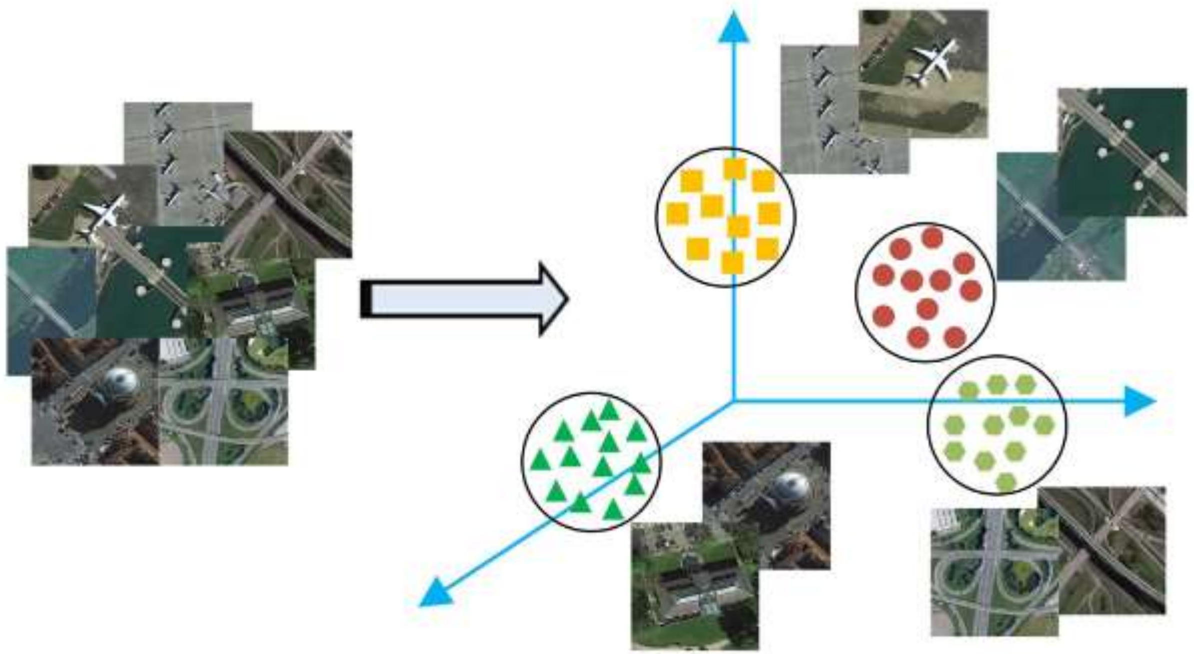
Applications in RS





Multi-class classification

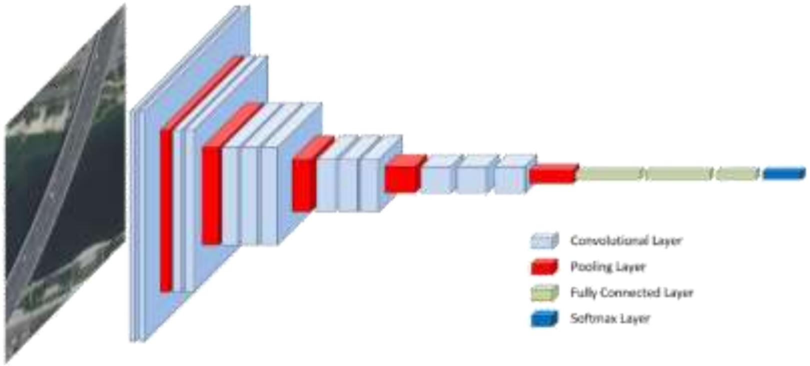
Scene classification



Remote Sensing Image Scene ClassificationMeets Deep Learning: Challenges, Methods,Benchmarks, and Opportunities

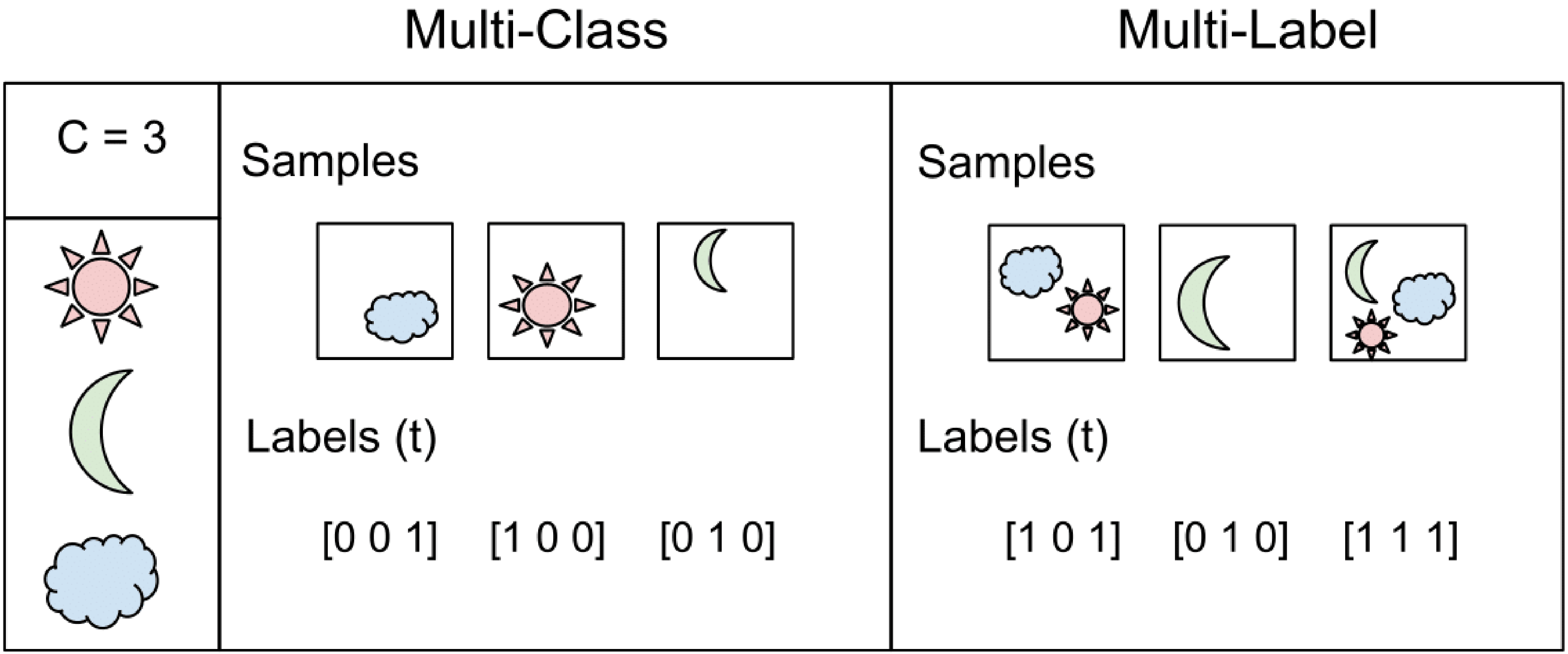
Performance on UCMerced

CNN-based	GBRCN [102]	2015	IEEE TGRS	-	94.53
	LPCNN [103]	2016	JARS	-	89.90
	Fusion by Addition [109]	2017	IEEE TGRS	-	97.42±1.79
	ARCNet-VGG16 [74]	2018	IEEE TGRS	96.81±0.14	99.12±0.40
	MSCP [112]	2018	IEEE TGRS	-	98.36±0.58
	D-CNNs [73]	2018	IEEE TGRS	-	98.93±0.10
	MCNN [116]	2018	IEEE TGRS	-	96.66±0.9
	ADSSM [138]	2018	IEEE TGRS	-	99.76±0.24
	FACNN [113]	2019	IEEE TGRS	-	98.81±0.24
	SF-CNN [118]	2019	IEEE TGRS	-	99.05±0.27
	SCCov [123]	2019	IEEE TNNLS	-	99.05±0.25
	RSFJR [117]	2019	IEEE TGRS	97.21±0.65	-
	GBN [119]	2019	IEEE TGRS	97.05±0.19	98.57±0.48
	ADFF [139]	2019	Remote Sensing	96.05±0.56	97.53±0.63
	CNN-CapsNet [140]	2019	Remote Sensing	97.59±0.16	99.05±0.24
	Siamese ResNet50 [141]	2019	IEEE GRSL	90.95	94.29



Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods ,Benchmarks, and Opportunities

Multi-class vs. Multi-label



Classes are mutually exclusive

No restrictions on the number of associated labels per sample

Multiclass vs Multi-label

Land Cover, Scene Classification

Dense Residential | **Airplane**

Sparse Residential | **Forest**

Scene characterization

Buildings, Cars, Pavement, Trees | **Airplane, Cars, Grass, Pavement**

Buildings, Cars, Chaparral, Pavement, Sand, Trees | **Trees**

A CNN based approach

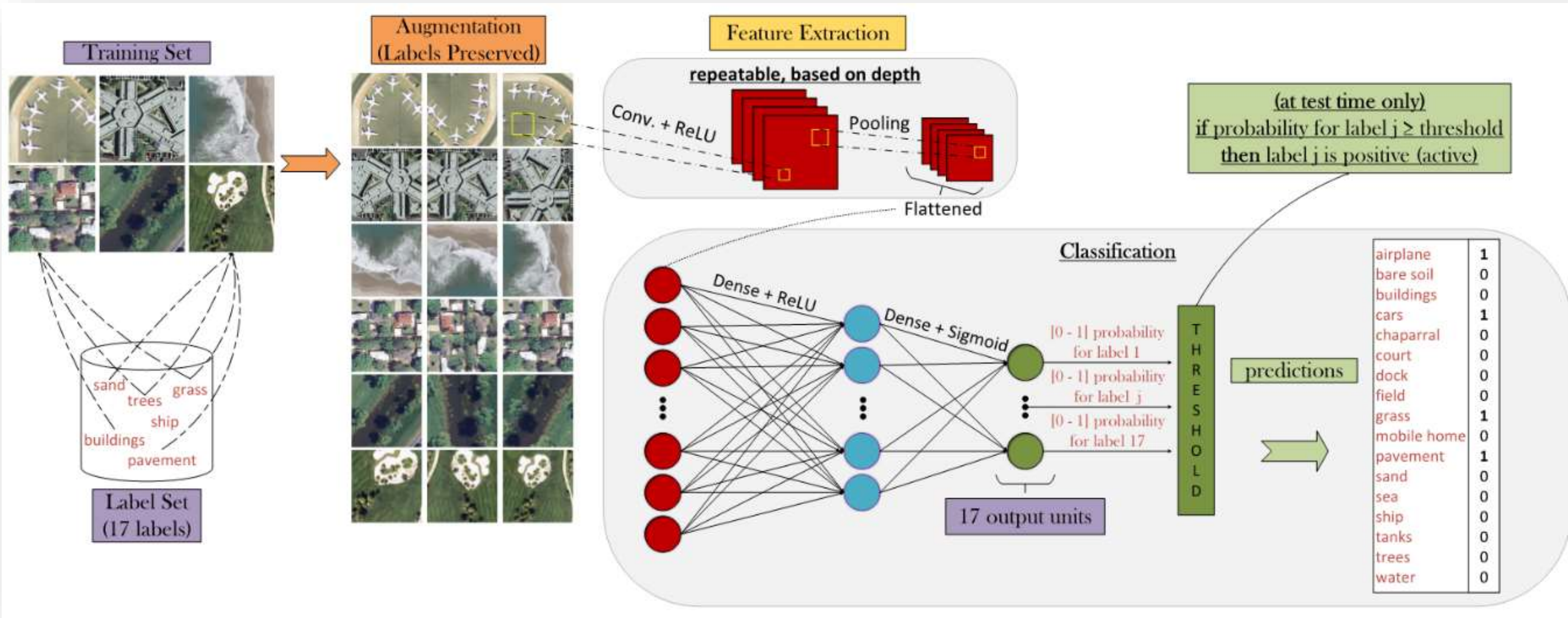
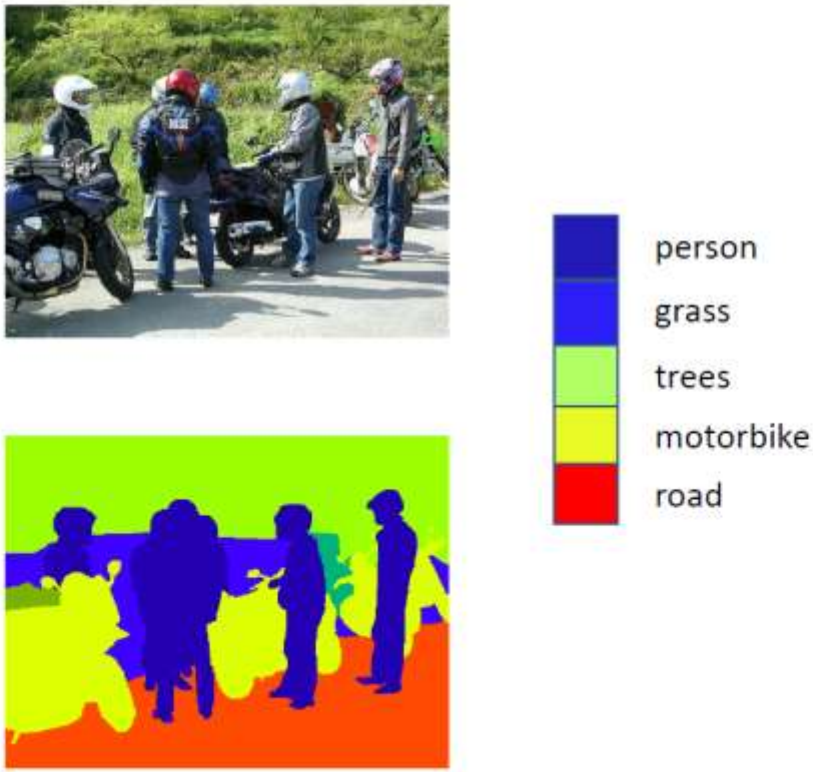
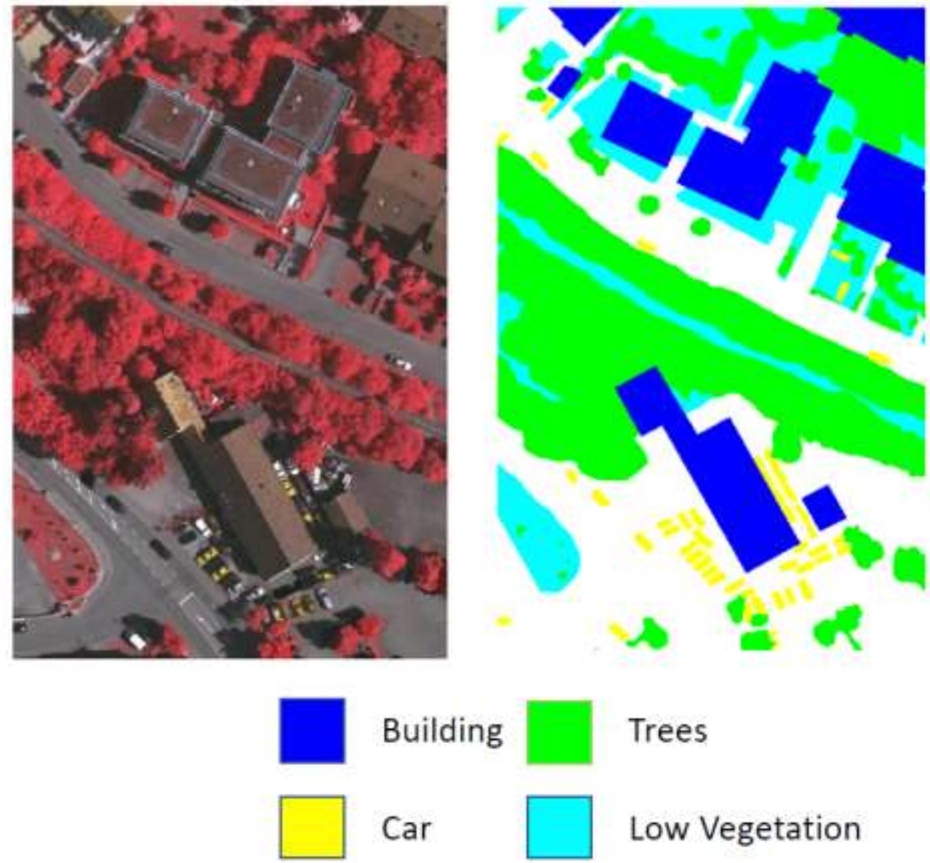


Image segmentation

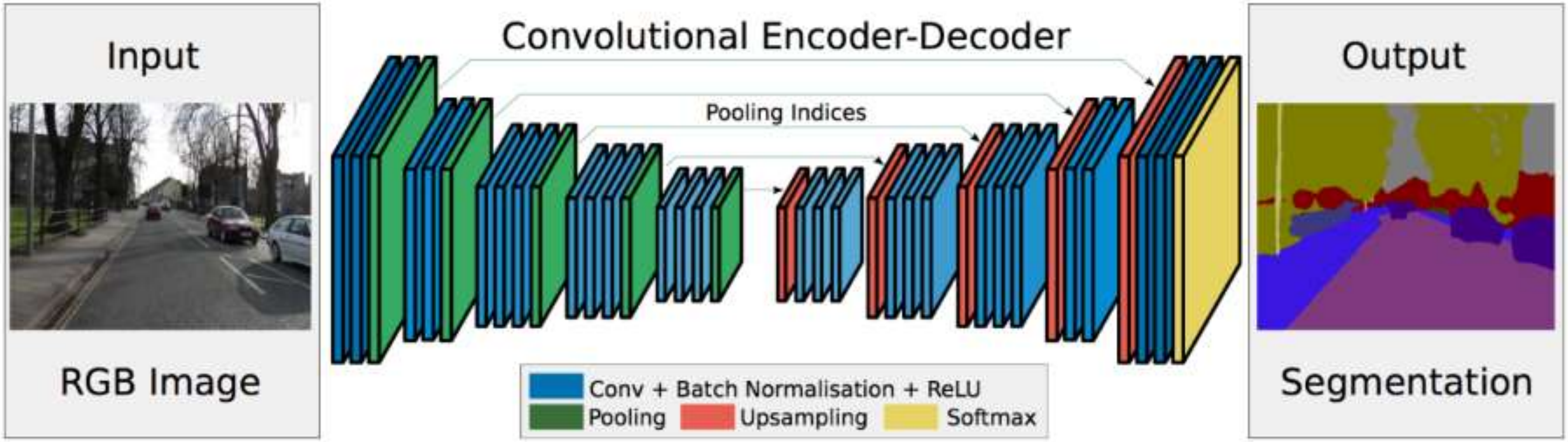




SegNet

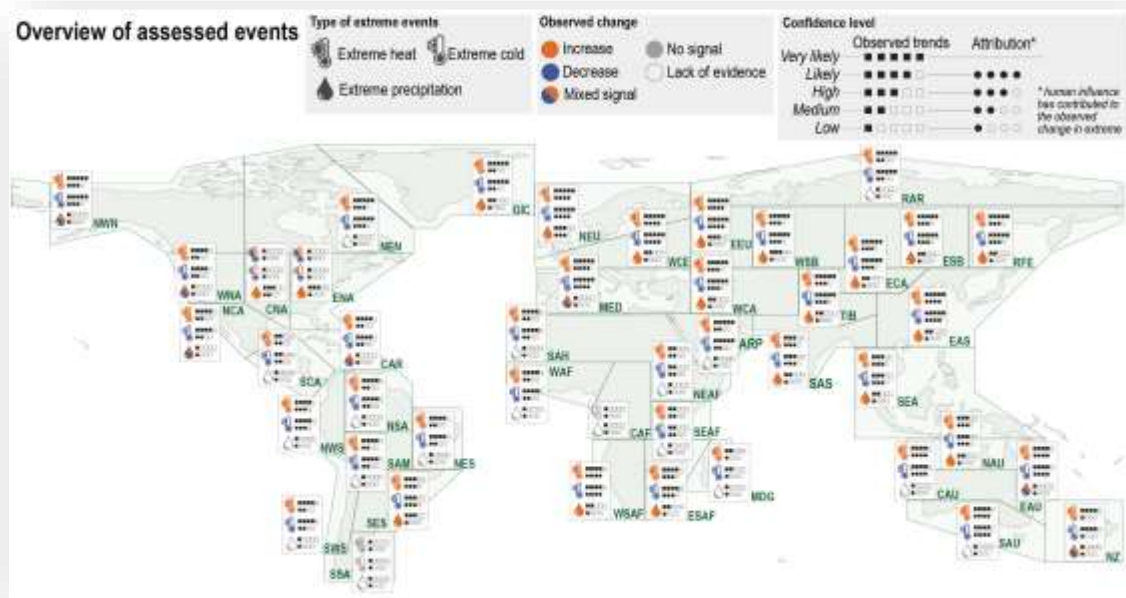
Fully convolutional networks (FCN) à no dense layers

Can handle different input sizes

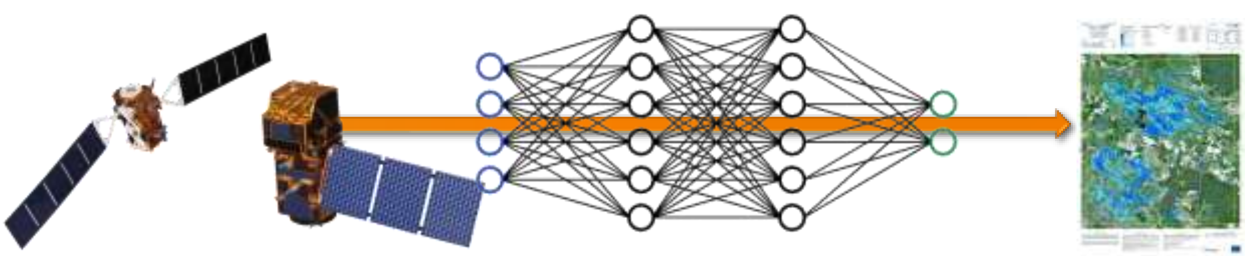


Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." IEEE Transactions on pattern analysis and machine intelligence 39.12 (2017): 2481-2495.

Extreme weather events



[IPCC Sixth Assessment Report]

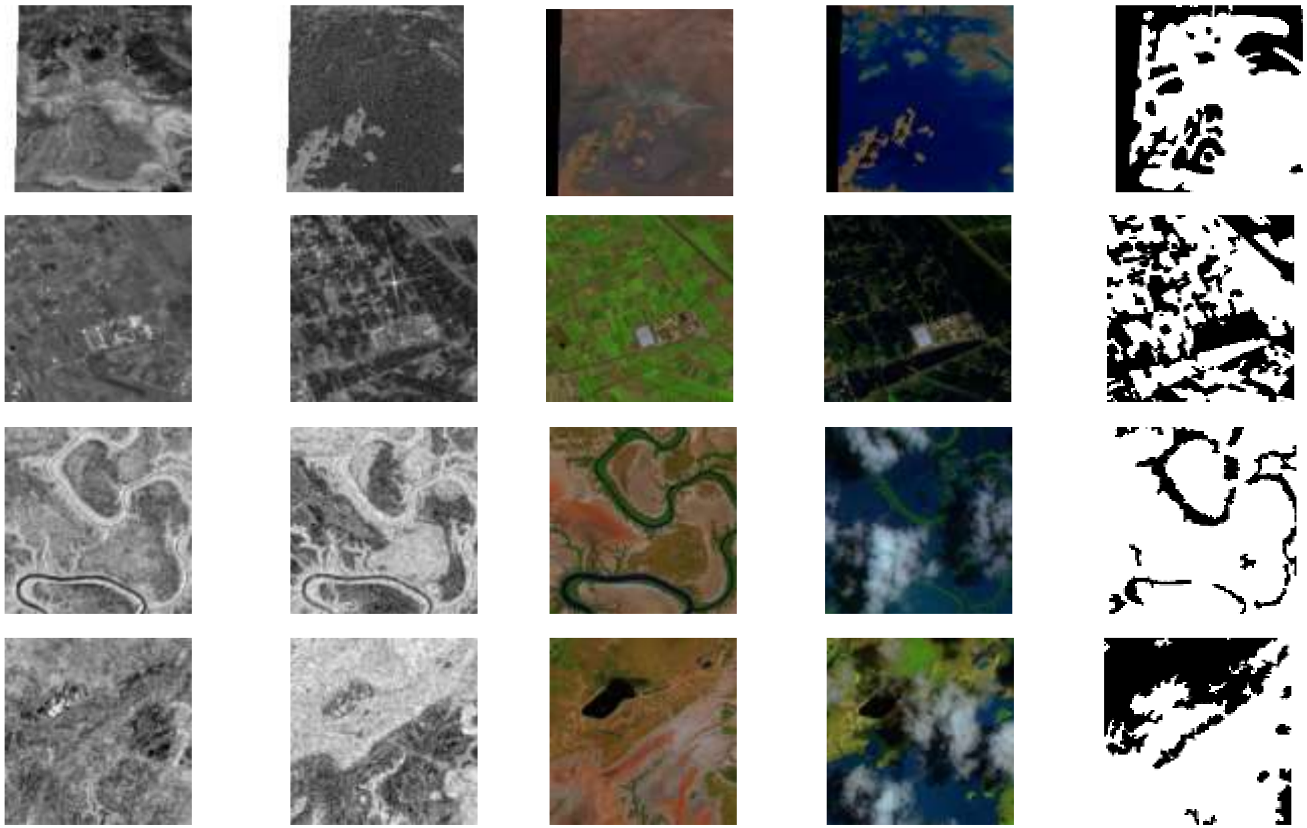


OMBRIA dataset

EMS ID	Country	Date 1	Date 2	UTM Zone
271	Greece	01/05/2017	28/02/2018	34 N
273	Albania	01/05/2017	11/03/2018	34 N
275	Croatia	01/05/2017	22/03/2018	33 N
279	Spain	01/05/2017	15/04/2018	30 N
324	France	01/05/2018	16/10/2018	31 N
342	Australia	15/04/2018	13/02/2019	54 S
388	Spain	01/05/2019	14/09/2019	30 N
416	France	01/05/2019	15/12/2019	30 N
417	Portugal	01/05/2019	23/12/2019	29 N
419	Iran	01/05/2019	13/01/2020	41 N
422	Spain	01/05/2019	26/01/2020	31 N
424	Madagascar	01/05/2019	29/01/2020	39 S
429	Ireland	01/05/2019	23/02/2020	29 N
441	Finland	01/05/2019	04/06/2020	34 N
465	Greece	01/05/2020	20/09/2020	31 N
466	Niger	01/05/2020	27/09/2020	32 N
468	Italy	01/05/2020	10/10/2020	32 N
470	Togo	01/05/2020	17/10/2020	31 N
482	Honduras	01/05/2020	22/11/2020	17 N
492	France	01/05/2020	02/01/2021	30 N
501	Albania	01/05/2020	15/02/2021	35 N
507	Timor	01/05/2020	06/04/2021	51 S
514	Guyana	01/05/2020	06/06/2021	21 S

Drakonakis, G. I., Tsagkatakis, G., Fotiadou, K., & Tsakalides, P. “Ombrianet—supervised flood mapping via convolutional neural networks using multitemporal sentinel-1 and sentinel-2 data fusion”. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2022.

OMBRIA dataset



Pre Post Pre Post

Sentinel 1 - SAR

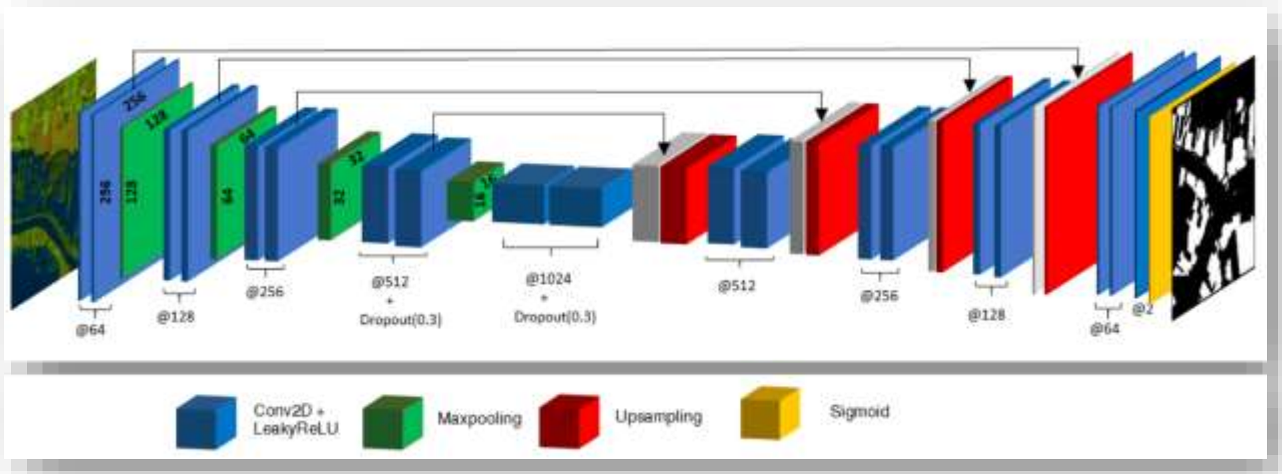
Sentinel 2 - MSI

Mask
(White pixels is flood)

2D UNET

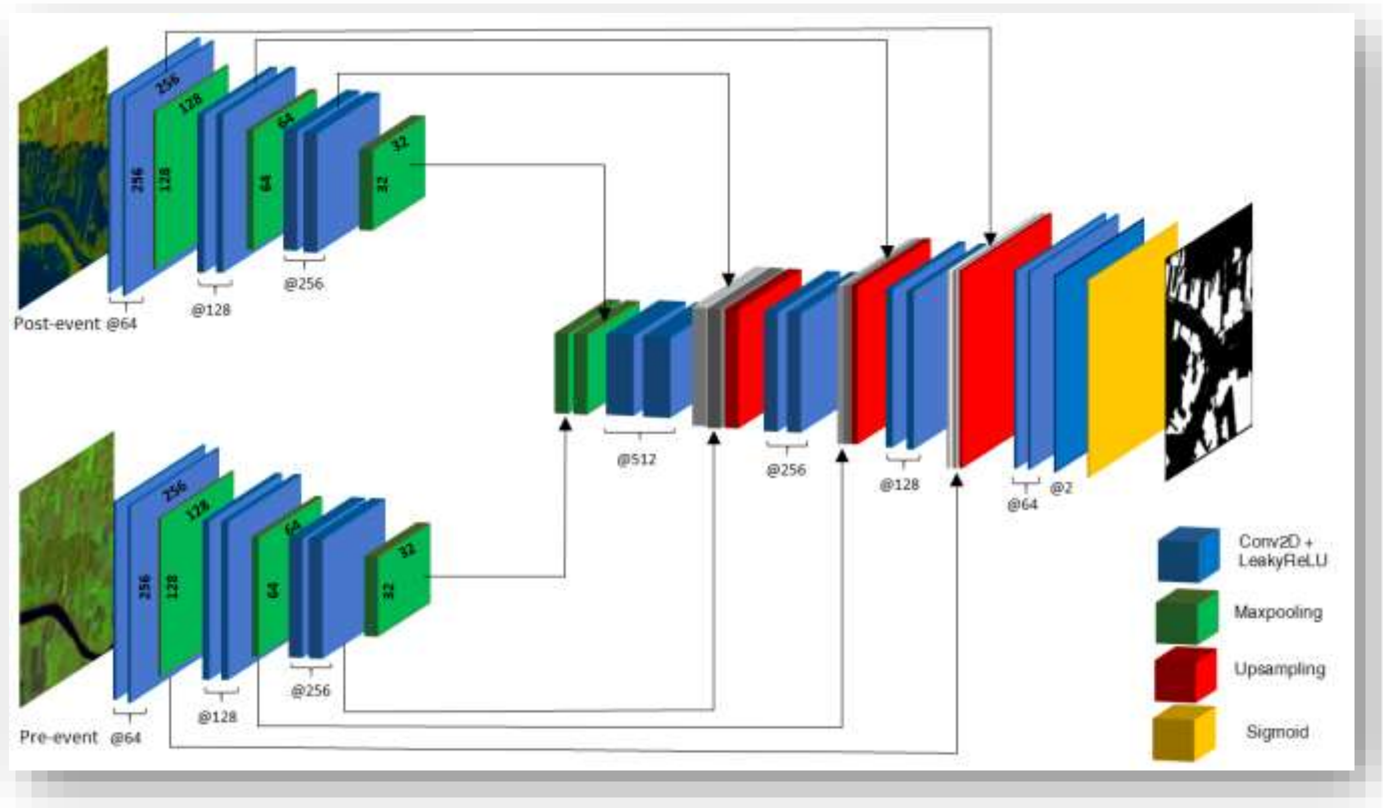
U-NET:

- Image -> image
- Image segmentation



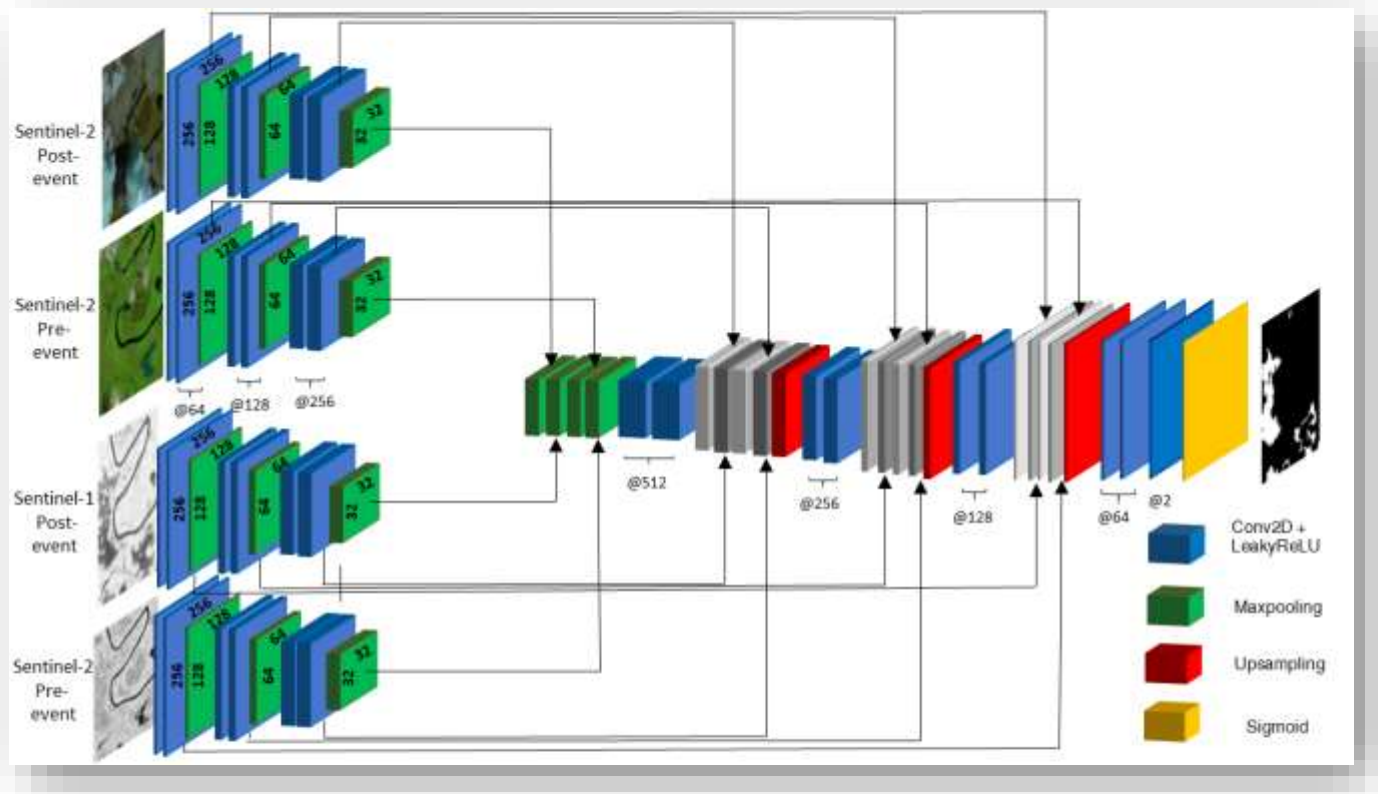
Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*.

3D UNET



Post-event @64
@128
@256
Pre-event @64
@128
@256
@512
@256
@128
@64
@2
Conv2D +
LeakyReLU
Maxpooling
Upsampling
Sigmoid

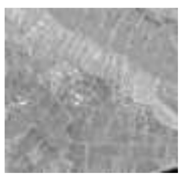
4D UNET

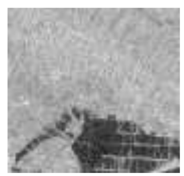


Sentinel-2
Post-
event
Sentinel-2
Pre-
event
Sentinel-1
Post-
event
Sentinel-2
Pre-
event
@64
@128
@256
@512
@256
@128
@64
@2
Conv2D +
LeakyReLU
Maxpooling
Upsampling
Sigmoid

OMBRIA NET

Methods	PA	IoU	FW IoU
Otsu's Thresholding (Sentinel-2)	0.7930	0.333	0.6889
Multimodal SVM	0.8504	0.6245	0.7631
U-Net (Sentinel-1)	0.7925	0.5734	0.6971
U-Net (Sentinel-2)	0.8251	0.5418	0.7221
Bitemporal OmbriaNet (Sentinel-1)	0.7203	0.5181	0.6229
Bitemporal OmbriaNet (Sentinel-2)	0.8733	0.6457	0.7919
Multimodal OmbriaNet (Sentinel-1 & Sentinel-2)	**0.9010**	**0.7236**	**0.8330**


(a) S-1 (pre)


(b) S-1 (post)


(c) S-2 (pre)


(d) S-2 (post)


(e) S-1 (63.8 %)

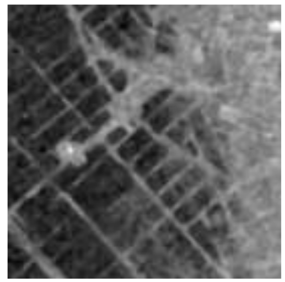

(f) S-2
(78.3%)


(g) OmbriaNet (89.1%)


(h) Ground Truth
(White pixels is flood)

Performance


(a) S-1 (Pre-event)


(b) S-1 (Post-event)


(c) S-2 (Pre-event)


(d) S-2 (Post-event)


(e) S-1 U-Net
(22.19%)

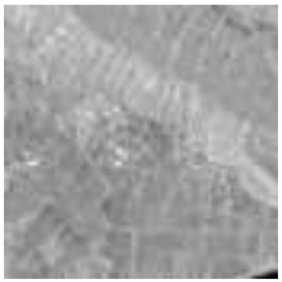

(f) S-2 U-Net
(59:13%)


(g) OmbriaNet
(81:90%)

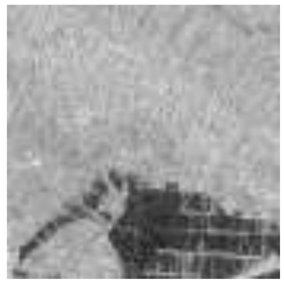

(h) Ground Truth

Performance

Comparison of selected sample from ID501 flood in Albania (IoU metric score)

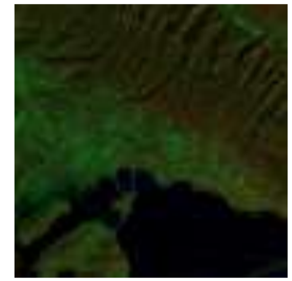


(a) S-1 (Pre-event)



(b) S-1 (Post-event)

(c) S-2 (Pre-event)



(d) S-2 (Post-event)

(e) Sentinel-1 U-Net (63.82 %)

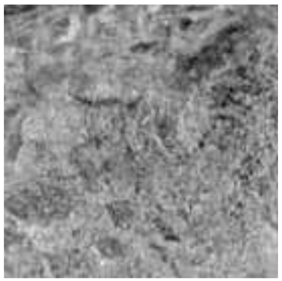
(f) Sentinel-2 U-Net (78:30%)

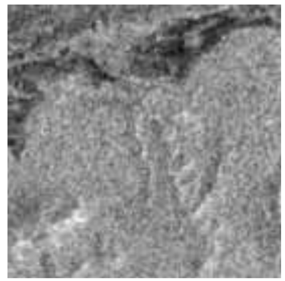
(g) OmbriaNet (89:14%)

(h) Ground Truth (White pixels is flood)

Performance

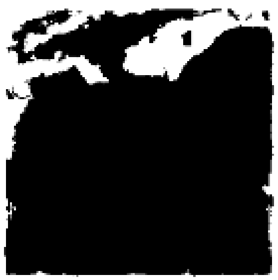
Comparison of selected sample from ID507 flood in Timor

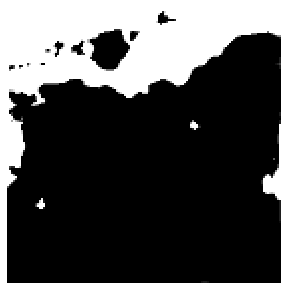

(a) S-1 (Pre-event)


(b) S-1 (Post-event)


(c) S-2 (Pre-event)


(d) S-2 (Post-event)


(e) Sentinel-1 U-Net (69:50 %)


(f) Sentinel-2 U-Net (79:08%)


(g) OmbriaNet (79:44%)


(h) Ground Truth (White pixels is flood)

Active fire detection

$$((R_{75} > 2.5) \text{ and } (\rho_7 - \rho_5 > 0.3) \text{ and } (\rho_7 > 0.5)) \text{ or } ((\rho_6 > 0.8) \text{ and } (\rho_1 < 0.2) \text{ and } (\rho_5 > 0.4 \text{ or } \rho_7 < 0.1))$$

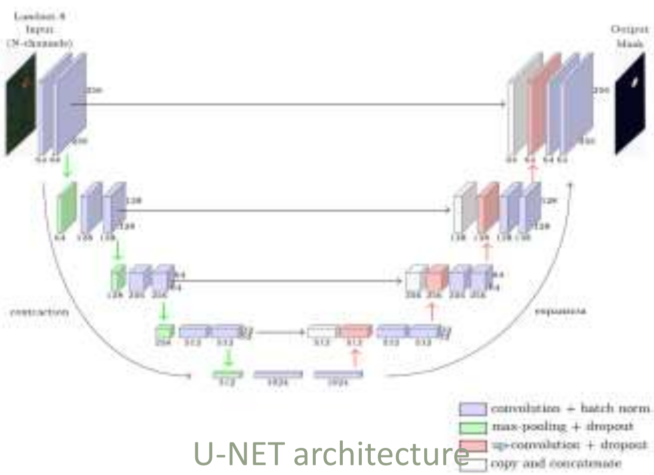
Schroeder et al. (2016)

$$(R_{76} \geqslant 1.4) \text{ and } (R_{75} \geqslant 1.4) \text{ and } (\rho_7 \geqslant 0.15)$$

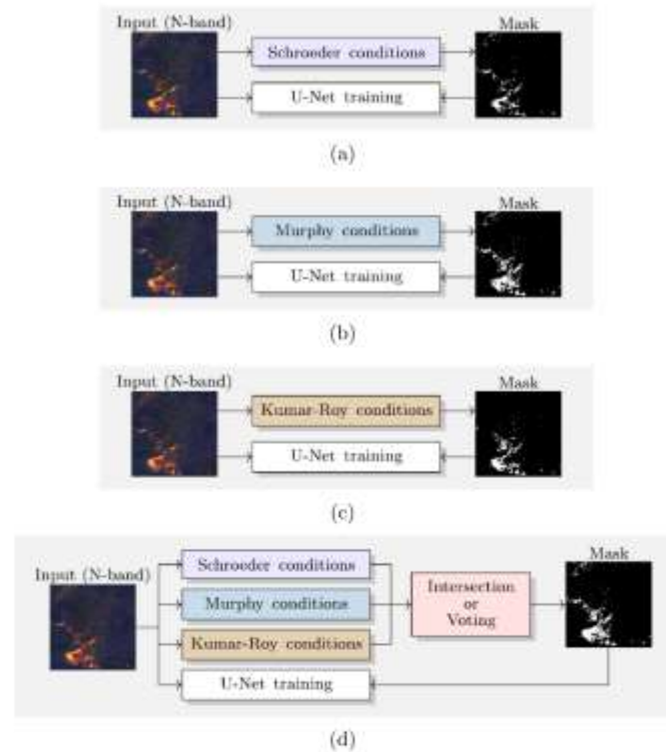
Murphy et al. (2016)

$$\rho_4 \leqslant 0.53\, \rho_7 - 0.214$$

Kumar-Roy



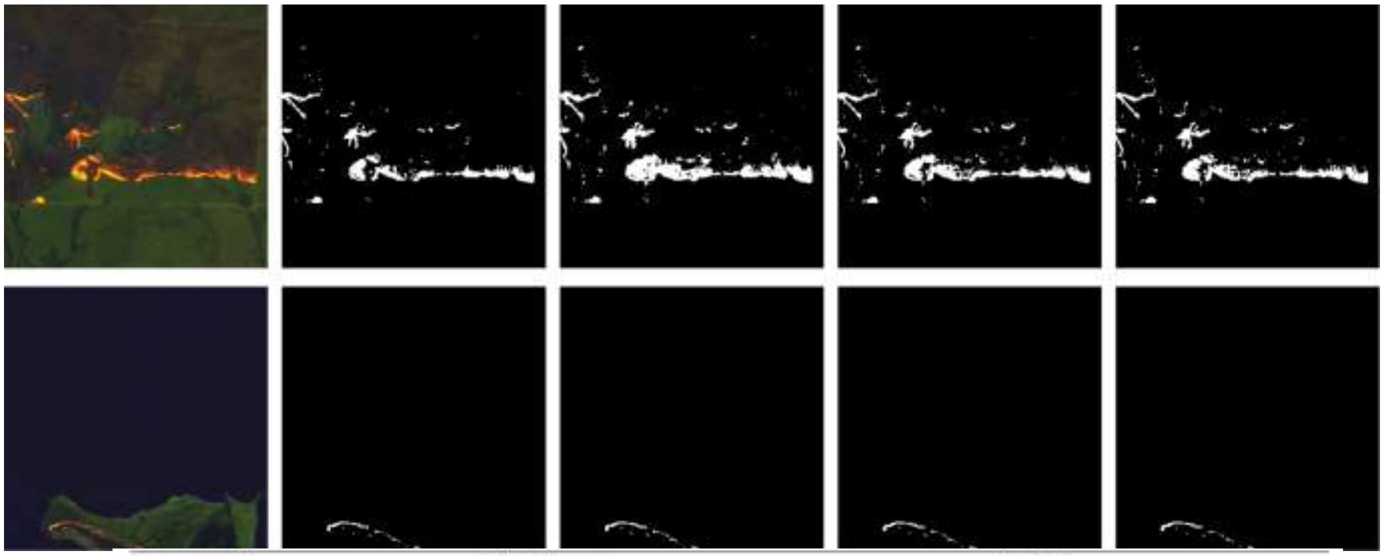
U-NET architecture



de Almeida Pereira, Gabriel Henrique, et al. "Active fire detection in Landsat-8 imagery: A large-scale dataset and a deep-learning study." *ISPRS Journal of Photogrammetry and Remote Sensing* 178 (2021): 171-186.

Active fire detection

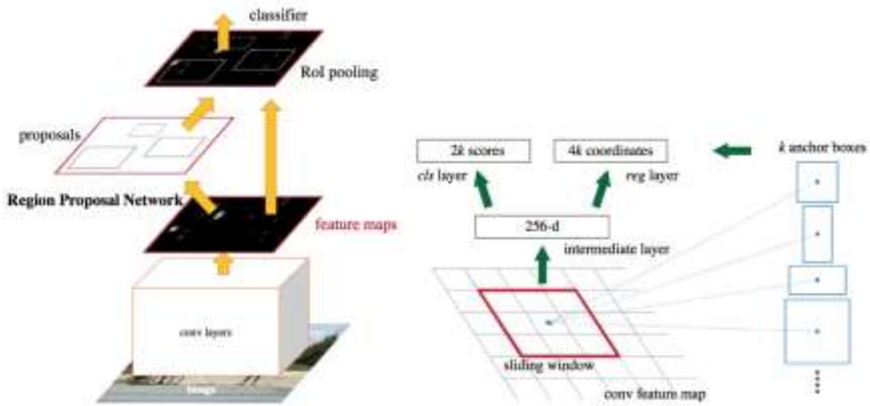
Input (LS8) | Schroeder et al | U-Net (10 ch) | U-Net (7-6-2) | U-Net light (7-6-2)



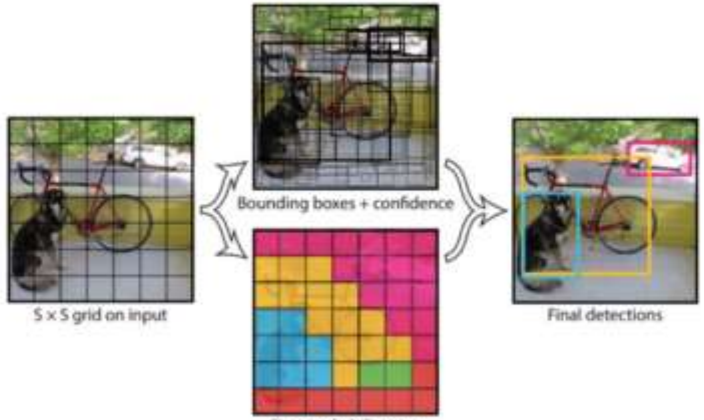
Mask	CNN Architecture	Metrics (%)			
		P ↑	R ↑	IoU	F
Schroeder et al. (non-CNN)	–	88.1	70.2	64.1	78.1
Schroeder et al.	U-Net (10c)	86.9	88.4	78.0	87.7
	U-Net (3c)	89.5	80.5	73.6	84.8
	U-Net-Light (3c)	89.0	78.8	71.8	83.6
Murphy et al. (non-CNN)	–	76.6	96.1	74.3	85.2
Murphy et al.	U-Net (10c)	74.8	96.9	73.0	84.4
	U-Net (3c)	72.7	97.2	71.2	83.2
	U-Net-Light (3c)	75.5	96.9	73.7	84.9
Kumar-Roy (non-CNN)	–	82.3	68.4	59.7	74.7
Kumar-Roy	U-Net (10c)	82.2	94.2	78.3	87.8
	U-Net (3c)	84.5	93.4	79.8	88.8
	U-Net-Light (3c)	78.8	96.9	76.9	86.9
Intersection (non-CNN)	–	92.6	57.6	55.1	71.0
Intersection {Schroeder et al., Murphy et al., Kumar – Roy}	U-Net (10c)	91.8	75.4	70.6	82.8
	U-Net (3c)	90.8	73.5	68.4	81.2
	U-Net-Light (3c)	90.8	72.8	67.8	80.8
Voting (non-CNN)	–	87.7	79.1	71.2	83.2
Voting {Schroeder et al., Murphy et al., Kumar – Roy}	U-Net (10c)	83.6	94.0	79.3	88.5
	U-Net (3c)	86.4	93.0	81.1	89.6
	U-Net-Light (3c)	87.2	92.4	81.4	89.7

Object detection

Faster R-CNN



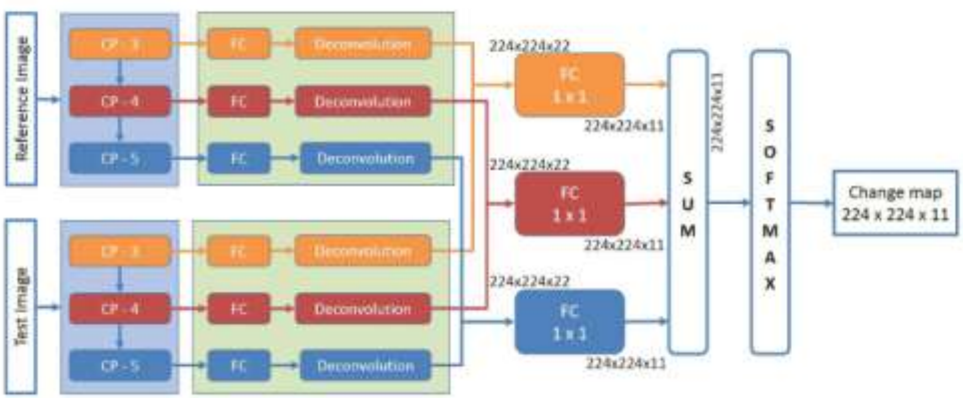
YOLO — You Only Look Once



Change detection

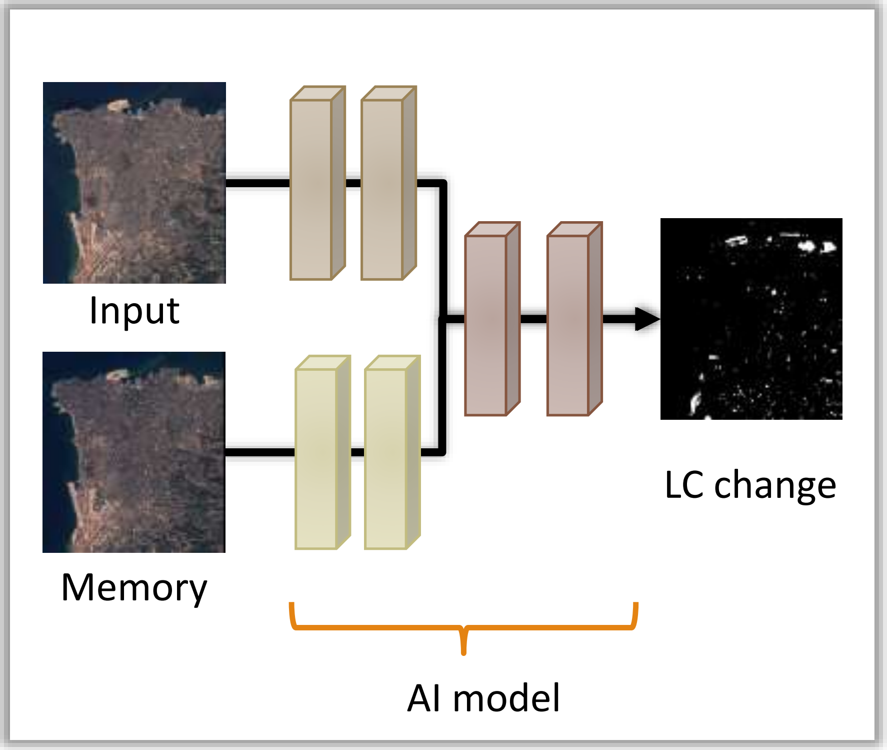
Challenges:

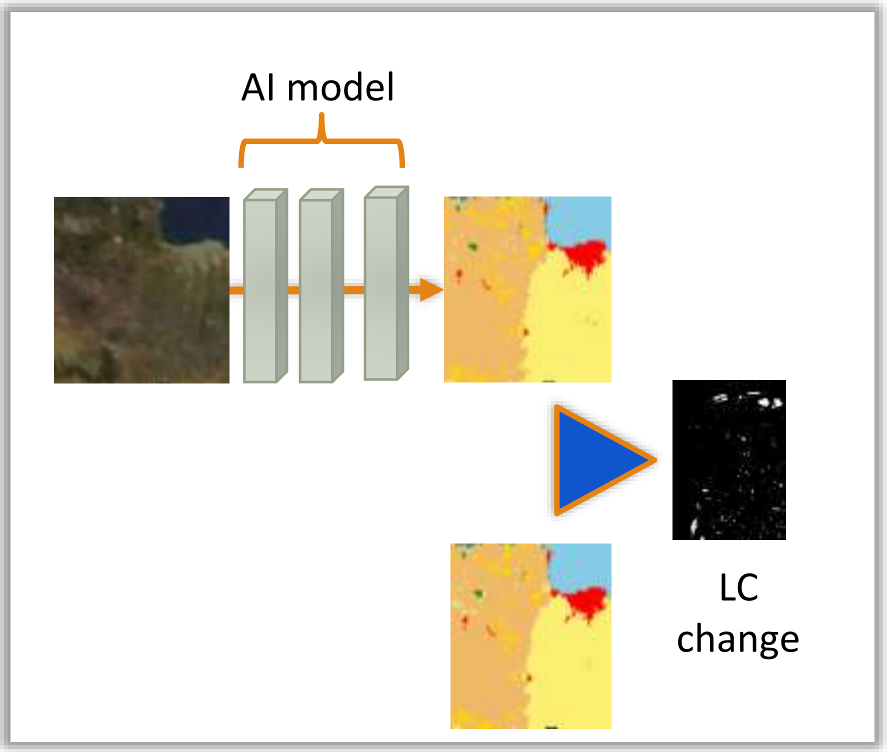
- ➢Changes in lighting, atmospheric conditions, and seasonality
- ➢Lack of High-Quality, Annotated Data
- ➢Time-series dynamics
- ➢Heterogeneity of Data



Varghese, Ashley, et al. "ChangeNet: A deep learning architecture for visual change detection." *Proceedings of the European conference on computer vision (ECCV) workshops*. 2018.

Land cover change detection





Performance metrics: memory, complexity, speed, accuracy

Performance

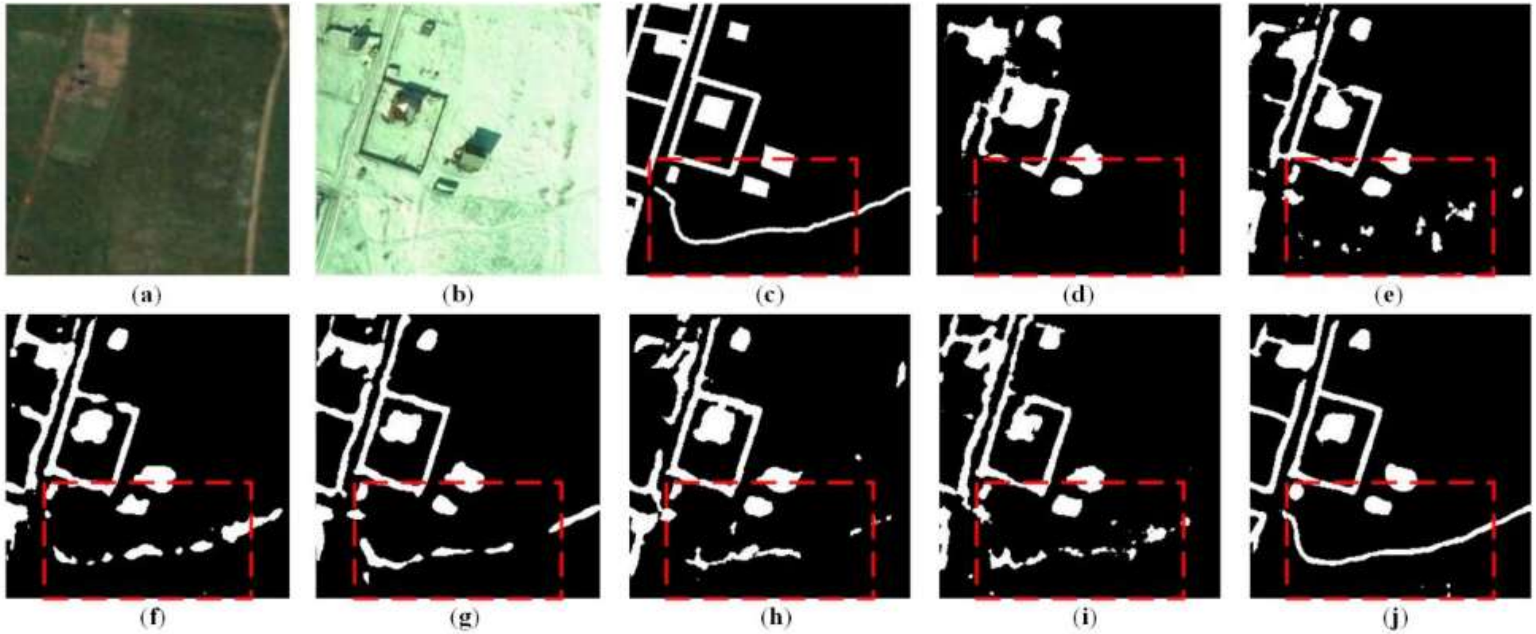


Figure 5. Visual comparison of CD results using various DL methods for area 6: (**a**) image T1, (**b**) image T2, (**c**) reference change map, (**d**) CDNet, (**e**) FC-EF, (**f**) FC-Siam-conc, (**g**) FC-Siam-diff, (**h**) FC-EF-Res, (**i**) FCN-PP, and (**j**) U-Net++. The changed parts are marked in white while the unchanged are in black.

Shafique, Ayesha, et al. "Deep learning-based change detection in remote sensing images: A review." *Remote Sensing* 14.4 (2022): 871.

TensorFlow

Deep learning library, open-sourced by Google (11/2015)

TensorFlow provides primitives for

- defining functions on tensors
- automatically computing their derivatives

What is a tensor

What is a computational graph

Material from lecture by Bharath Ramsundar, March 2018, Stanford

Introduction to Keras

Official high-level API of TensorFlow

- Python
- 250K developers

Same front-end <-> Different back-ends

- TensorFlow (Google)
- CNTK (Microsoft)
- MXNet (Apache)
- Theano (RIP)

Hardware

- GPU (Nvidia)
- CPU (Intel/AMD)
- TPU (Google)

Companies: Netflix, Uber, Google, Nvidia…

Material from lecture by Francois Chollet, 2018, Stanford

Keras models

Installation

- Anaconda -> Tensorflow -> Keras

Build-in

- Conv1D, Conv2D, Conv3D…
- MaxPooling1D, MaxPooling2D, MaxPooling3D…
- Dense, Activation, RNN…

The Sequential Model

- Very simple
- Single-input, Single-output, sequential layer stacks

The functional API

- Mix & Match
- Multi-input, multi-output, arbitrary static graph topologies

Sequential

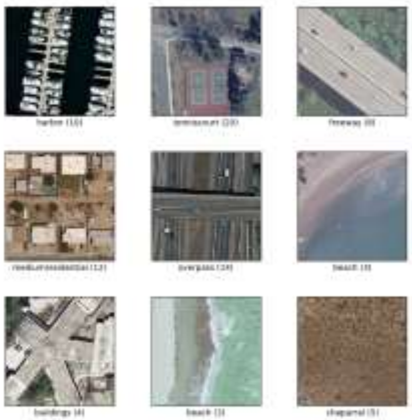
```
>>from keras.models import Sequential
>>model = Sequential()
>> from keras.layers import Dense
>> model.add(Dense(units=64, activation='relu', input_dim=100))
>> model.add(Dense(units=10, activation='softmax'))
>> model.compile(loss='categorical_crossentropy',
optimizer='sgd',    metrics=['accuracy'])
>> model.fit(x_train, y_train, epochs=5, batch_size=32)
>> loss_and_metrics = model.evaluate(x_test, y_test,
batch_size=128)
>> classes = model.predict(x_test)
```

Functional

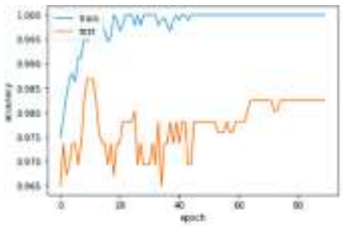
```
>> from keras.layers import Input, Dense
>> from keras.models import Model
>> inputs = Input(shape=(784,))
>> x = Dense(64, activation='relu')(inputs)
>> x = Dense(64, activation='relu')(x)
>> predictions = Dense(10, activation='softmax')(x)
>> model = Model(inputs=inputs, outputs=predictions)
>> model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', metrics=['accuracy'])
>> model.fit(data, labels)
```

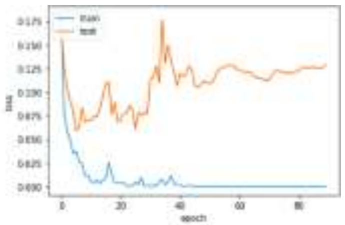
TensorFlow	PyTorch
Written in C++ and is, as a result, very fast and efficient.	Written in Python making it more accessible and flattening the learning curve. However, the C++ core means PyTorch is still quite fast.
Feature rich; TensorFlow can be used for training data as well as for inference.	Very flexible; as data size can also be changed during data training.
Very good documentation; TensorFlow has many users and an big community which has led to strong documentation.	Popular at research level; Pytorch was by far the most talked about ML library at CVPR, one of the most important computer vision conferences.
High popularity; TensorFlow has established itself as the most used ML library over a number of years now.	Rapid growth in popularity in both business and research use cases.
Many APIs available; TensorFlow is a library with a rich choice of easy to use APIs.	Many libraries available; PyTorch is composed of multiple libraries and platforms.
Supports JavaScript; TensorFlow supports JavaScript, C++ and Java in addition to Python.	Python-based; PyTorch allows developers to write code in Python
For Mobile & IoT, inferences can be performed with TensorFlow Lite on mobile devices such as Android or iOS, as well as on Edge TPU or Raspberry Pi.	PyTorch API; the PyTorch API is often preferred as it is better designed - plus TensorFlow has historically changed their API frequently.

Hands-on




FEATURE LEARNING
CLASSIFICATION


train
test
epoch


train
test
epoch