Biomedical Imaging Technology (BIT) www.die.upm.es/im
Departamento de Ingeniería Electrónica, ETSIT-UPM

From Machine Learning to Deep Learning (Part 2)

David Bermejo Peláez

María Jesús Ledesma Carbayo

Temas Avanzados en Señales e Imágenes Médicas

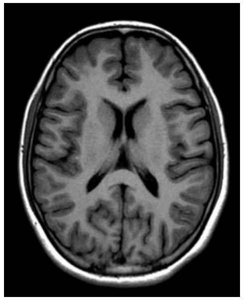
Máster en Ingeniería Biomédica

Convolutional Neural Networks (CNN) – Recap

[WxH] Input image

Function

[N] Class probability



p=0,8 Alzheimer
p=0,2 NO-Alzheimer

[HxW] = 512x320

[N] = 2

Convolutional Neural Networks (CNN) – Recap

[WxH] Input image

Function

Function

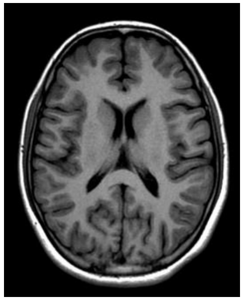
Function

Function

Function

…

[N] Class probability

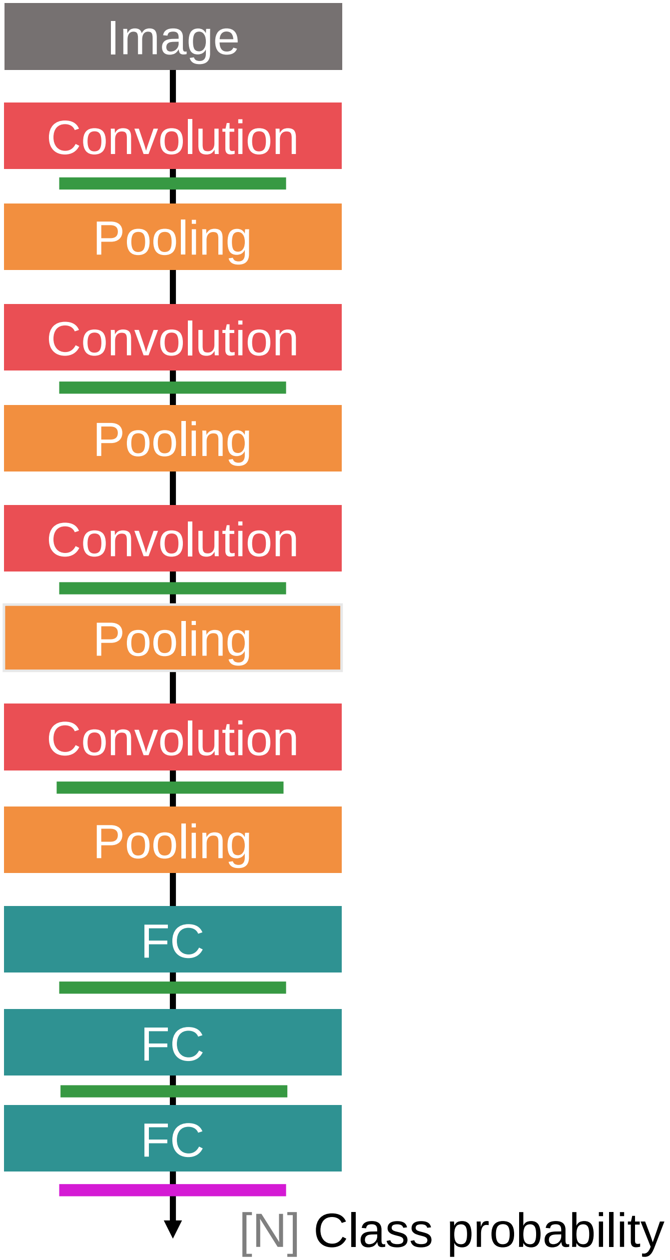


[HxW] = 512x320

p=0,8 Alzheimer
p=0,2 NO-Alzheimer

[N] = 2

Convolutional Neural Networks (CNN) – Recap



Convolution Layer

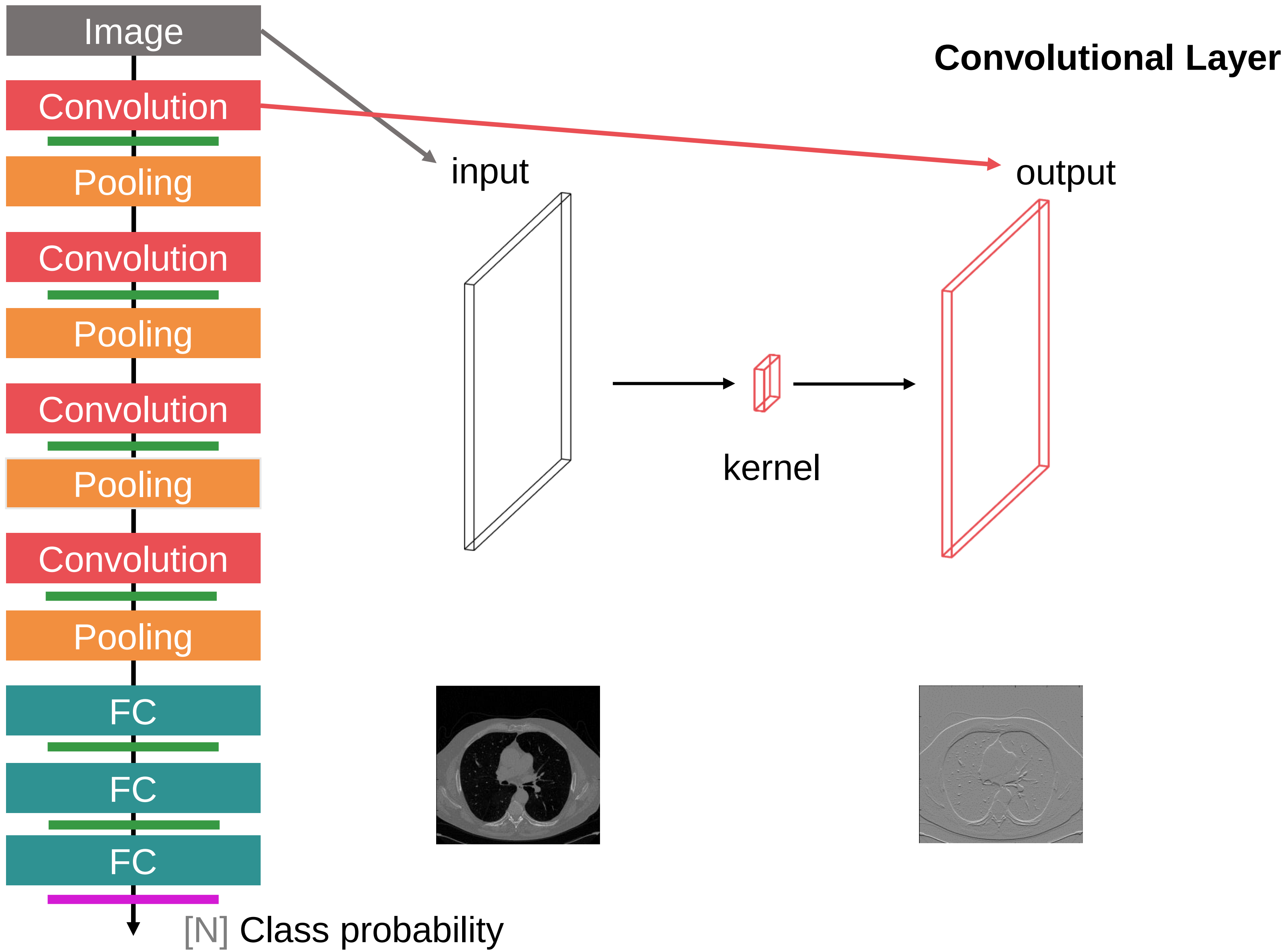
Pooling Layer

Fully-connected Layer

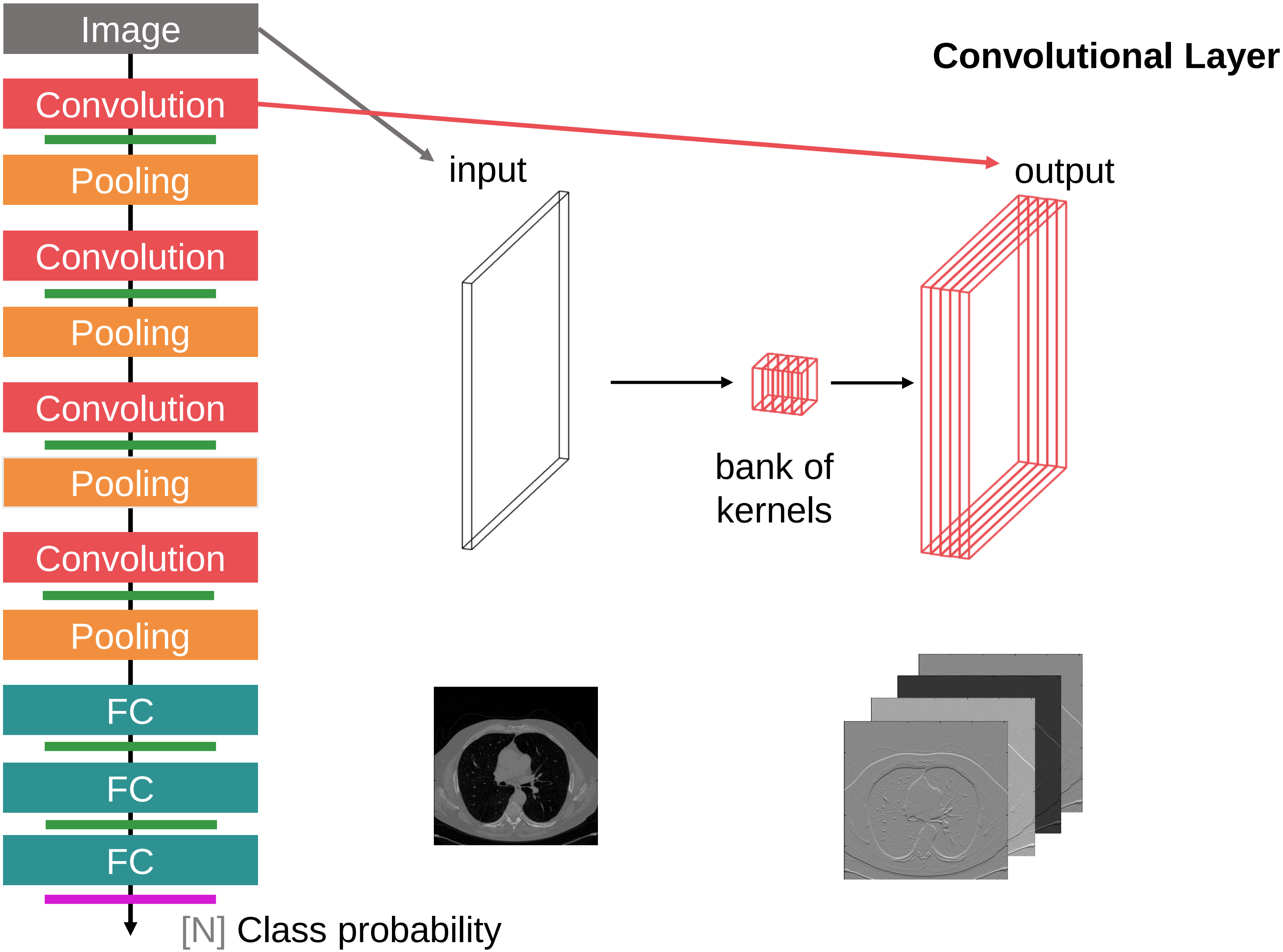
Nonlinearilies (ReLU…)

Softmax

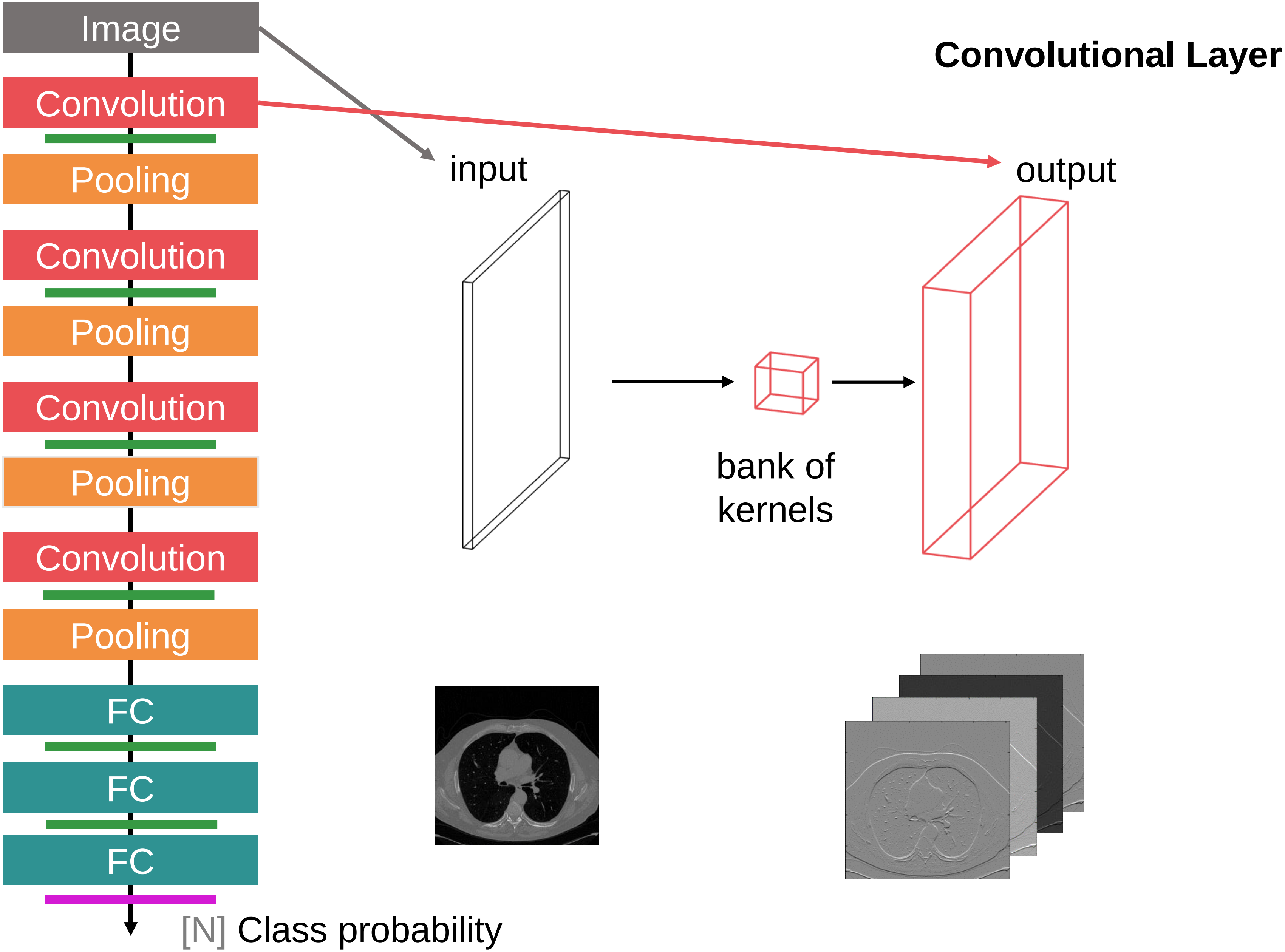
Convolutional Neural Networks (CNN) – Recap



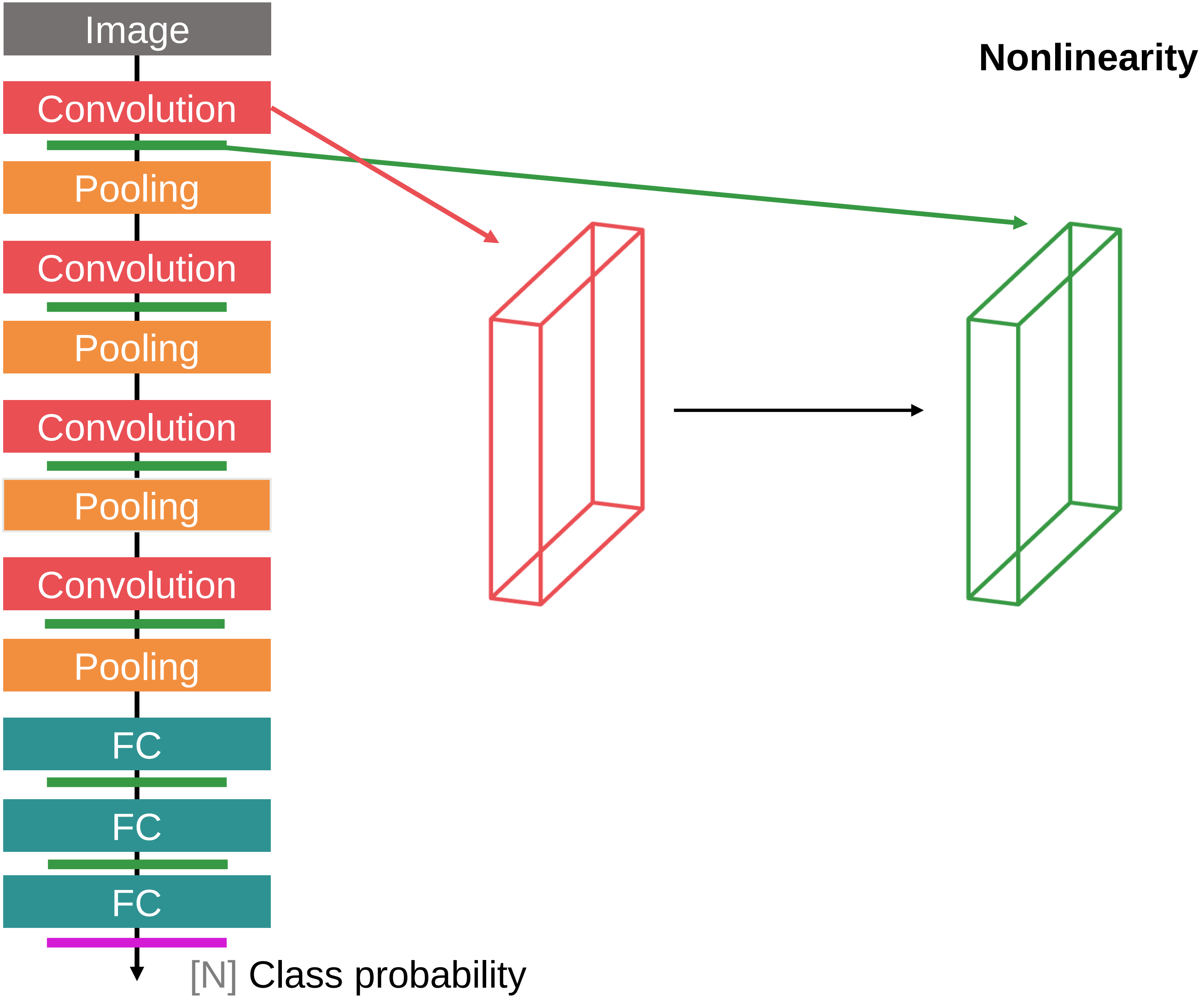
Convolutional Neural Networks (CNN) – Recap



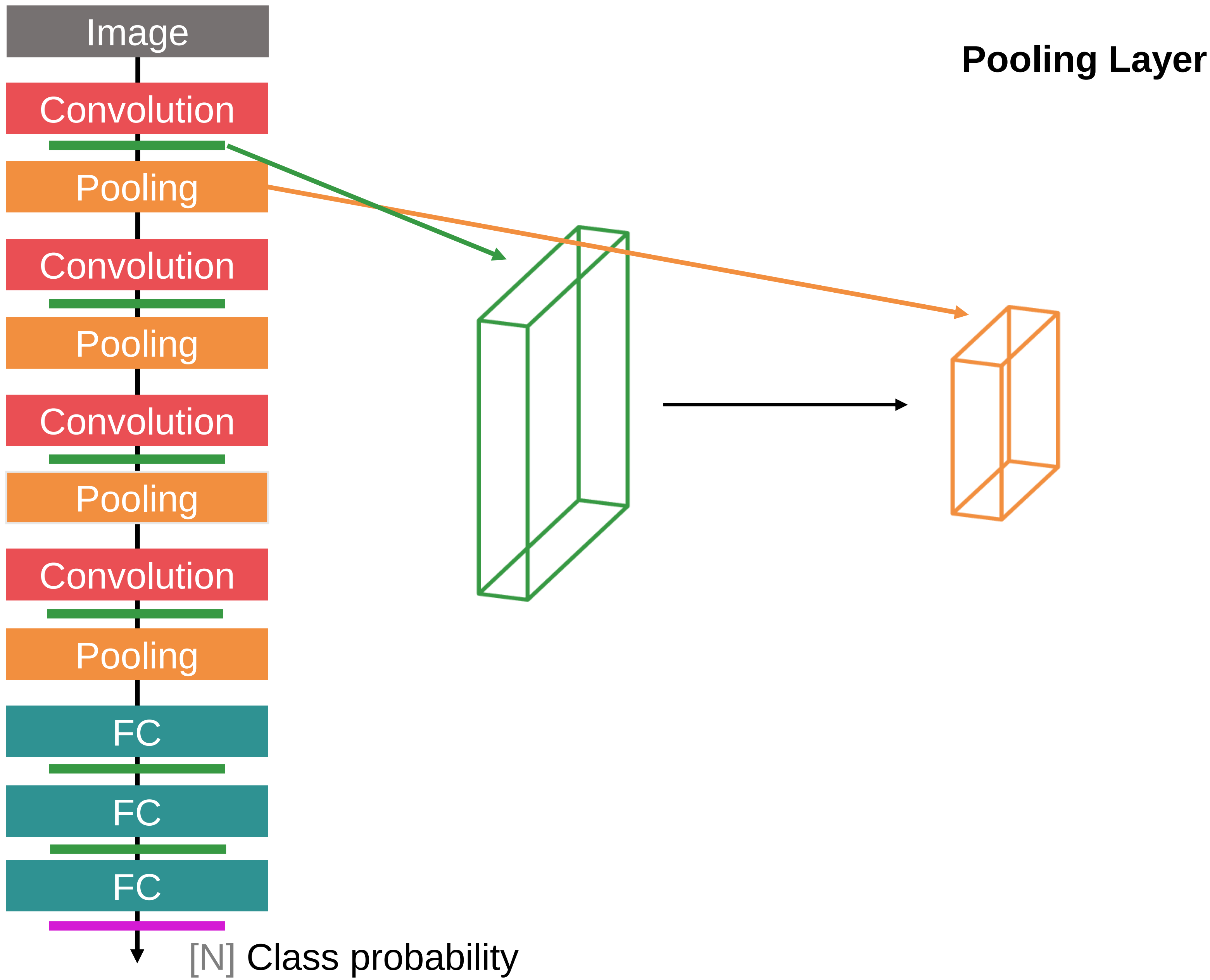
Convolutional Neural Networks (CNN) – Recap



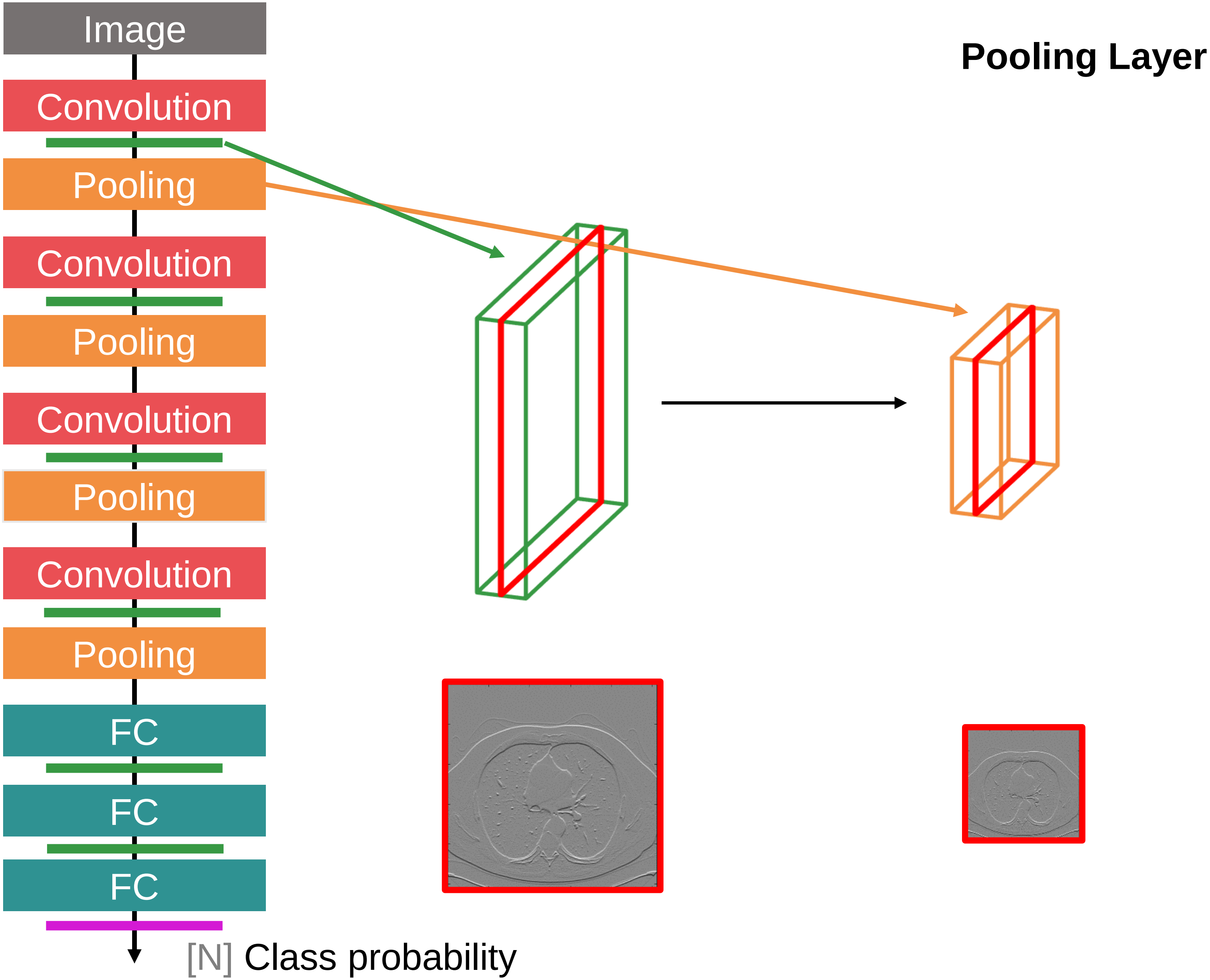
Convolutional Neural Networks (CNN) – Recap



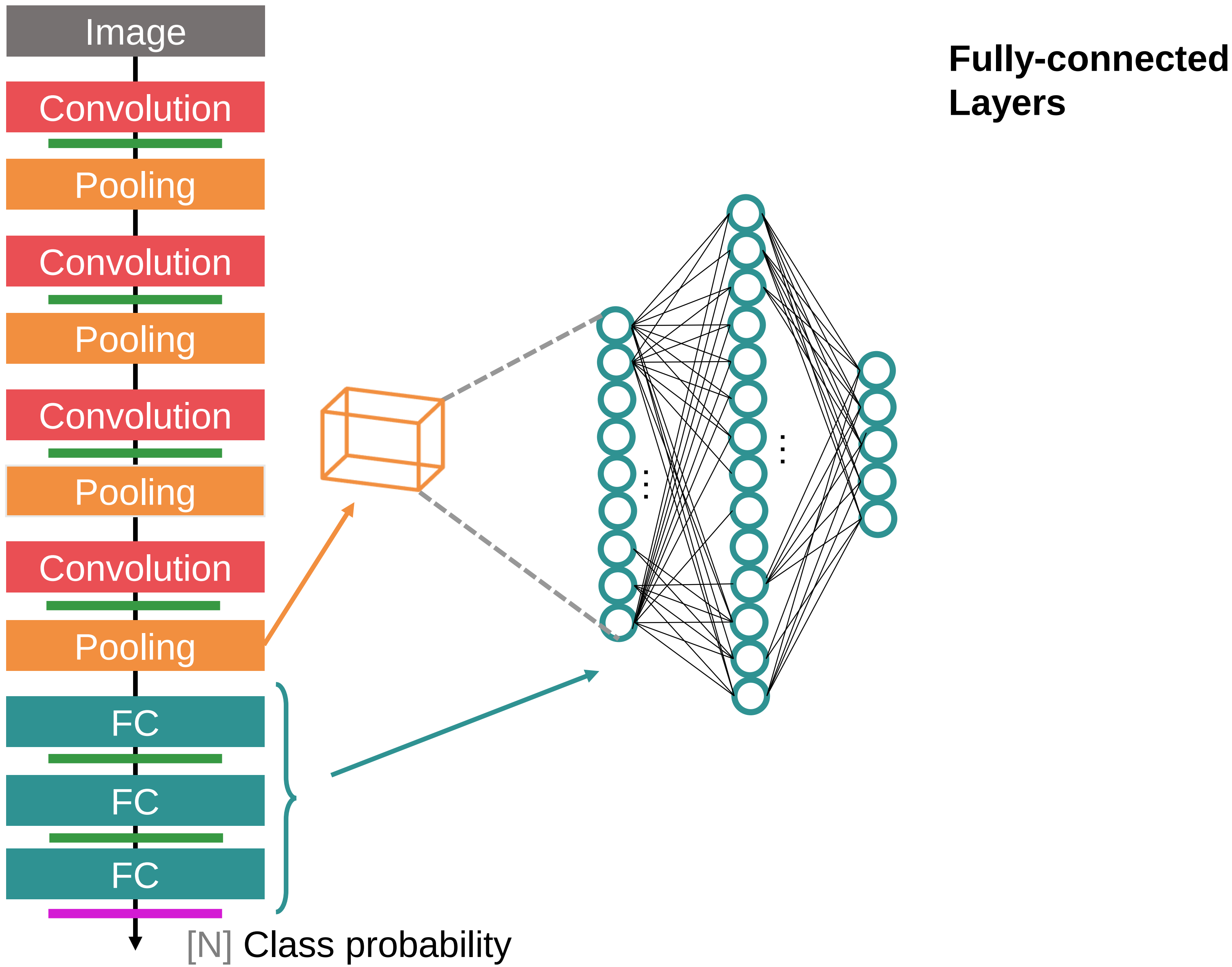
Convolutional Neural Networks (CNN) – Recap



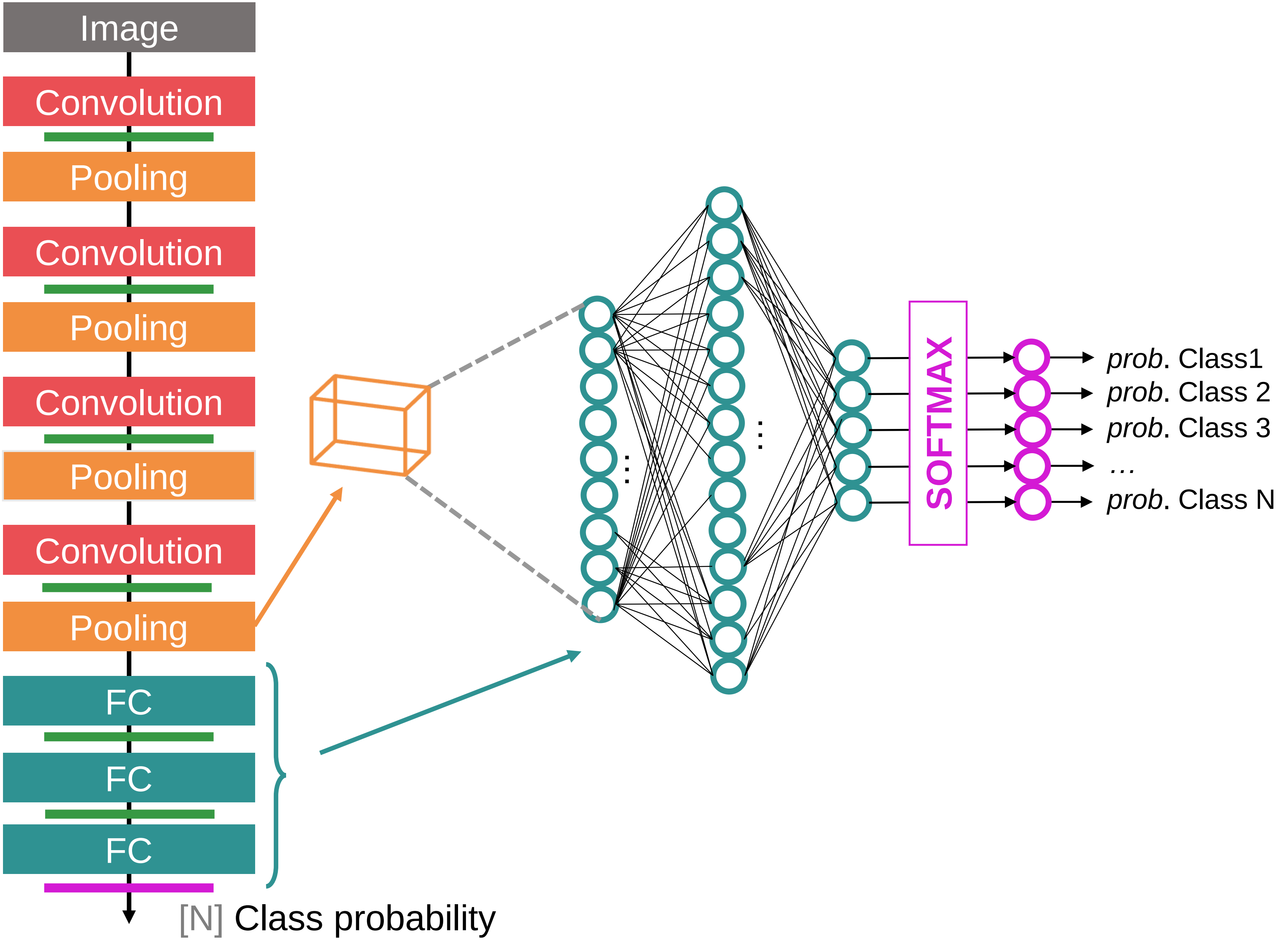
Convolutional Neural Networks (CNN) – Recap



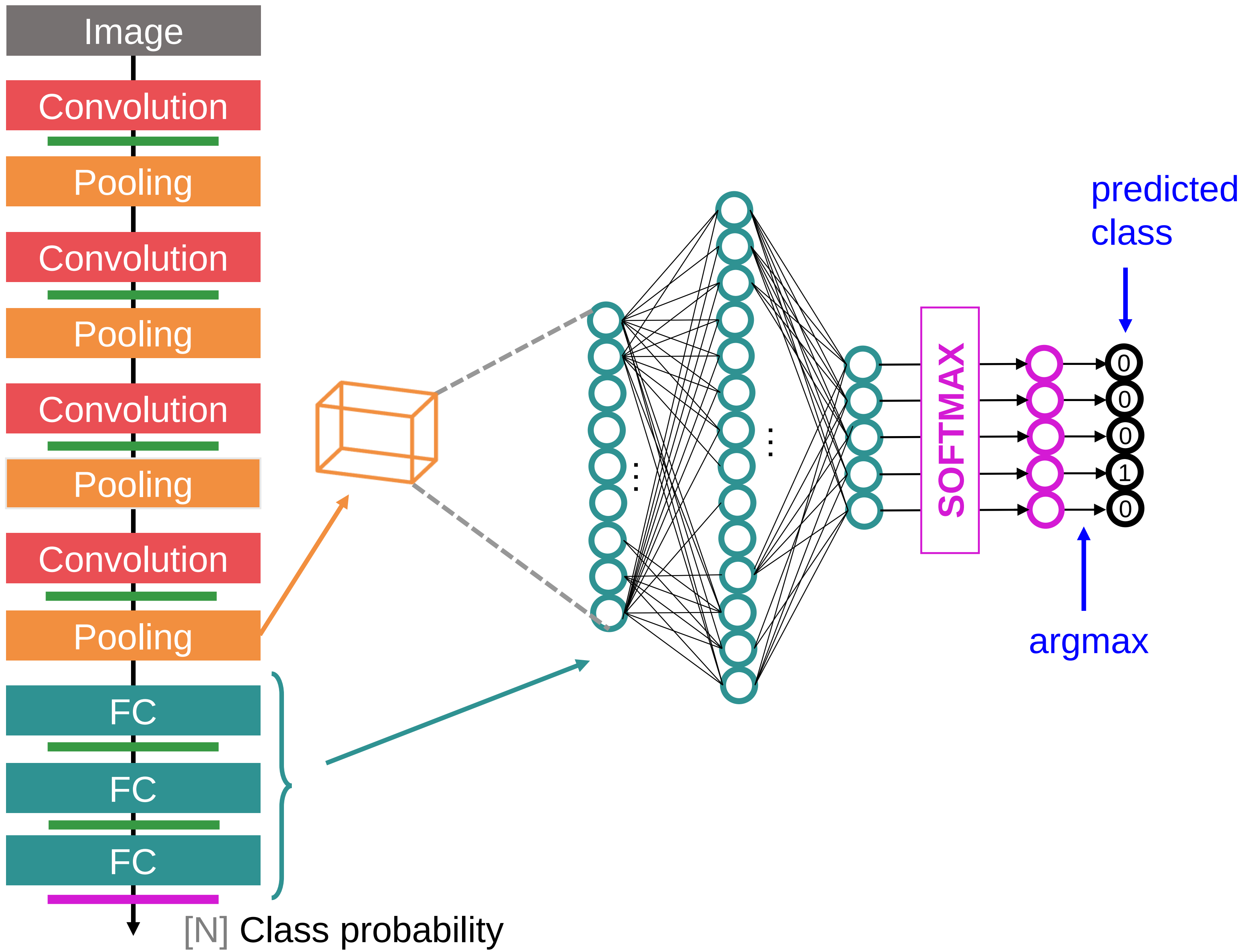
Convolutional Neural Networks (CNN) – Recap



Convolutional Neural Networks (CNN) – Recap



Convolutional Neural Networks (CNN) – Recap



Training – Loss function

- We have defined a function $f(x)$ from the pixel values (images) to class probabilities. This function is parametrized by a set of weights W (weights from the convolutional and fully-connected layers).
- We want to set them so that the predicted class scores are consistent with the ground truth lables in the training data.
- We are going to measure the error (difference between out predicted class S and the ground truth L) with a **loss function** (or sometimes also named the **cost function** or the **objective**). Intuitively, the loss will be high if we are doing a poor job of classifying the training data, and it will be low if we are doing well.
- One simple measurement could be the **mean squared error** (**MSE**):

$$D(S,L) = \frac{1}{n}\sum_{i=1}^{n}(S_i - L_i)^2$$

- A more sophisticated measurement is **the cross entropy**:

$$D(S,L) = -\sum_{i} L_i \log(S_i)$$

Training – Optimization

– We have to come up with a way of efficiently finding the parameters W that minimize the loss function. This is a **optimization** problem.

– It can be solved as a iterative process:

- Set initial values for the parameters (W).
- Execute the model.
- Evaluate the objective function, i.e., compare the predicted probabilities with the original labels, e.g:

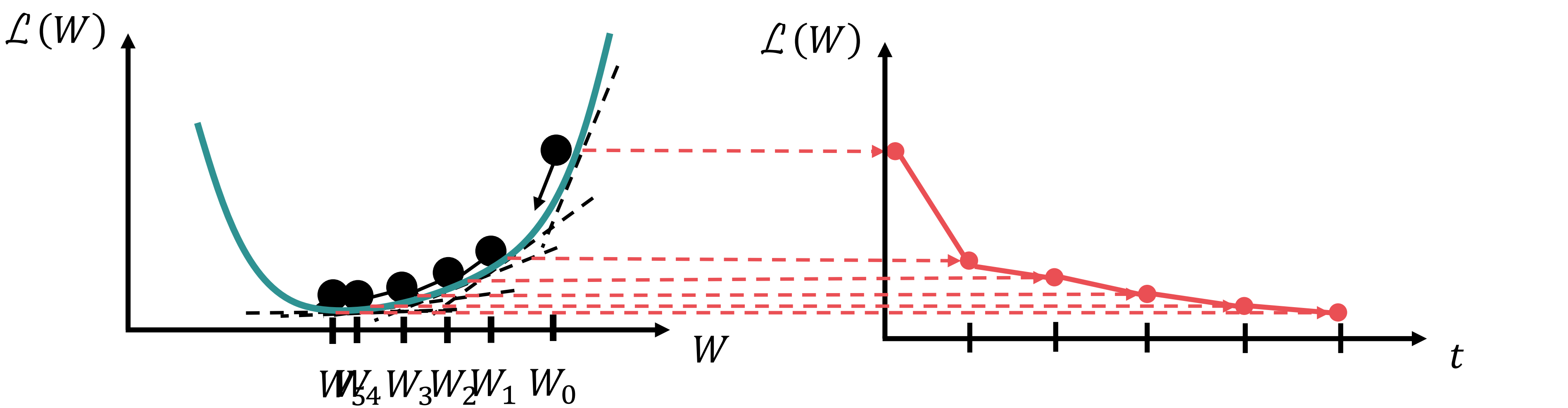
$$\mathcal{L}(W) = \frac{1}{N} \sum_i D(S_i, L_i)$$

- Stop if stopping criterion is satisfied, e.g., $\mathcal{L}(W)$ is not decreasing anymore.
- Else, adjust the values of parameters in the model according to some strategy.
- Execute the new model.

Optimization strategies

- A first very bad idea solution is **random search**.
- Another strategy is **Gradient Descent (GD)**.
  - Simple gradient method
  - Search for a local minimum based on the first derivatives of the objective function.
  - Follows the direction where objective function decreases most quickly which is opposite to the gradient of the objective function.
  - Takes a small step (learning rate: r) along the direction of the gradient.

$$W_{k+1} = W_k - r \bigtriangleup \mathcal{L}(W_k)$$



Optimization strategies

- The standard gradient descent algorithm (GD) evaluate the cost and gradient **over the full training set**.

  - for iteration:
    - for each weight W:

      $W_{k+1} = W_k - r \vartriangle \mathcal{L}(W_k)$, where $\vartriangle \mathcal{L}(W_k) = \sum_i D(S_i, L_i)(-W_k)$

- In case of very large datasets, using GD can be very costly since we are only taking a single step for one pass over the training set.

- In **Stochastic Gradient Descent (SGD)** we don't accumulate the weight updates. Instead, we update the weights **after each training sample.**

  - for iteration:
    - for training simple i:
      - for each weight W:

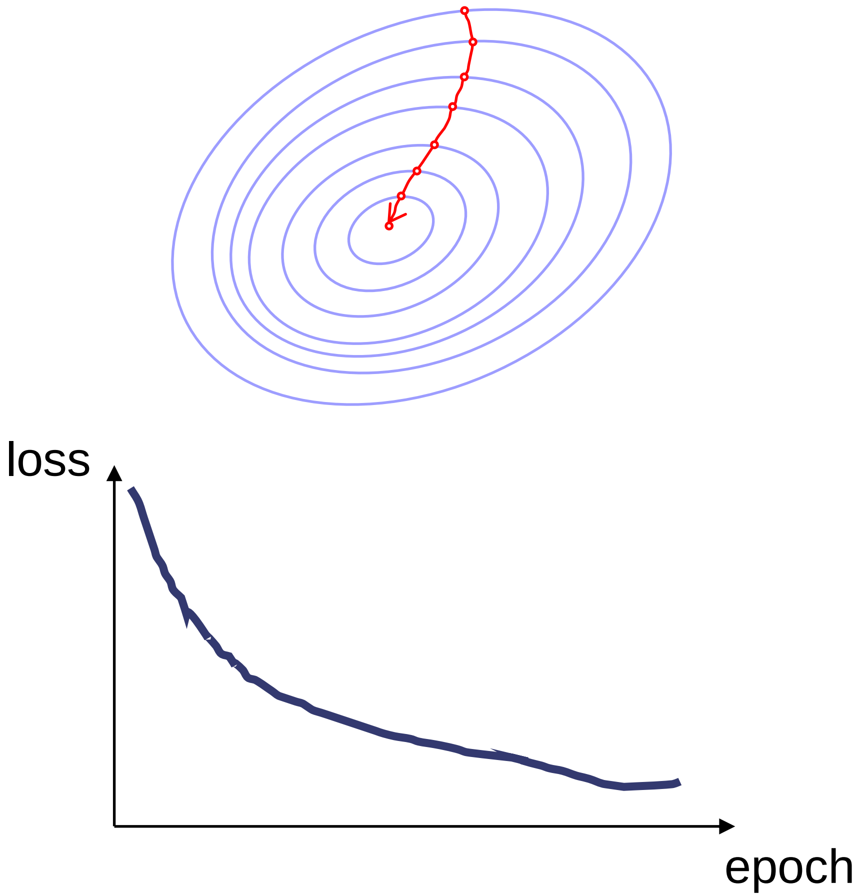
        $W_{k+1} = W_k - r \vartriangle \mathcal{L}(W_k)$, where $\vartriangle \mathcal{L}(W_k) = D(S_i, L_i)(-W_k)$

- The gradient based on a single training sample is a "stochastic approximation" of the "true" cost gradient.

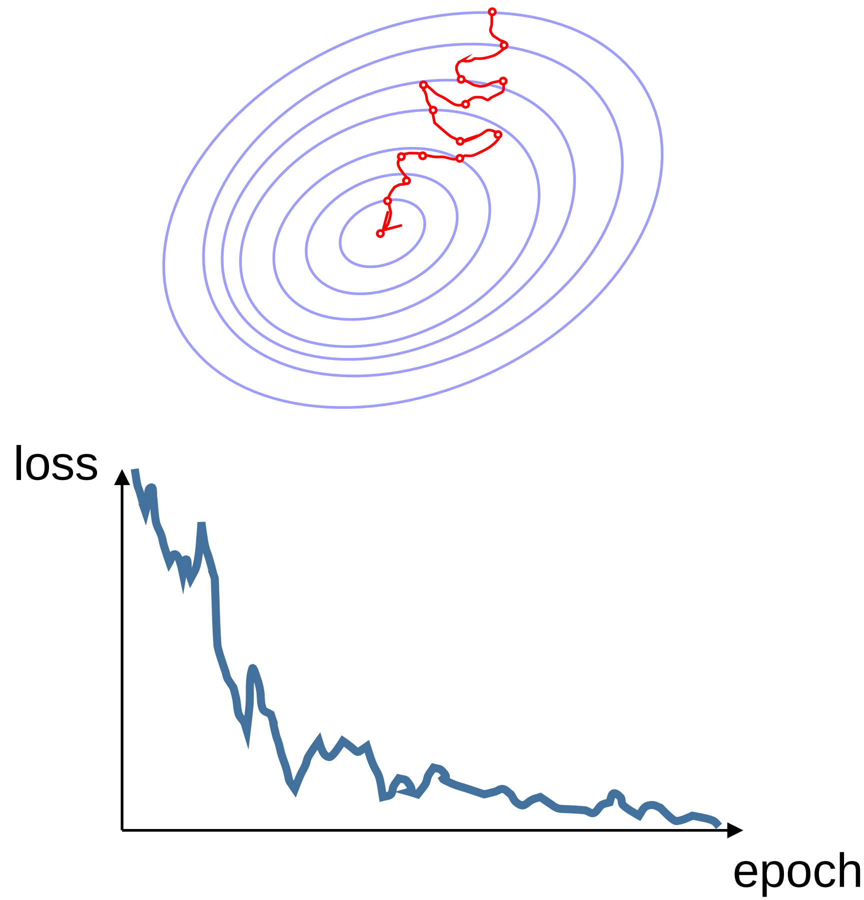
- SGD often converges much faster compared to GD. Also, SGD helps the algorithm to skip some local minima.

Optimization strategies

Gradient Descent
(**GD**)



Stochastic Gradient Descent
(**SGD**)

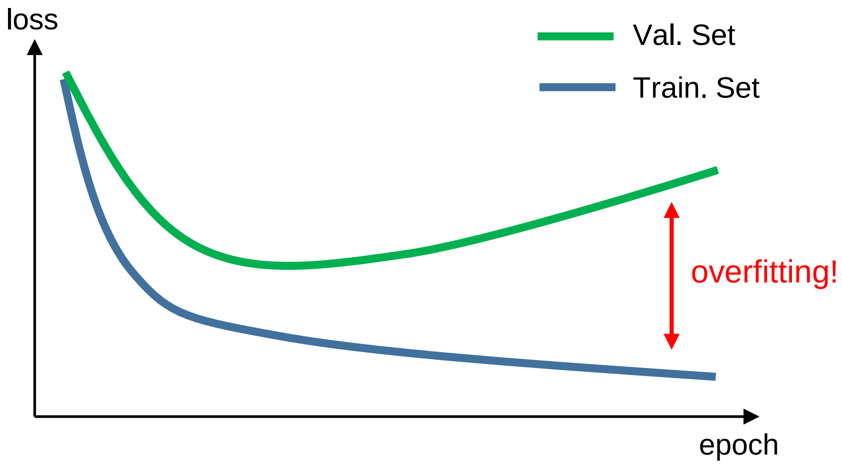


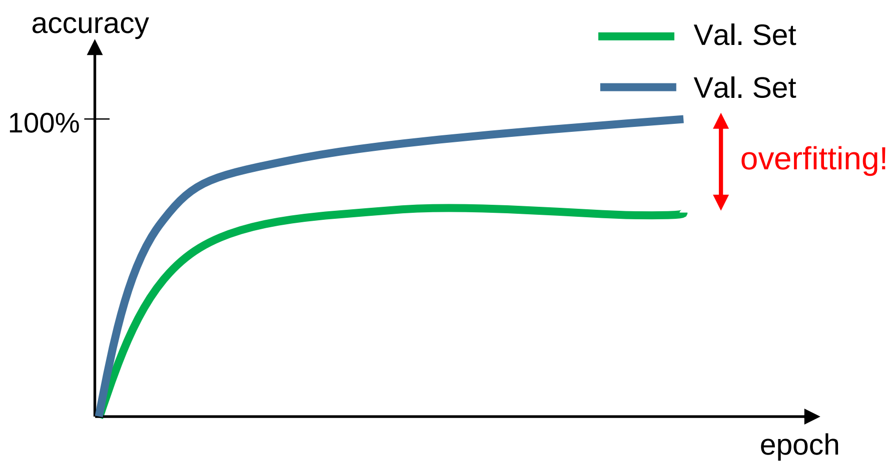
– **Backpropagation** is the algorithm that computes in a computationally efficient way the gradients of the loss function. Is based on the chain rule from multivariable calculus.

Overfitting

– When training deep networks and other complex networks of predictors, the risk of **overfitting** is typically of large concern.

– Overfitting is observed when a high capacity model (such as a CNN) performs very well on training data but poorly when new data is presented. The network has memorized the training examples, but it has not learned to generalize to new situations.





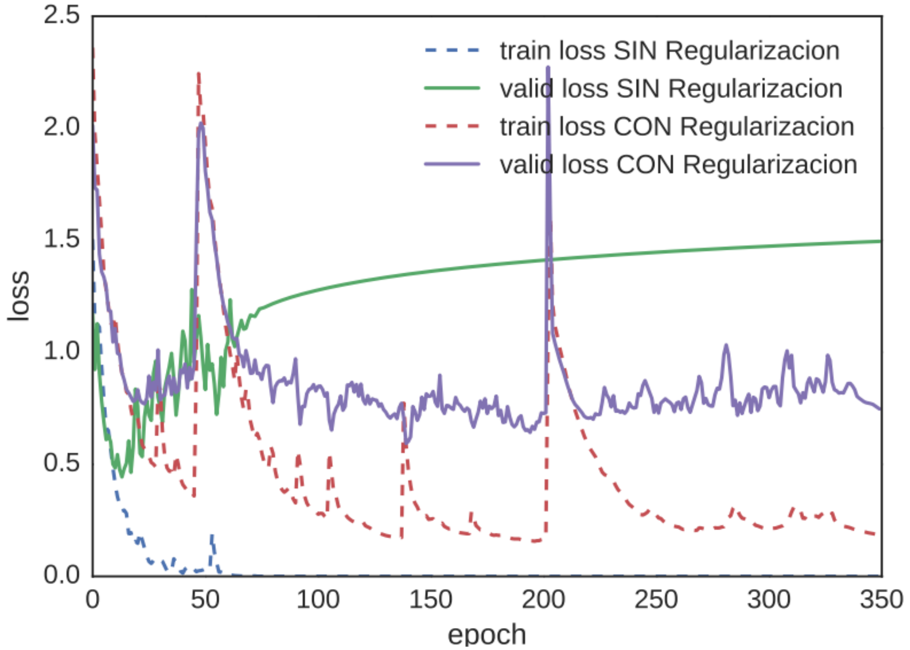
– We can apply different techniques to prevent the overfitting:
- L2 Regularization
- Early Stopping
- Data Augmentation
- Dropout

Overfitting – L2 Regularization

- **L2 regularization** can be implemented by penalizing the squared magnitude of all parameters directly in the loss function. That is, for every weight W in the network, we add the term ${}^{1}/_{2}\,\lambda W^2$ to the loss function, where λ is the regularization strength.
- The L2 regularization has the interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors

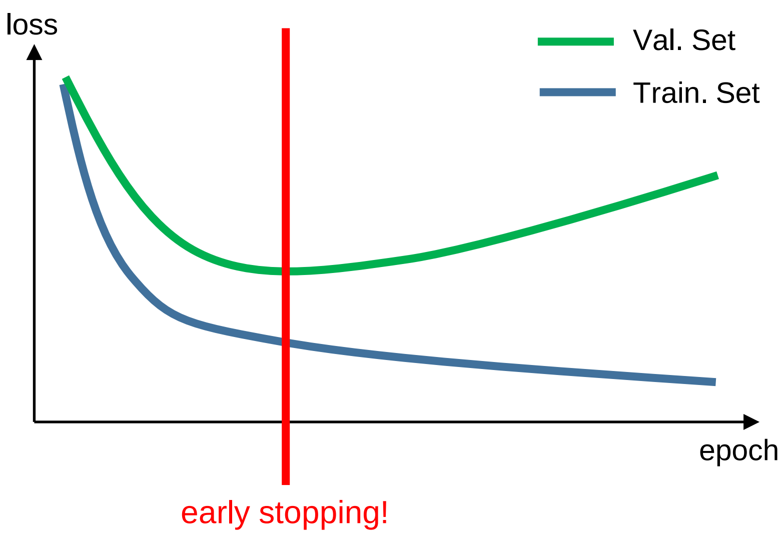
$$\mathcal{L}_R = \underbrace{\mathcal{L}}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

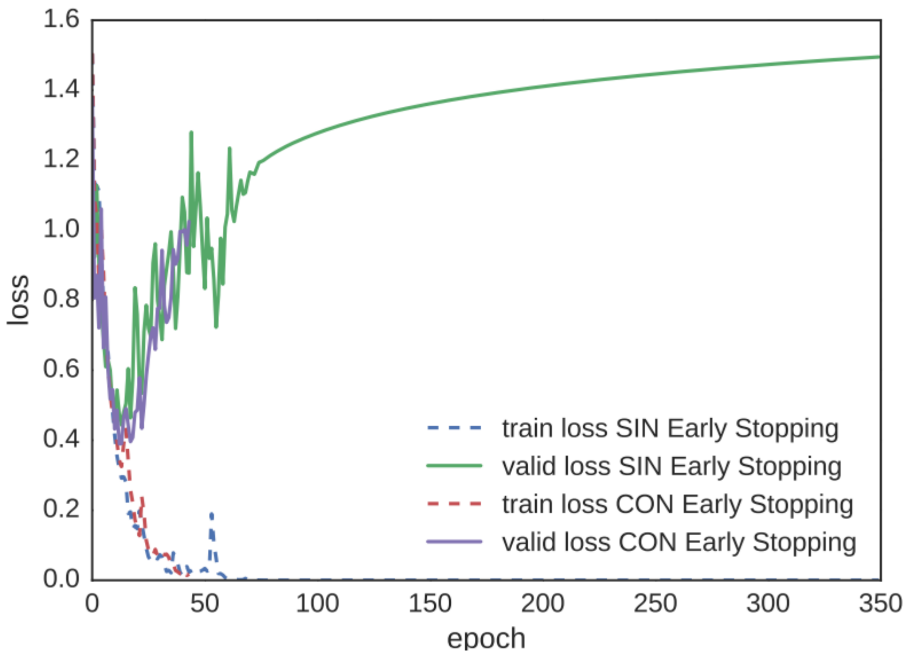
$$R(W) = \frac{1}{2} W^2$$



Overfitting – Early Stopping

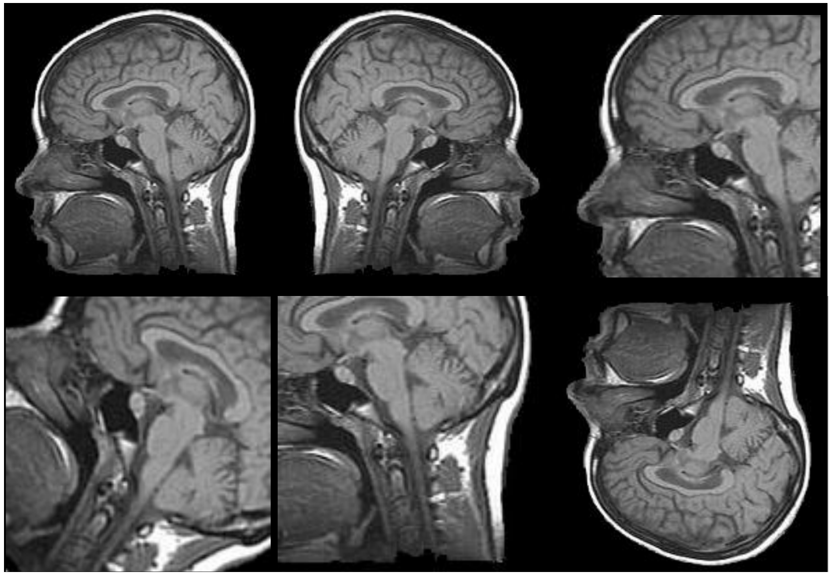
- When **Early Stopping** is used, the error on the validation set is monitored during the training process.
- The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to rise.
- When the validation error increases for a specified number of iterations, the training is stopped.

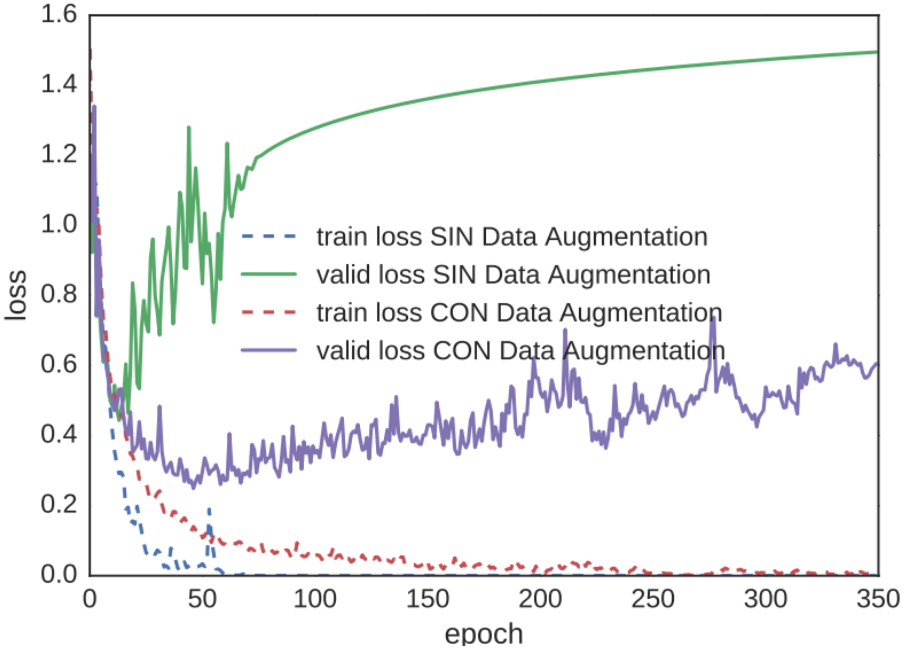




Overfitting – Data Augmentation

- The greater the size of the dataset, the higher the variability in the data, thus the lower the overfitting.
- But we have a limited dataset. **Data augmentation** increase the dataset size by artificially creating data training samples during the training.
  - Random crops on the original image
  - Translations
  - Rotations
  - Horitzontal reflection
  - …

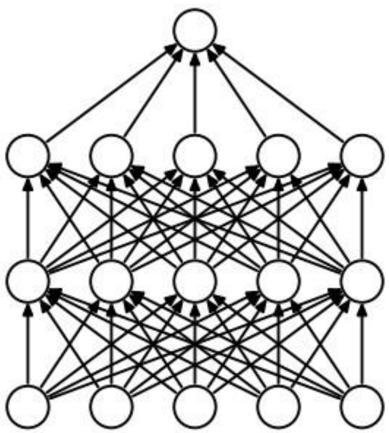




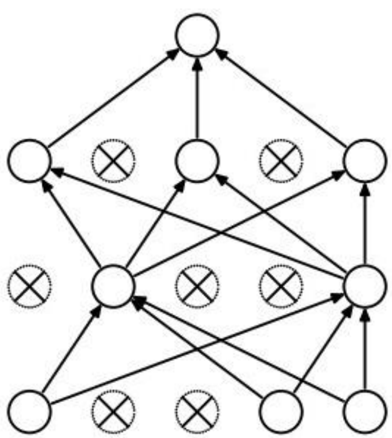
Overfitting – Dropout

– **Dropout** is an extremely effective, simple and recently introduced regularization technique. While training, dropout is implemented by only keeping a neuron active with some probability *p*, or setting it to zero otherwise.

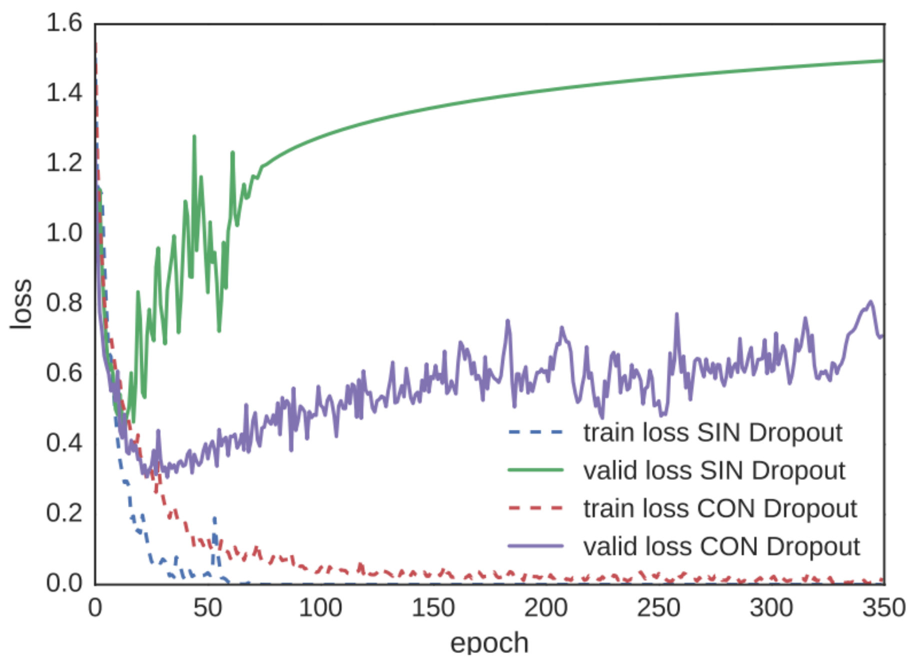
[Srivastava et al., 2014]



(a) Standard Neural Net



(b) After applying dropout.



Deep Learning software

Software links

1.[**Theano**] – CPU/GPU symbolic expression compiler in python (from MILA lab at University of Montreal)

2.[**Torch**] – provides a Matlab-like environment for state-of-the-art machine learning algorithms in lua (from Ronan Collobert, Clement Farabet and Koray Kavukcuoglu)

3.[**Pylearn2**] - Pylearn2 is a library designed to make machine learning research easy.

4.[**Blocks**] - A Theano framework for training neural networks

5.[**Tensorflow**] - TensorFlow™ is an open source software library for numerical computation using data flow graphs.

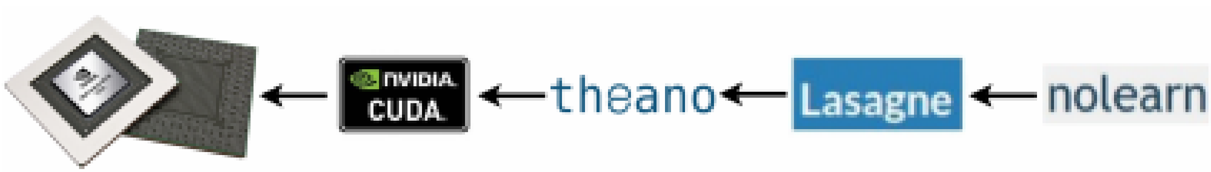
6.[**MXNet**] - MXNet is a deep learning framework designed for both efficiency and flexibility.

7.[**Caffe**] -Caffe is a deep learning framework made with expression, speed, and modularity in mind.Caffe is a deep learning framework made with expression, speed, and modularity in mind.

8.[**Lasagne**] - Lasagne is a lightweight library to build and train neural networks in Theano.

9. [**Nolearn**]

10. etc



Introductory tutorials (nolearn.lasagne)

- Two introductory tutorials exist for *nolearn.lasagne*:
  - Training convolutional neural networks with nolearn
  - Using convolutional neural nets to detect facial keypoints tutorial with code
- Finally, there's a few presentations and examples from around the web. Note that some of these might need a specific version of nolearn and Lasange to run:
  - Oliver Dürr's Convolutional Neural Nets II Hands On with code
  - Roelof Pieters' presentation Python for Image Understanding comes with nolearn.lasagne code examples
  - Benjamin Bossan's Otto Group Product Classification Challenge using nolearn/lasagne
  - Kaggle's instructions on how to set up an AWS GPU instance to run nolearn.lasagne and the facial keypoint detection tutorial
  - An example convolutional autoencoder
  - Winners of the saliency prediction task in the 2015 LSUN Challenge have published their lasagne/nolearn-based code.

Source: https://pythonhosted.org/nolearn/lasagne.html

Code example

```
network =  NeuralNet(
    layers = [
    (InputLayer, {'shape': (None, 1, 32, 32)}),
    (Conv2DLayer, {'num_filters': 48, 'filter_size': 3}),
    (MaxPool2DLayer, {'pool_size': 2}),
    (Conv2DLayer, {'num_filters': 32, 'filter_size': 3}),
    (Conv2DLayer, {'num_filters': 64, 'filter_size': 3}),
    (MaxPool2DLayer, {'pool_size': 2}),
    (Conv2DLayer, {'num_filters': 64, 'filter_size': 2}),
    (MaxPool2DLayer, {'pool_size': 2}),
    (DropoutLayer, {'p':0.5}),
    (DenseLayer, {'num_units': 64}),
    (DenseLayer, {'num_units': 112}),
    (DropoutLayer, {'p':0.5}),
    (DenseLayer, {'num_units': 5, 'nonlinearity': softmax}),
    ],
    update=sgd,
    update_learning_rate=0.01,
    max_epochs=350,
    train_split=TrainSplit(eval_size=0.25),
    objective=categorical_cross_entropy,
    objective_lambda2=0.0025,
    on_epoch_finished=[
        EarlyStopping(patience=30),
        ],
    )
network.fit(X_train,y_train)
predictions=network.predict(X_test)
score=accuracy_score(predictions,Y_test)
```

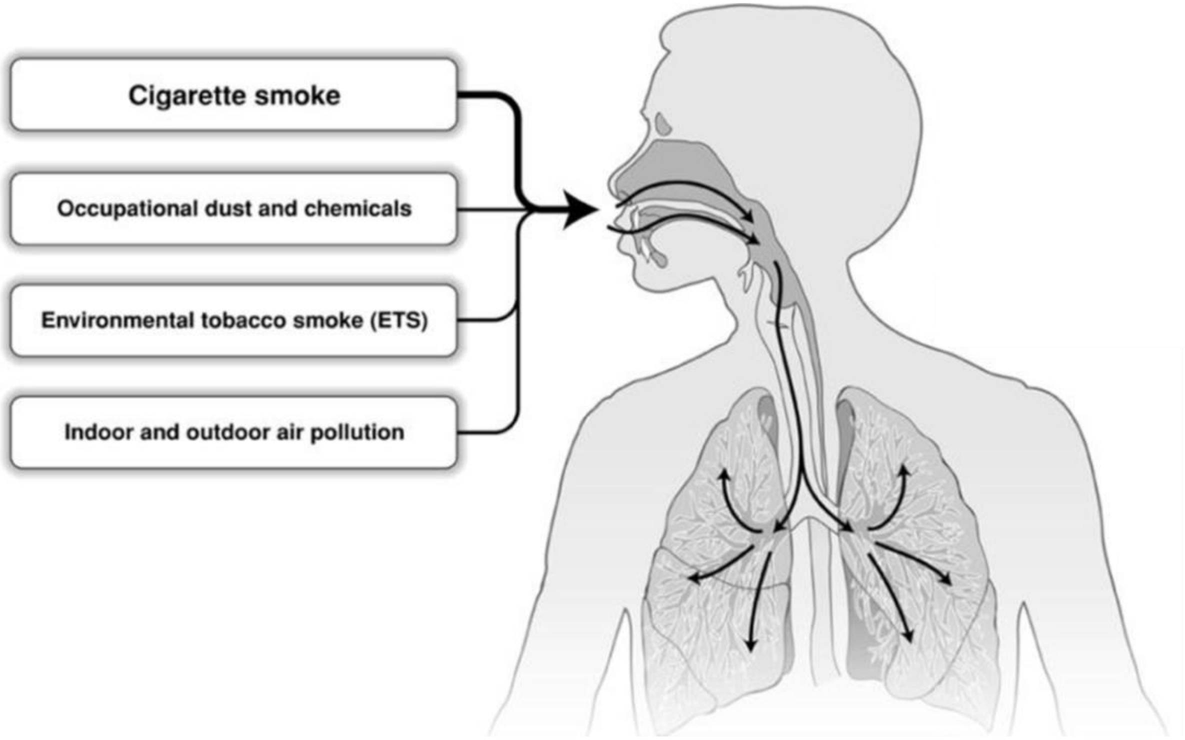
Case study.

Comparative study of machine learning and deep learning techniques for lung images classification.

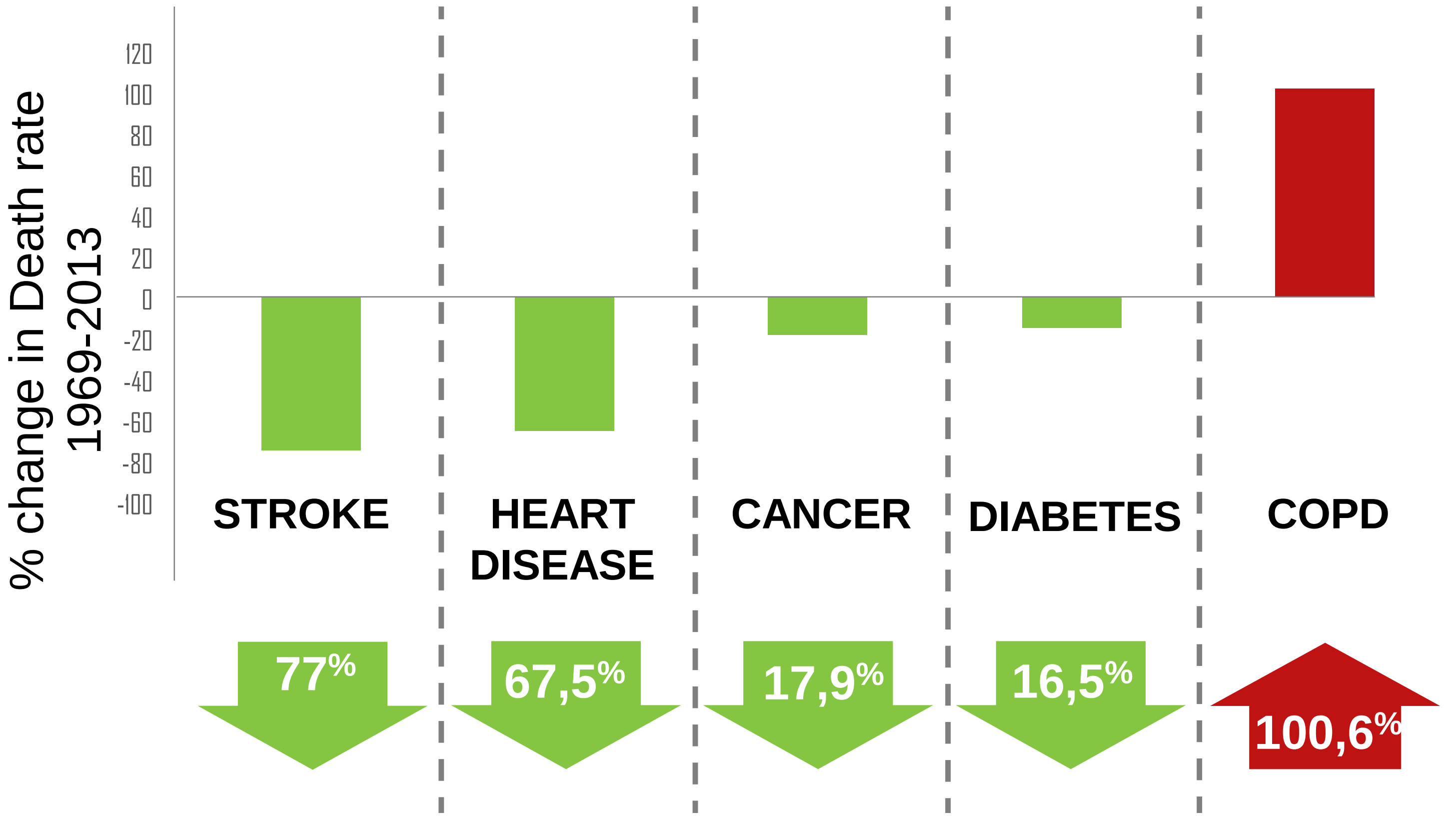
COPD: Chronic Obstructive Pulmonary Disease

Definition: persistent airflow limitation. The airflow limitation is usually progressive and associated with an abnormal inflammatory response.

Causes:

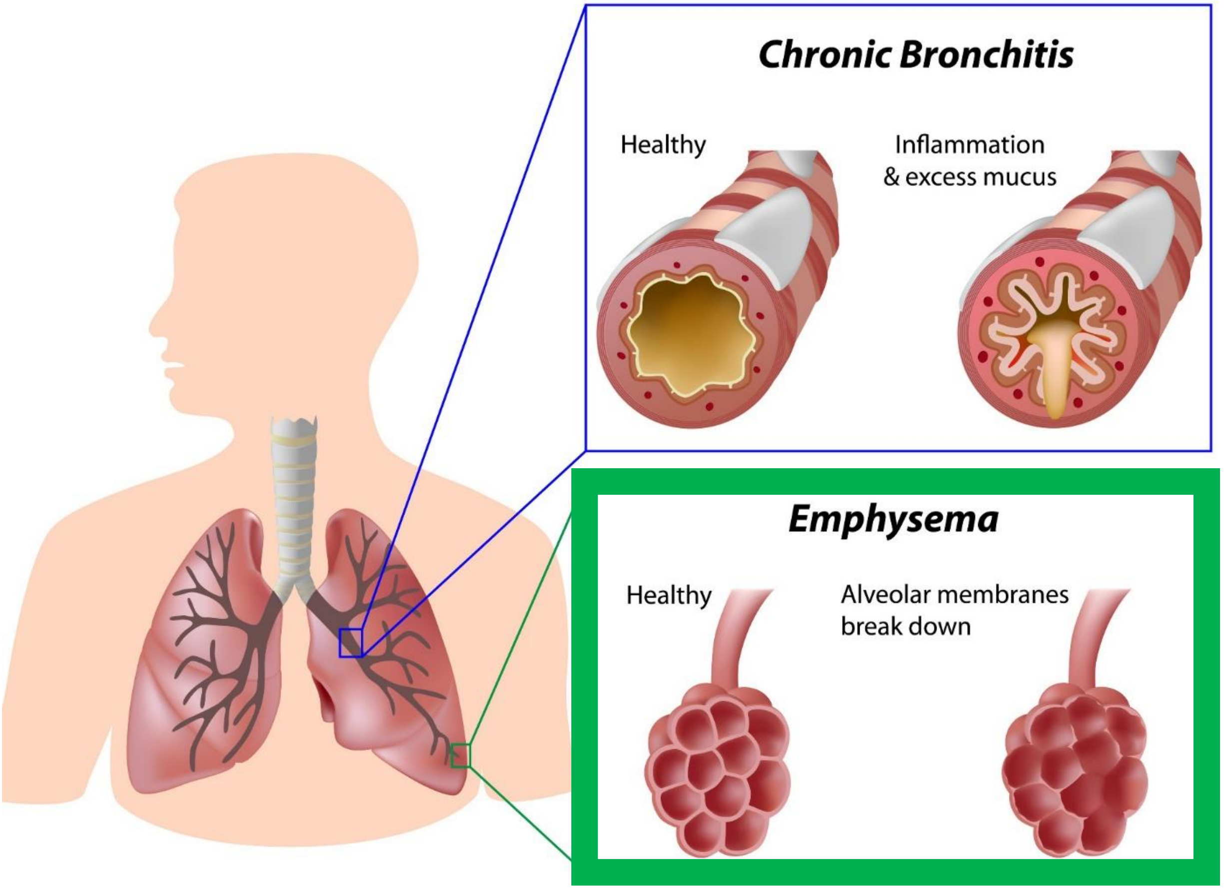


COPD: a growing disease

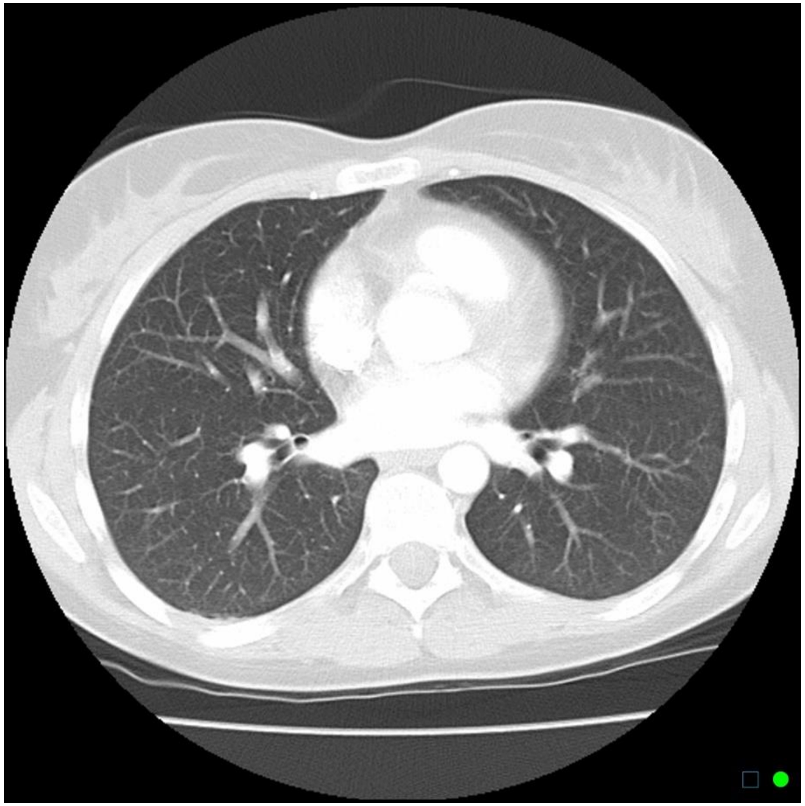


Source: JAMA 2015 “Temporal Trends in Mortality in the United States, 1969-2013”

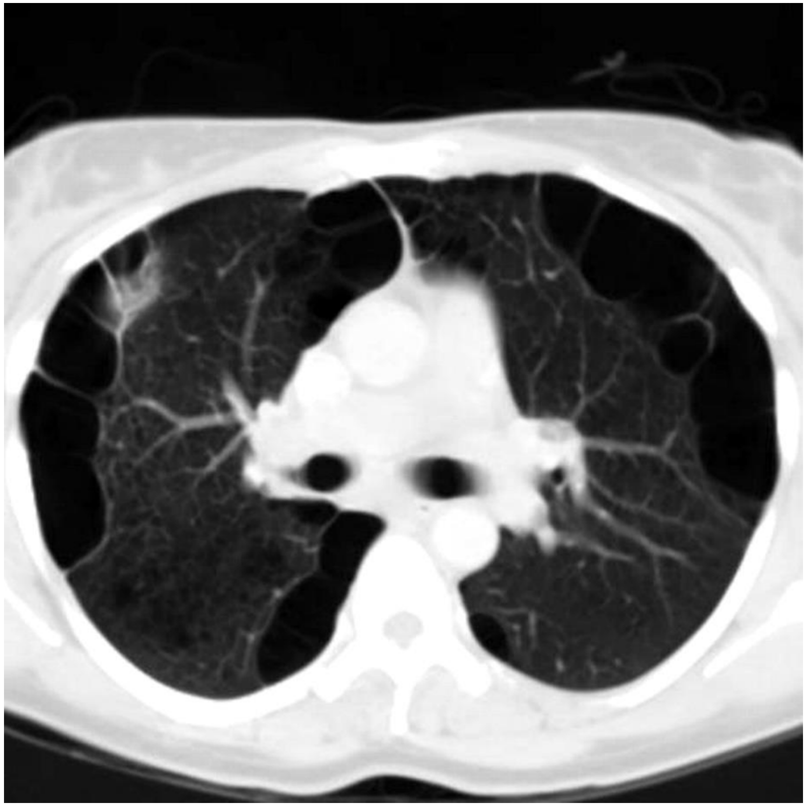
COPD: emphysema and chronic bronchitis



COPD: emphysema and chronic bronchitis



Healthy



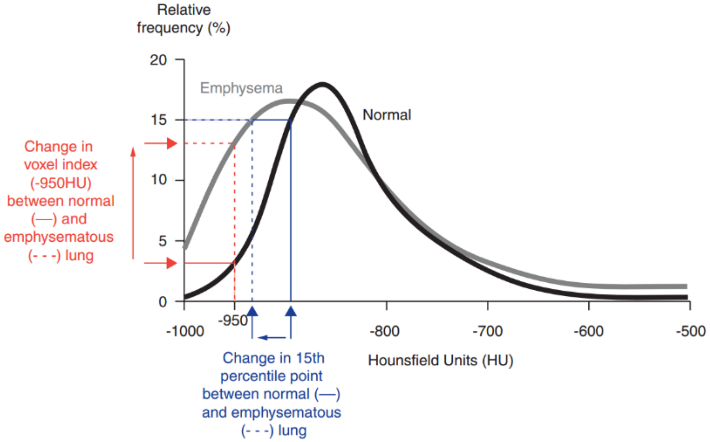
Emphysema

Emphysema: quantification

– Changes in the pulmonary density, hence the emphysema progression, can be measured and quantified using Computed Tomography (CT).

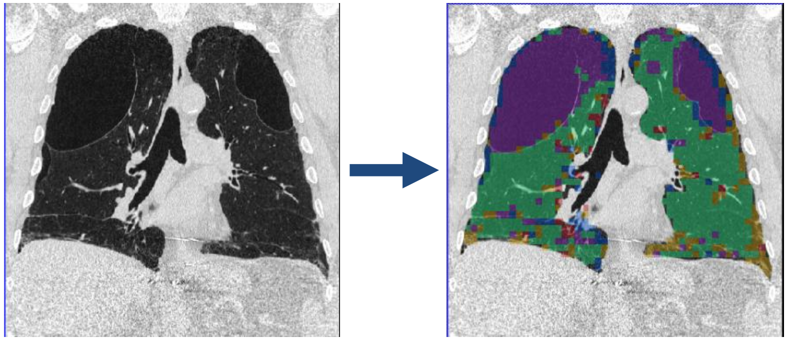
– CT densitometric analysis in Chest CT:

  - Widely accepted as measurement of emphysema.
    Low Attenuetion Areas % (LAA%)
    15th Percentile



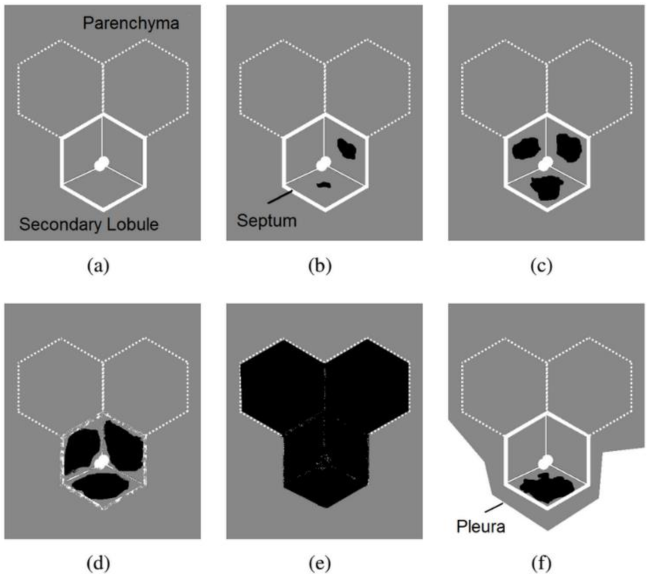
– Others methods based on local information emphysema:

  - Emphysema classification

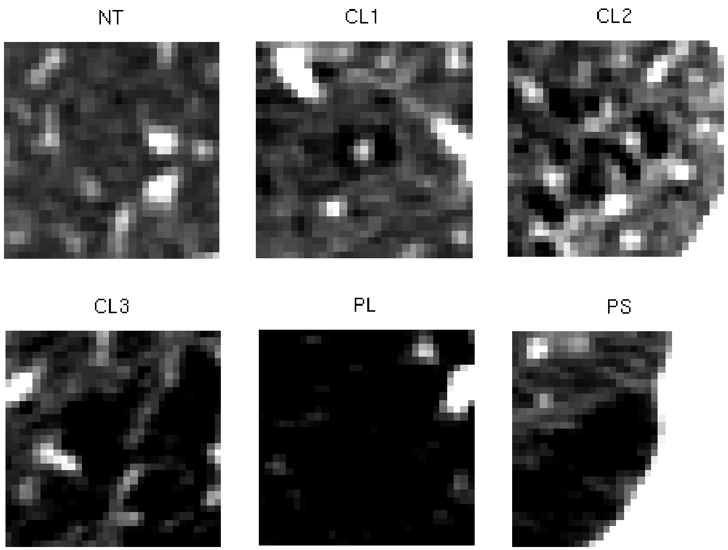


Emphysema classification

– There exist different radiologic patterns of emphysematous tissue.
  - Normal tissue (NT)
  - Paraseptal emphysema (PS)
  - Panlobular emphysema (PL)
  - Mild centrilobular emphysema (CL1)
  - Moderate centrilobular emphysema (CL2)
  - Severe centrilobular emphysema (CL3)

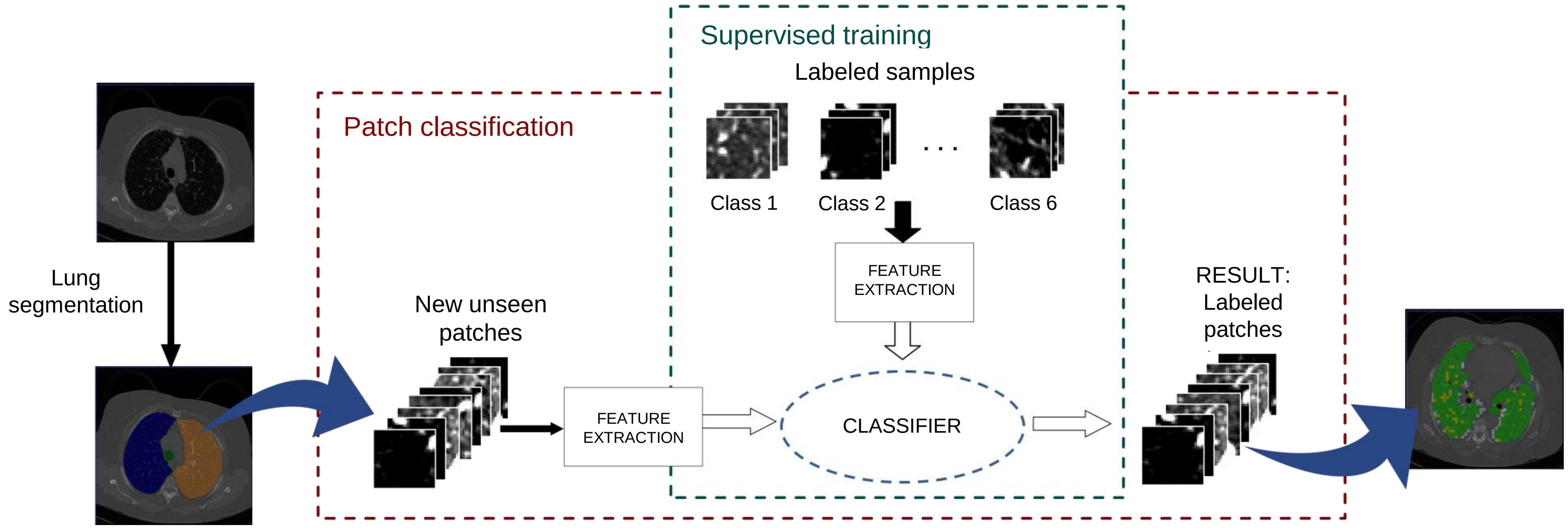


(a) NT. (b) CL1. (c) CL2. (d) CL3 (e) PL. (f) PS.



Examples of regions of all patterns to classify

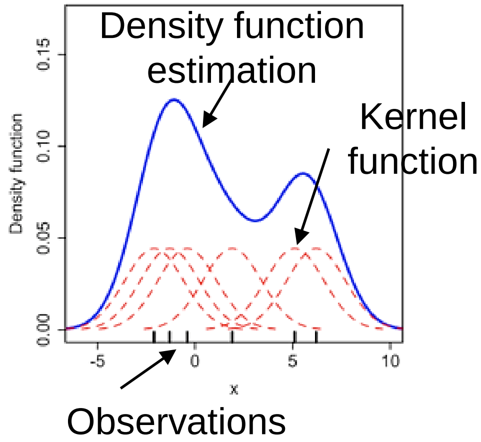
Emphysema classification



MACHINE LEARNING

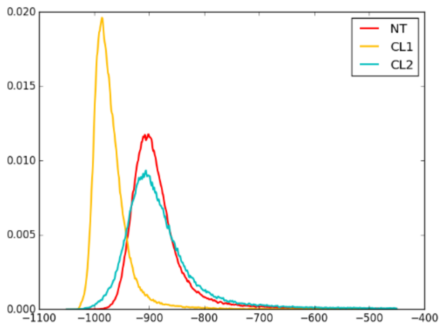
Feature extraction and classification

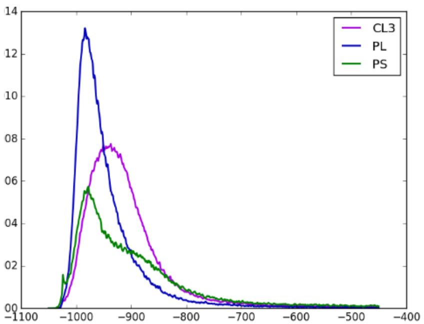
- **Local intensity probability distributions functions**, estimated by Kernel Density Estimation (KDE).



$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h\left(x - x_i\right)$$

- This feature has enough discriminative power in emphysema classification problem.

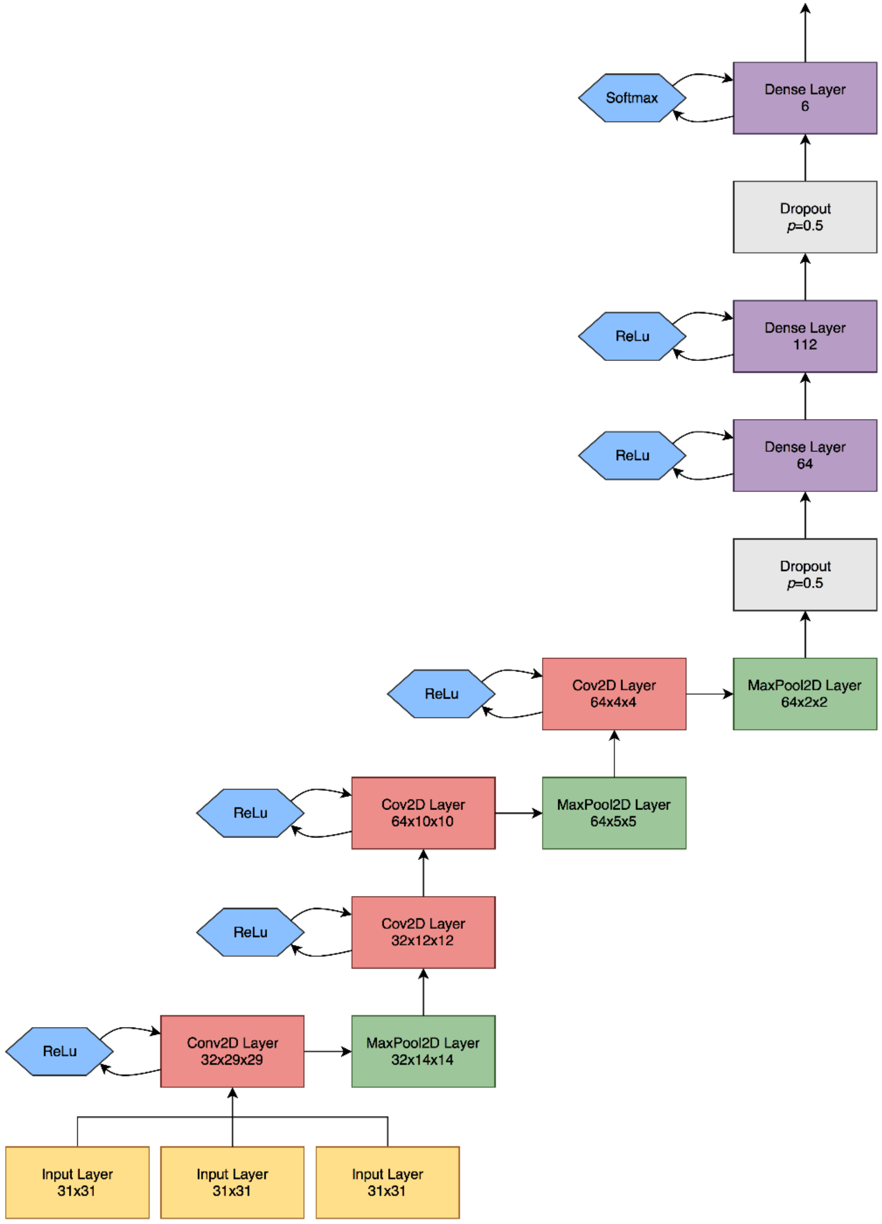




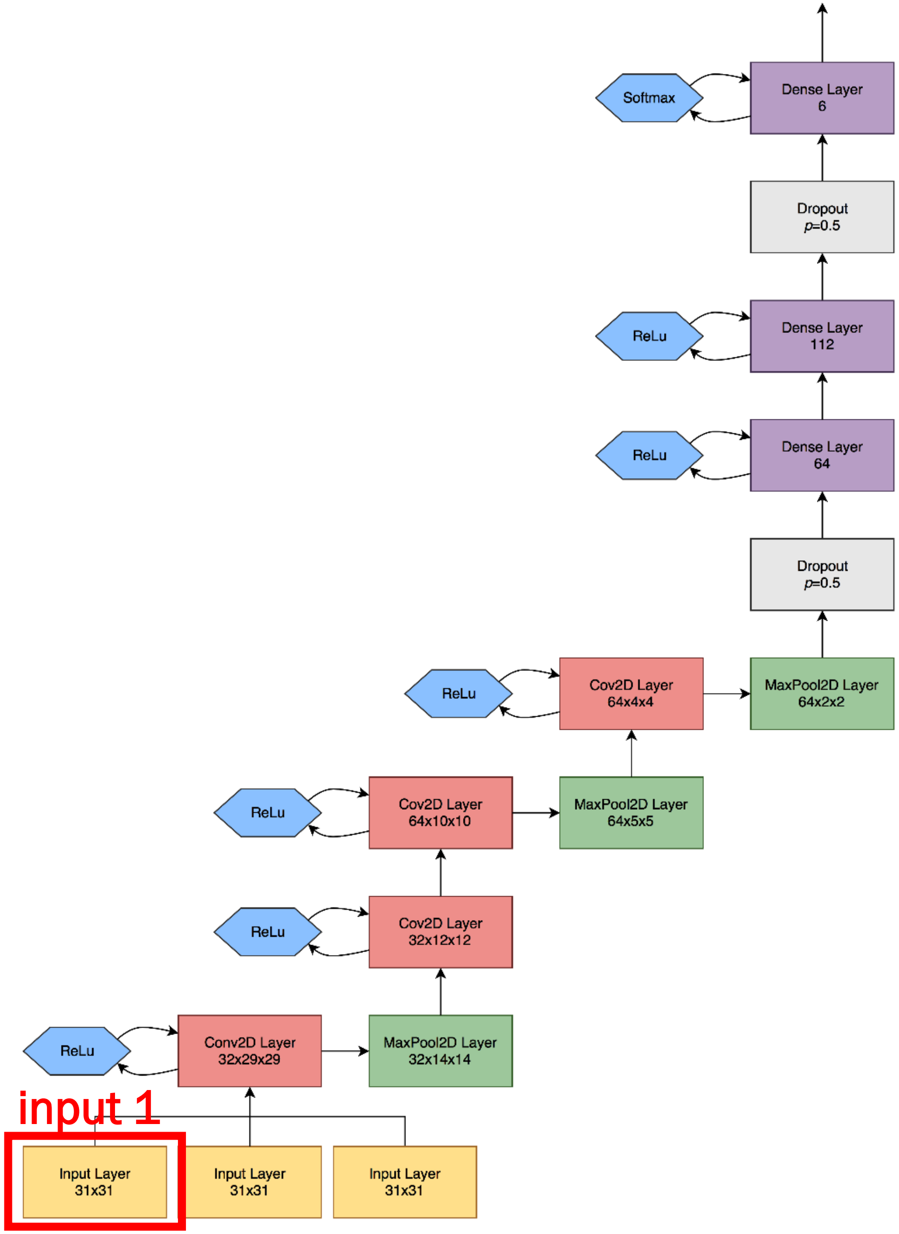
- We use the **KNN classifier**.

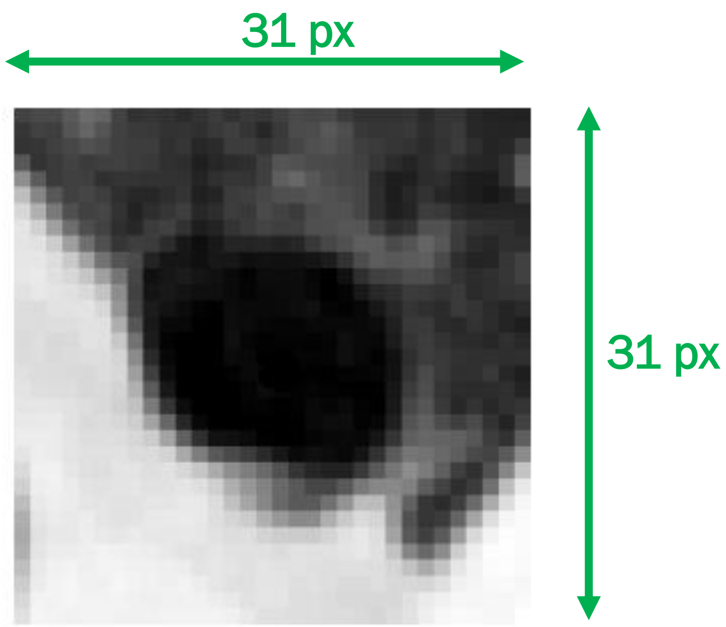
DEEP LEARNING

Multiscale Convolutional Neural Network (M-CNN)

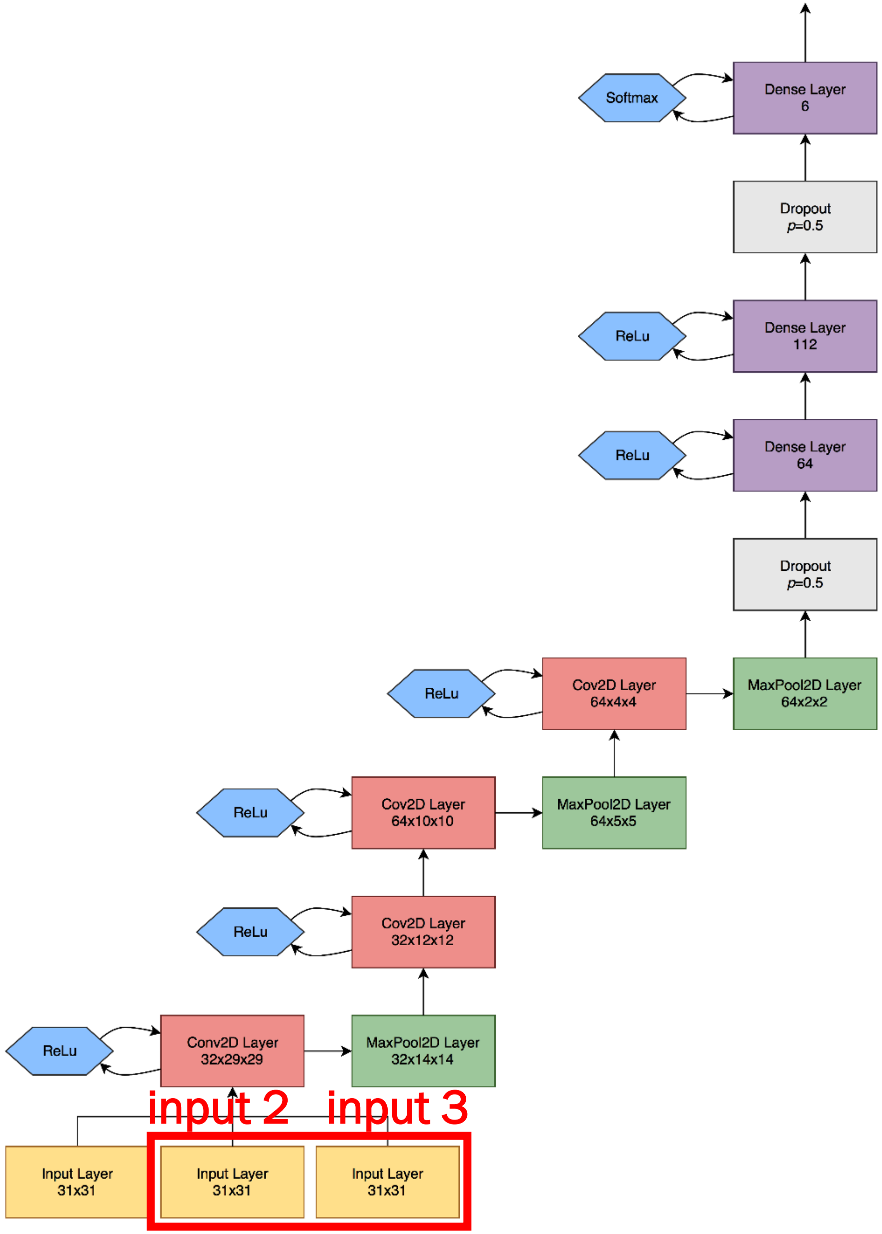


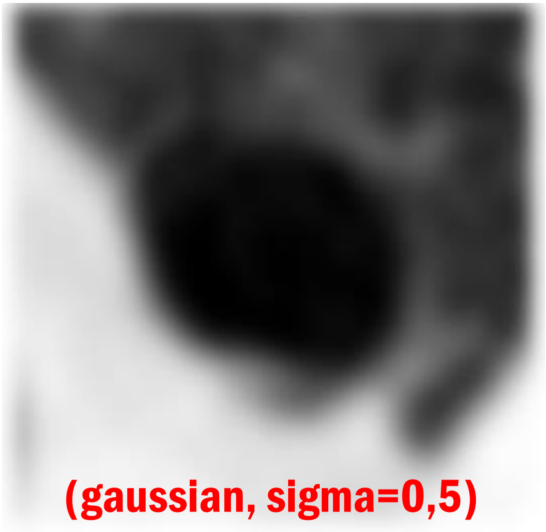
Multiscale Convolutional Neural Network (M-CNN)

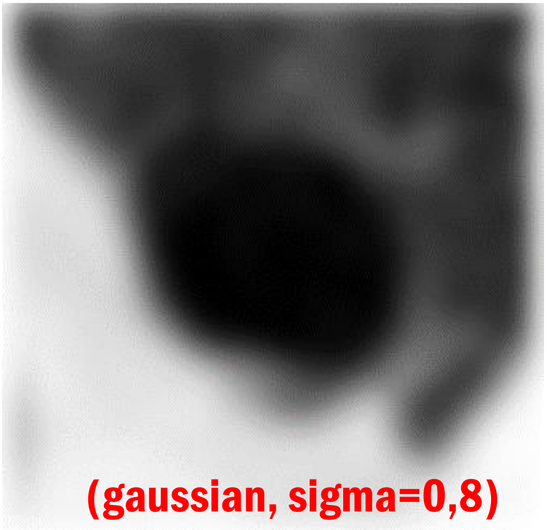




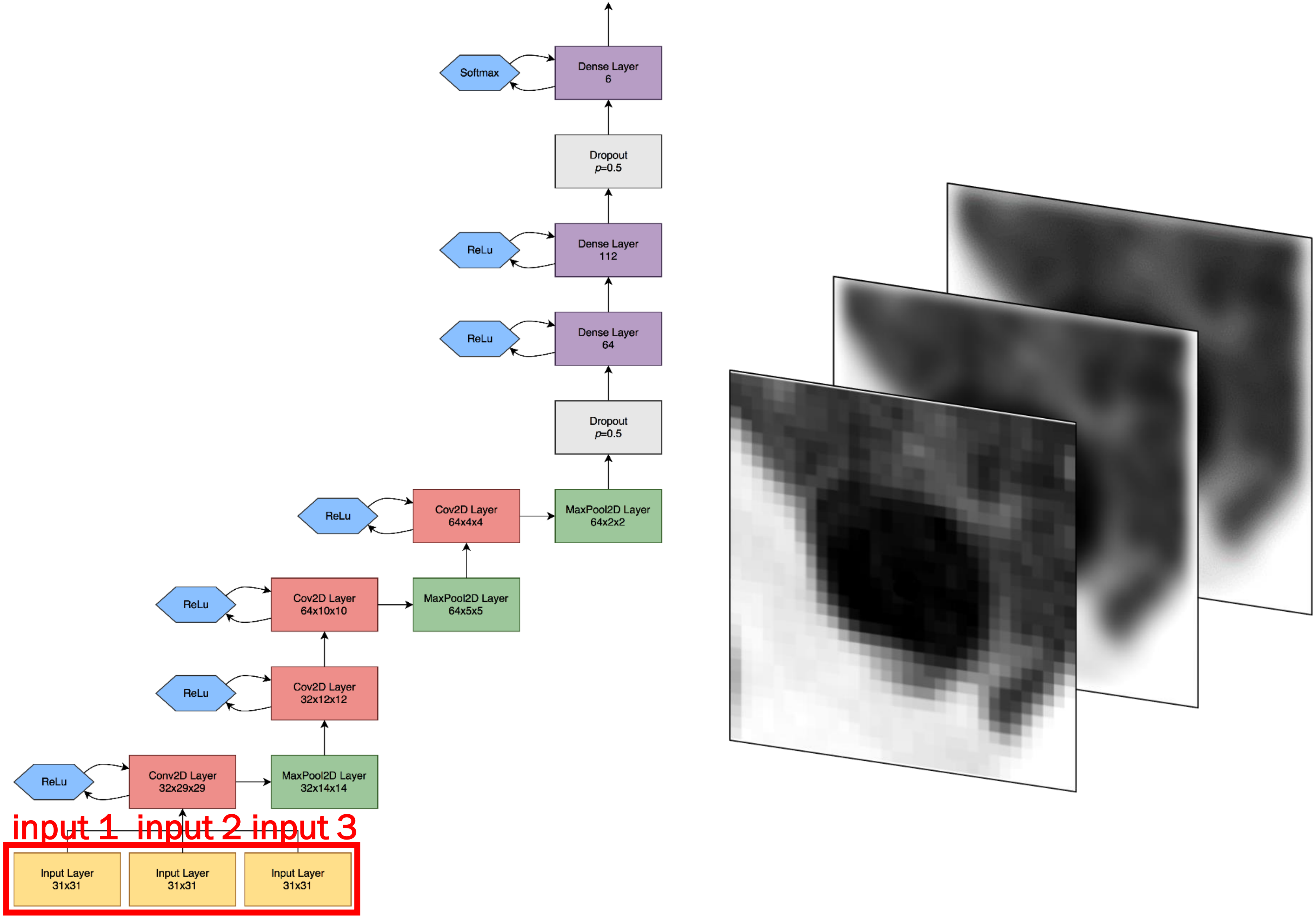
Multiscale Convolutional Neural Network (M-CNN)



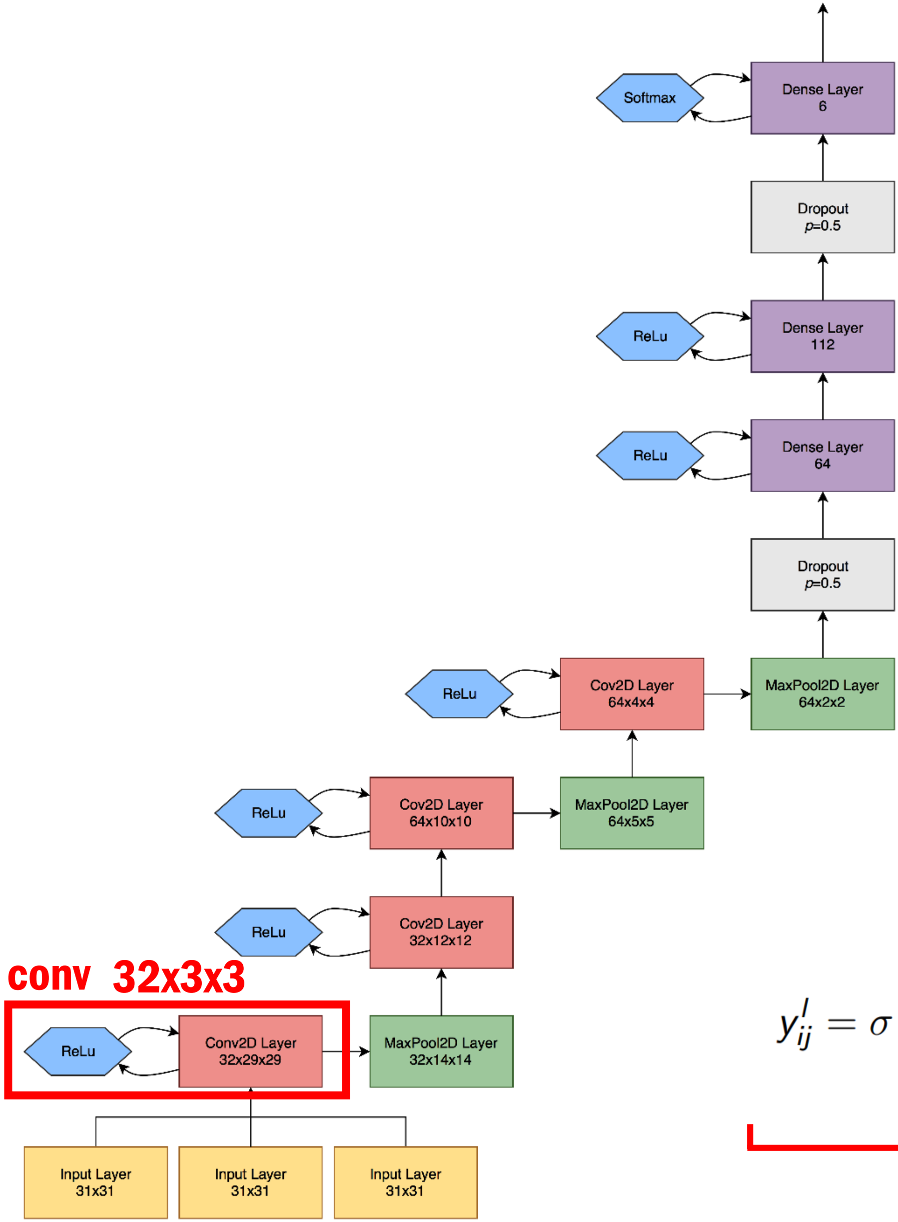


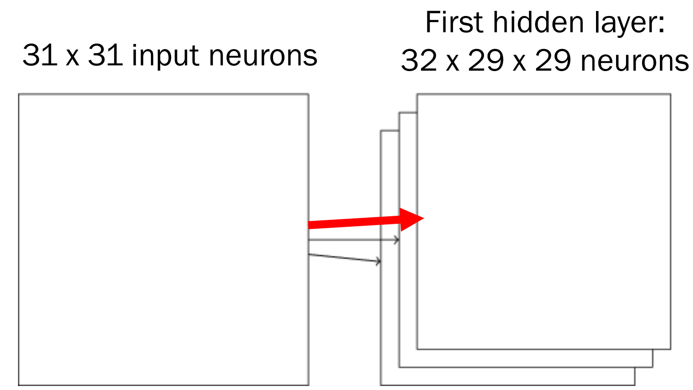


Multiscale Convolutional Neural Network (M-CNN)



Multiscale Convolutional Neural Network (M-CNN)





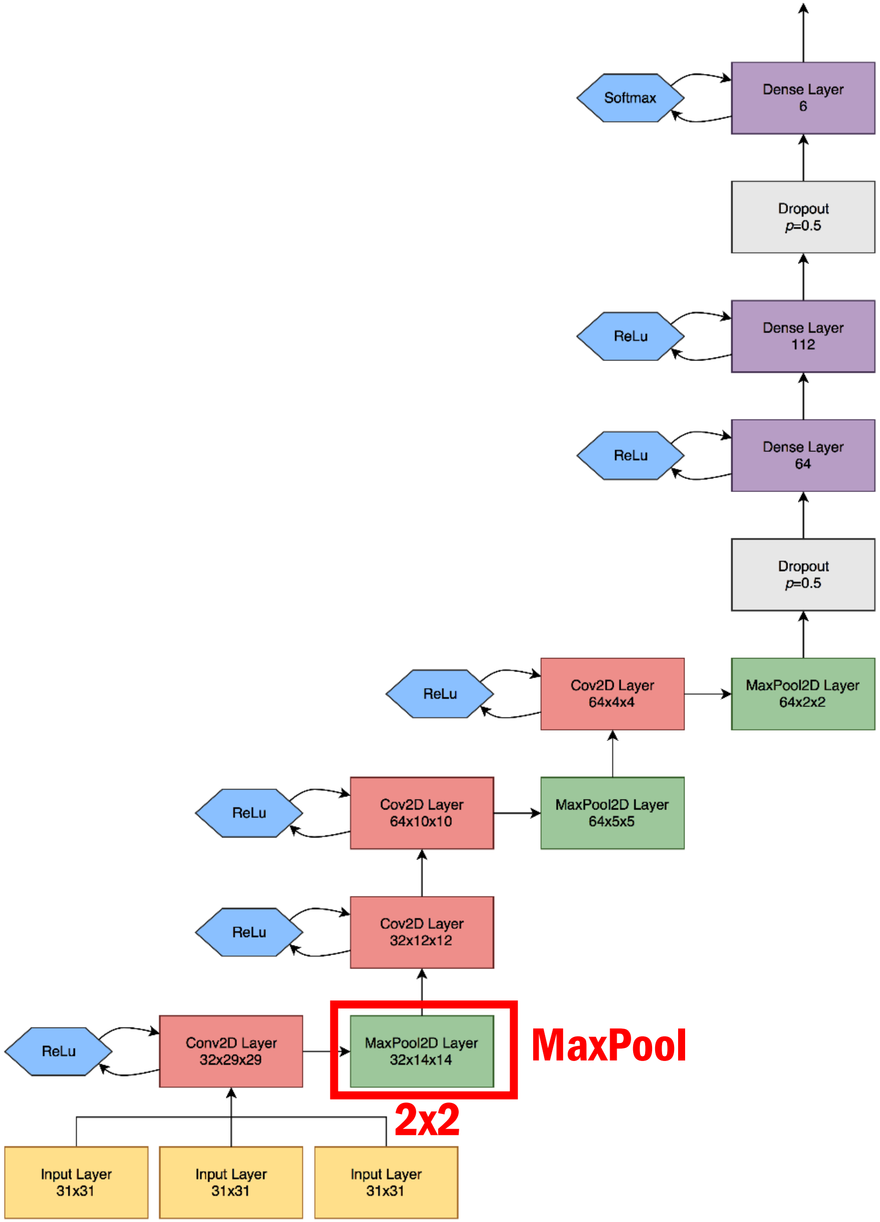
ReLU, Rectified Linear Units

$$y_{ij}^{l} = \sigma\left(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\omega_{ab}\, y_{(i+a)(j+b)}^{l-1}\right) \quad + \quad \varphi(v) = max(0, v)$$

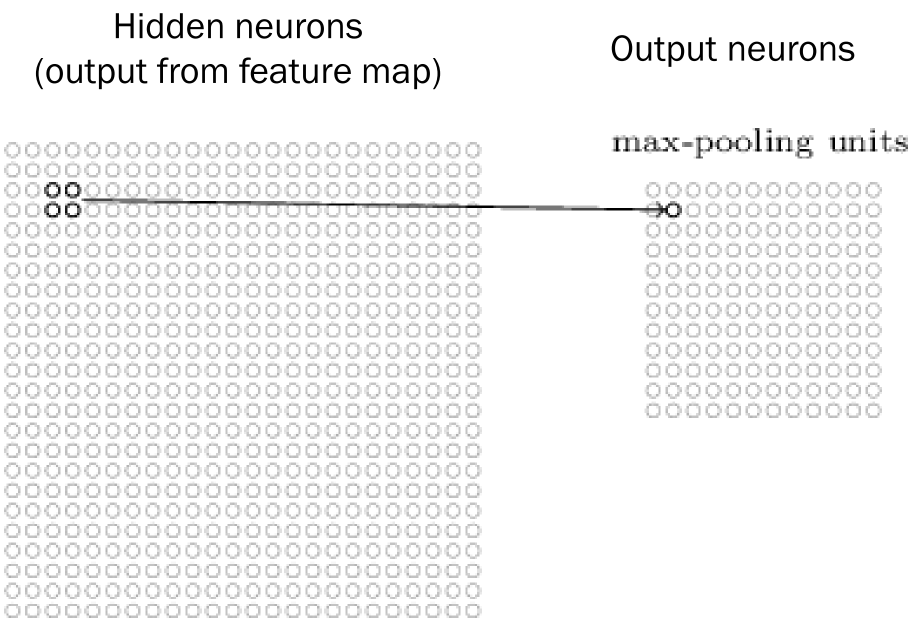
2D Convolution

Non-linear transformation

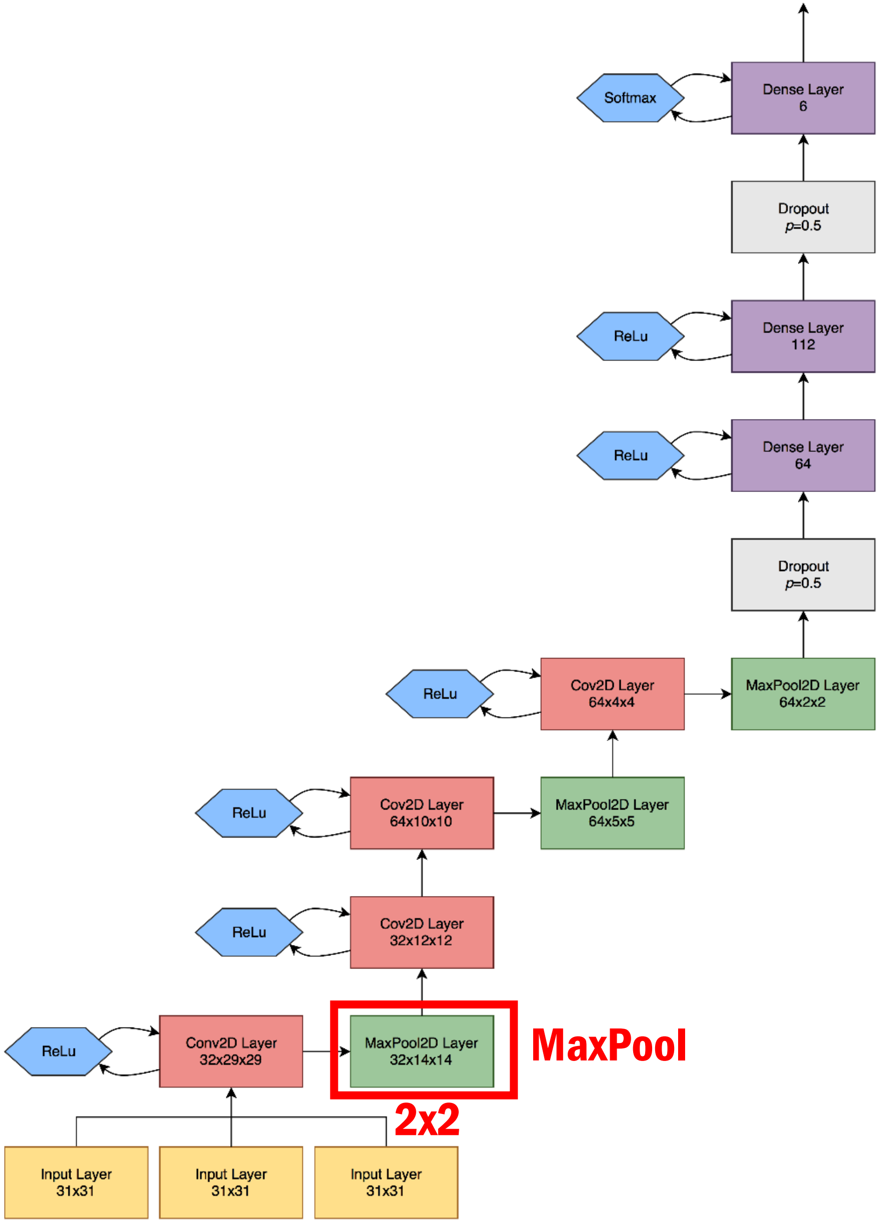
Multiscale Convolutional Neural Network (M-CNN)



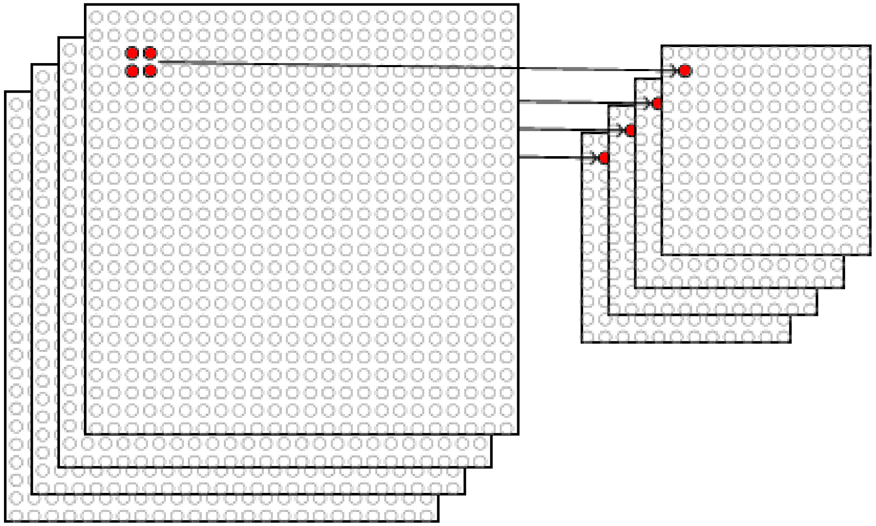
$$y_{ij}^{l} = \max_{a,b \in \{0,1,\ldots,m\}} y_{(i+a-1)(j+b-1)}^{l-1}$$



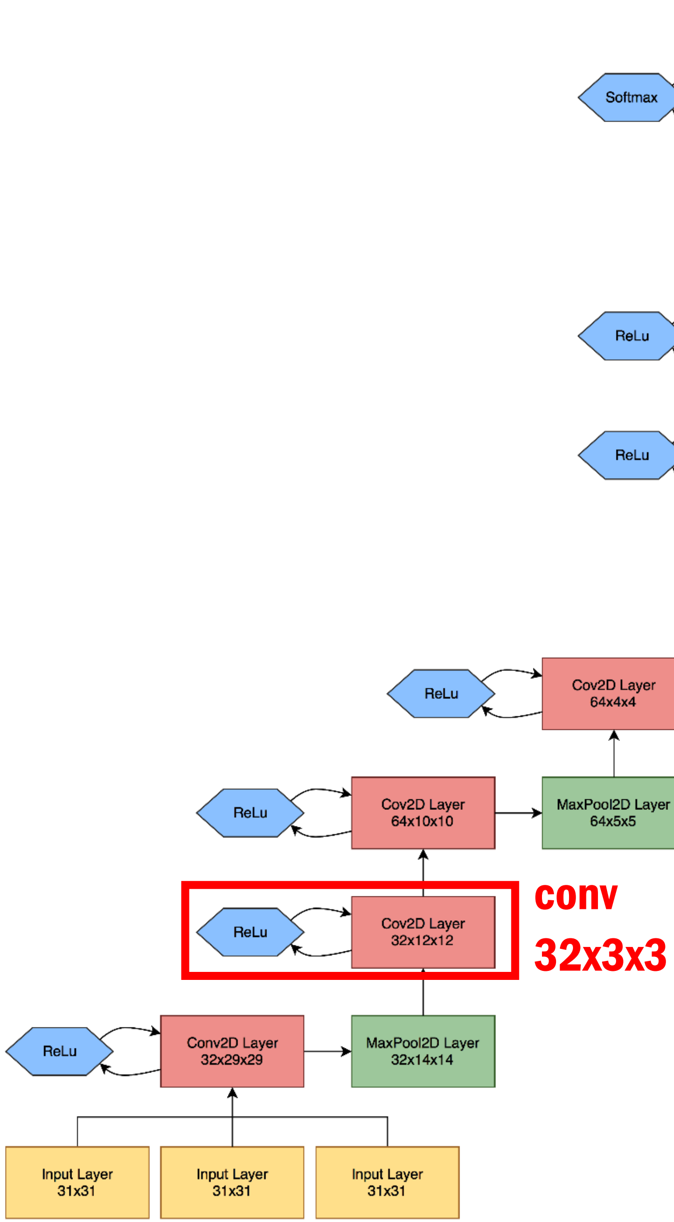
Multiscale Convolutional Neural Network (M-CNN)

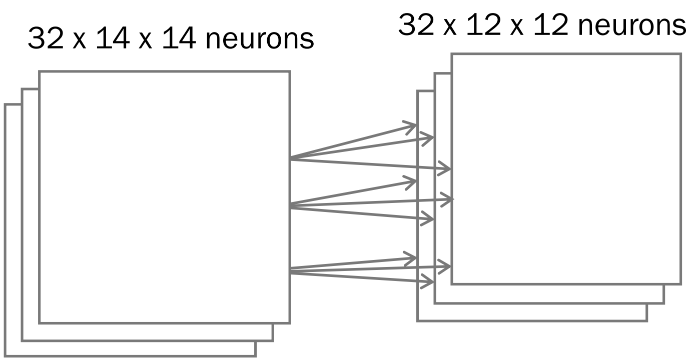


$$y_{ij}^{l} = \underset{a,b \in \{0,1,\ldots,m\}}{\text{máx}} y_{(i+a-1)(j+b-1)}^{l-1}$$

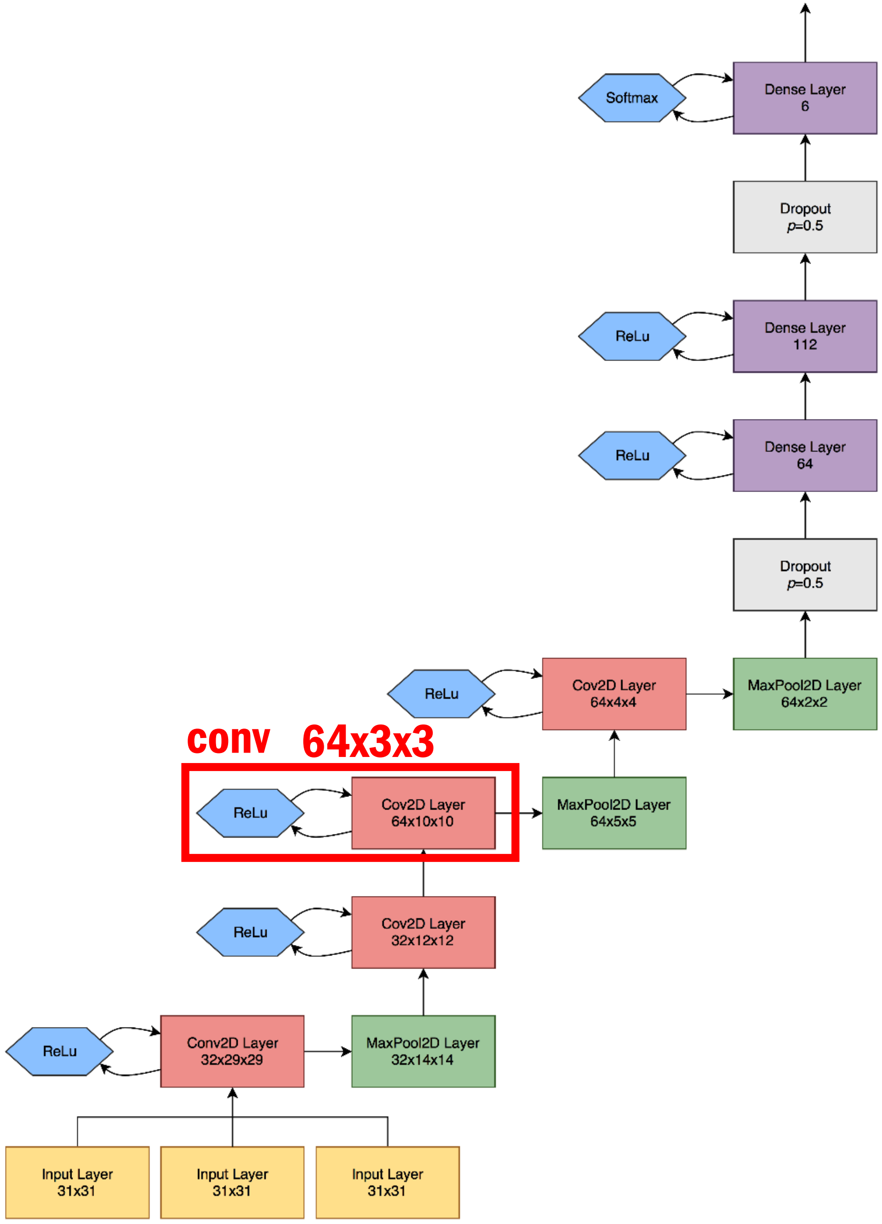


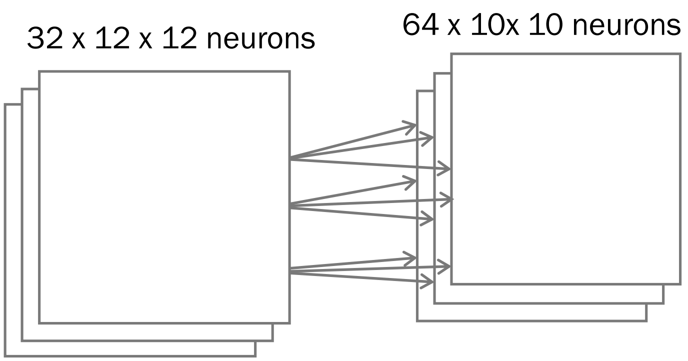
Multiscale Convolutional Neural Network (M-CNN)



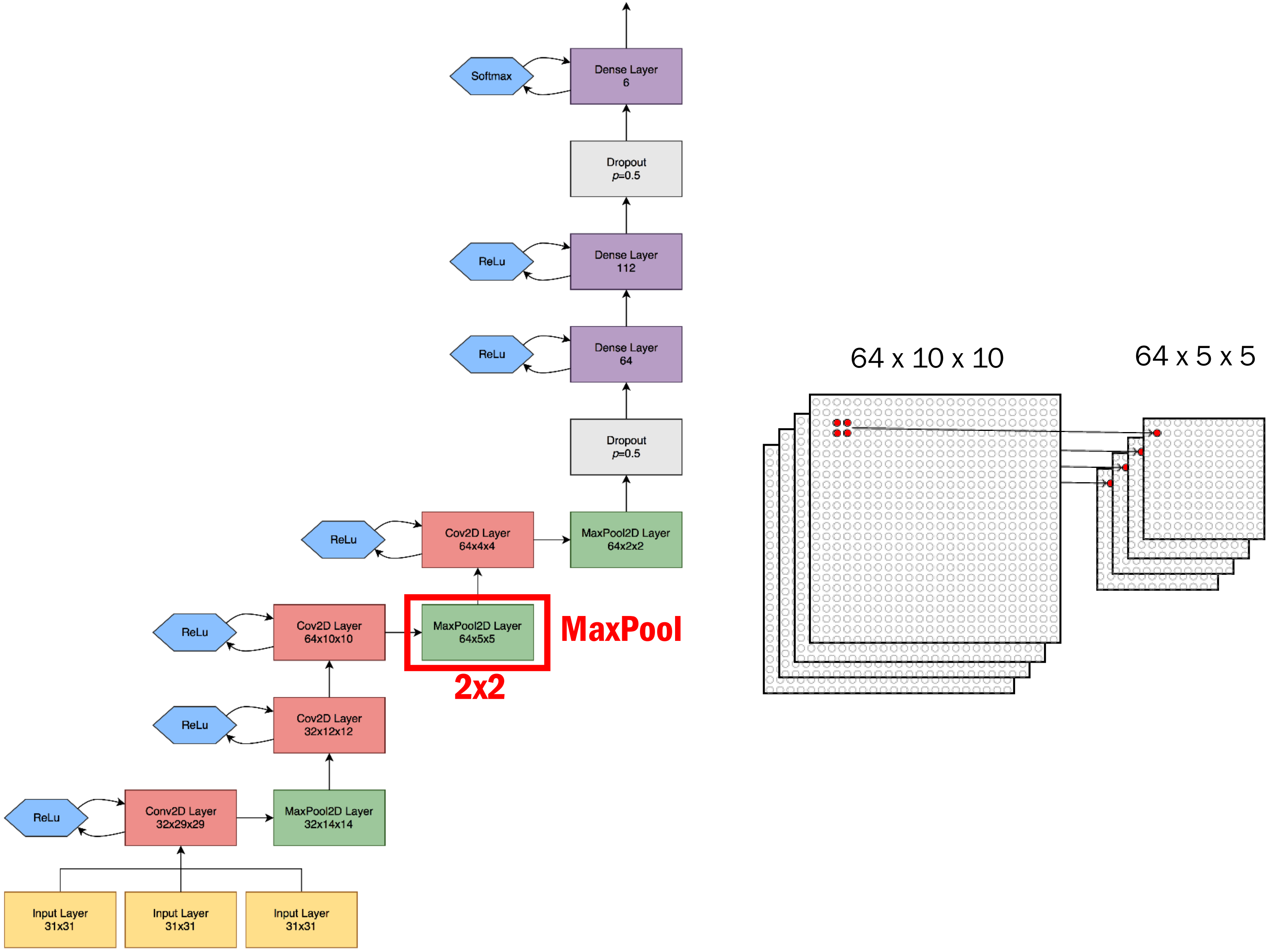


Multiscale Convolutional Neural Network (M-CNN)

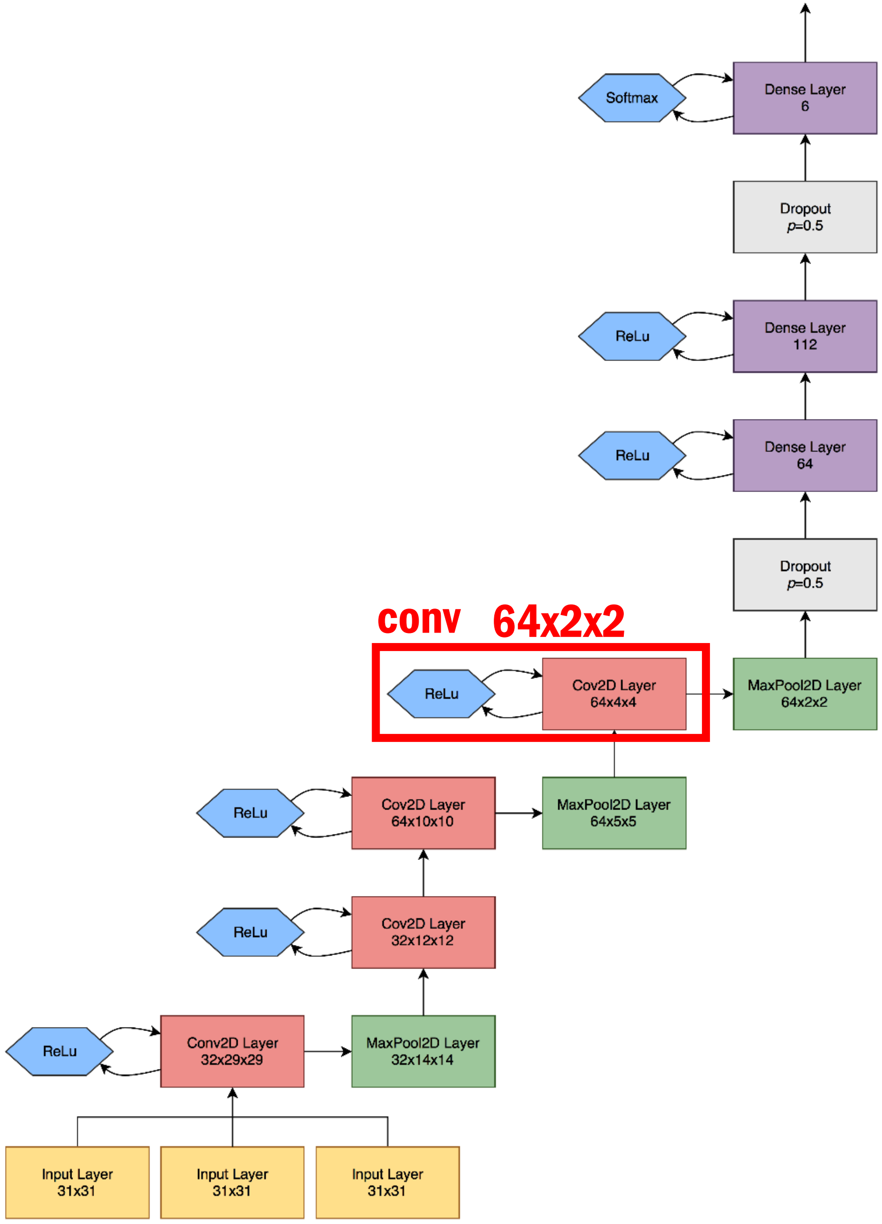


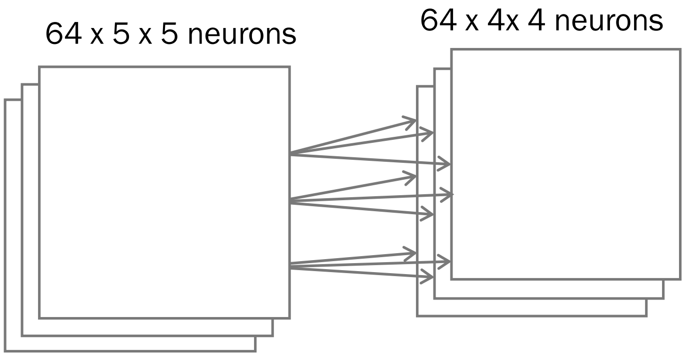


Multiscale Convolutional Neural Network (M-CNN)

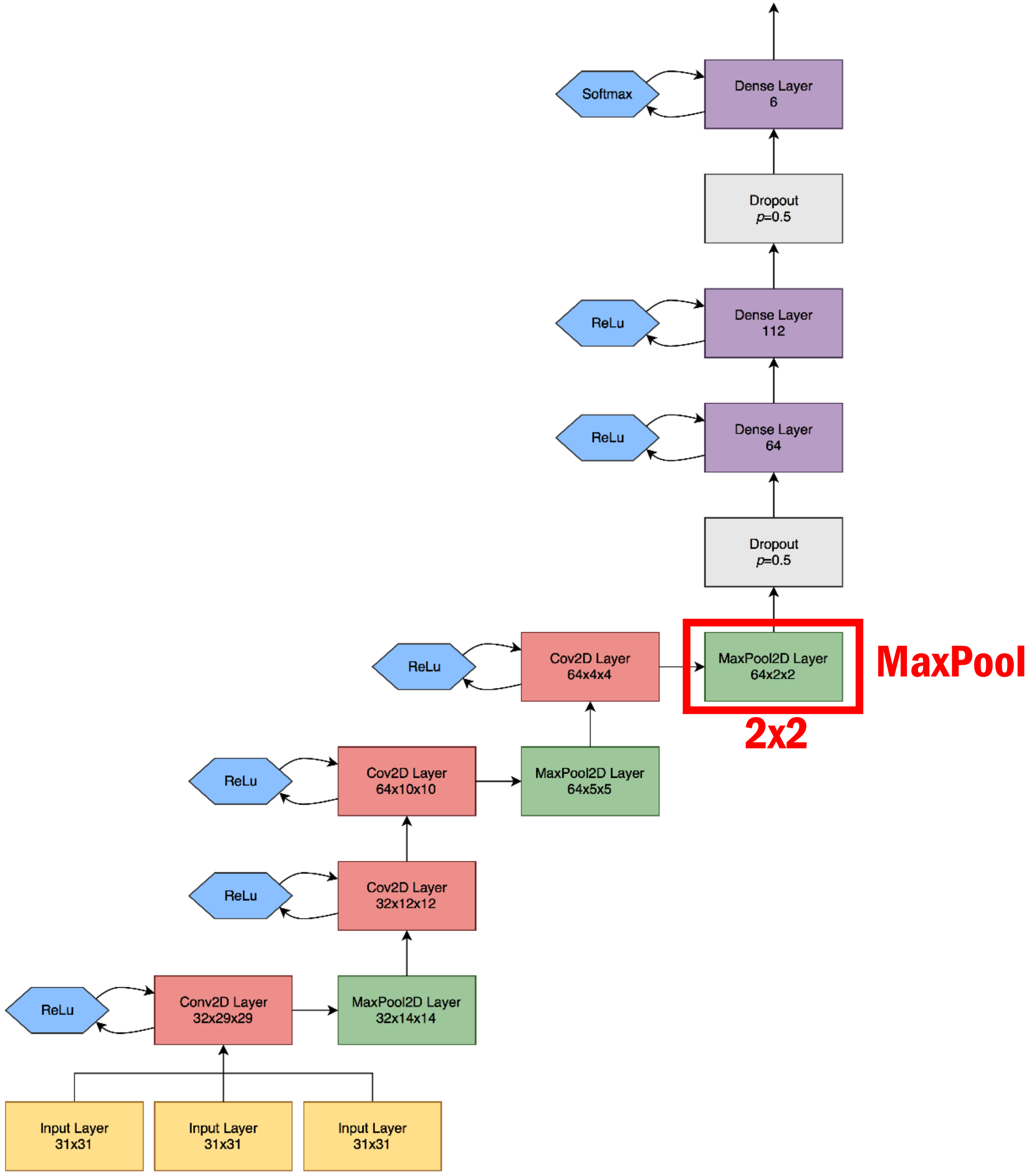


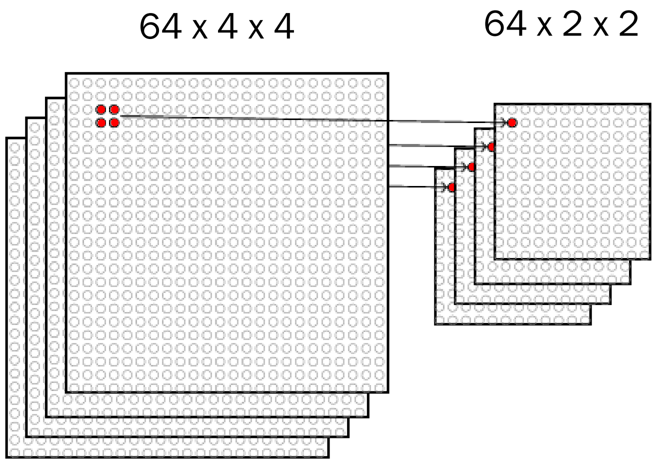
Multiscale Convolutional Neural Network (M-CNN)



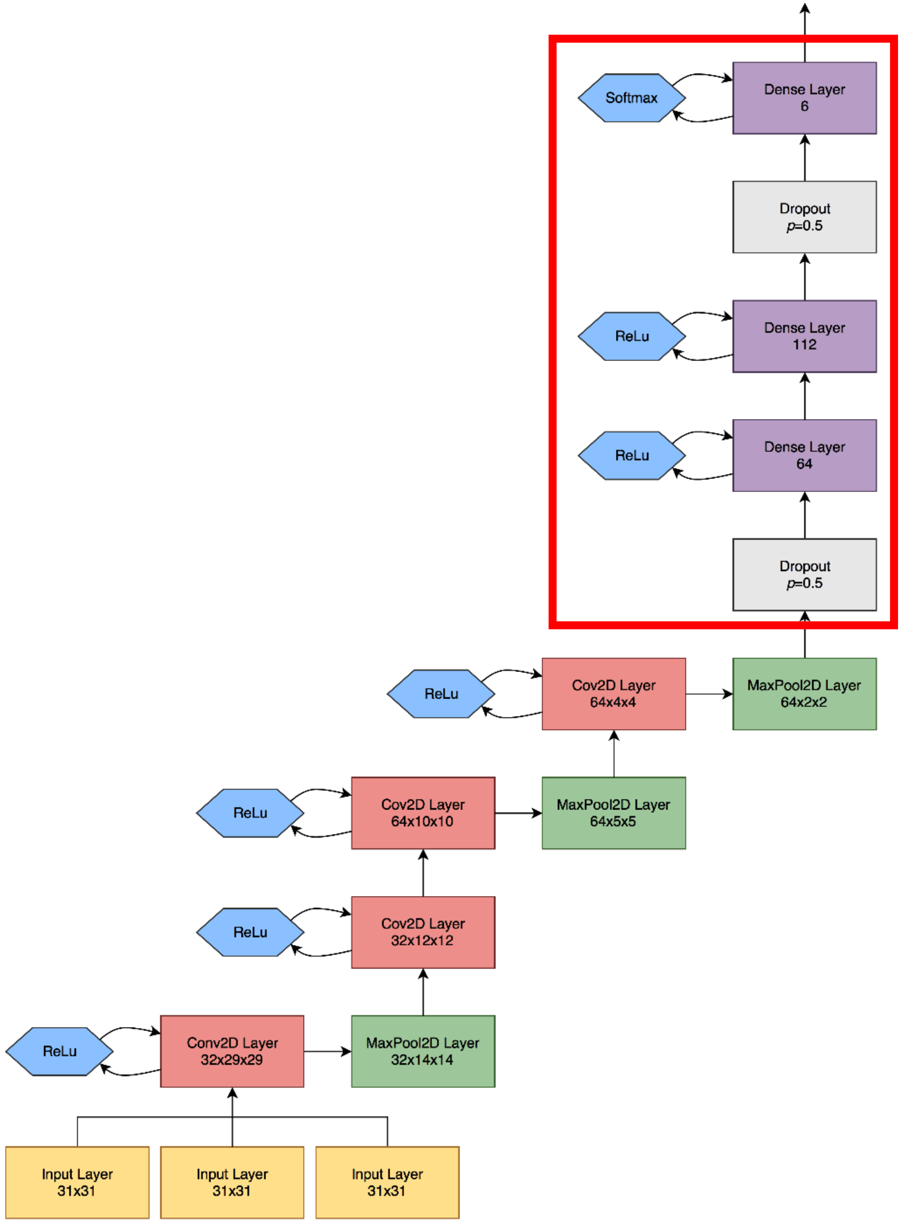


Multiscale Convolutional Neural Network (M-CNN)

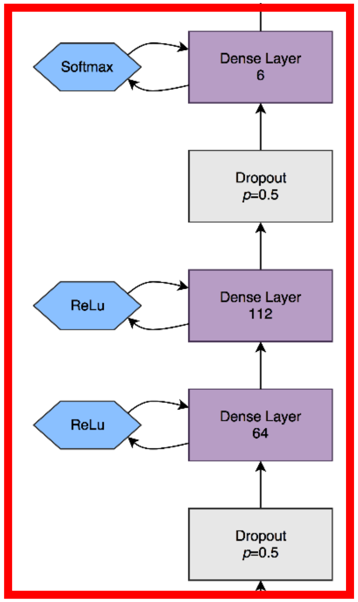




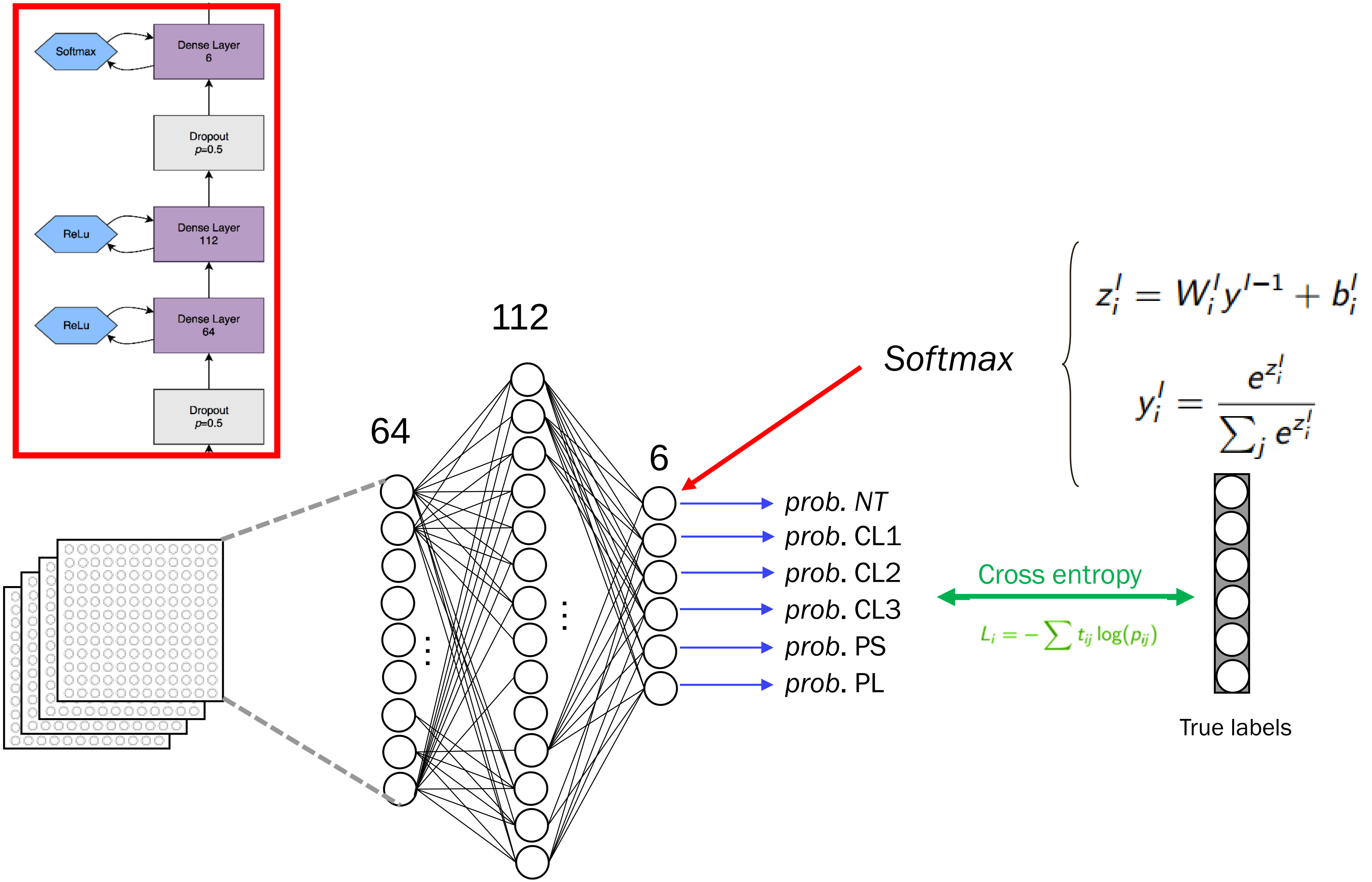
Multiscale Convolutional Neural Network (M-CNN)



Multiscale Convolutional Neural Network (M-CNN)



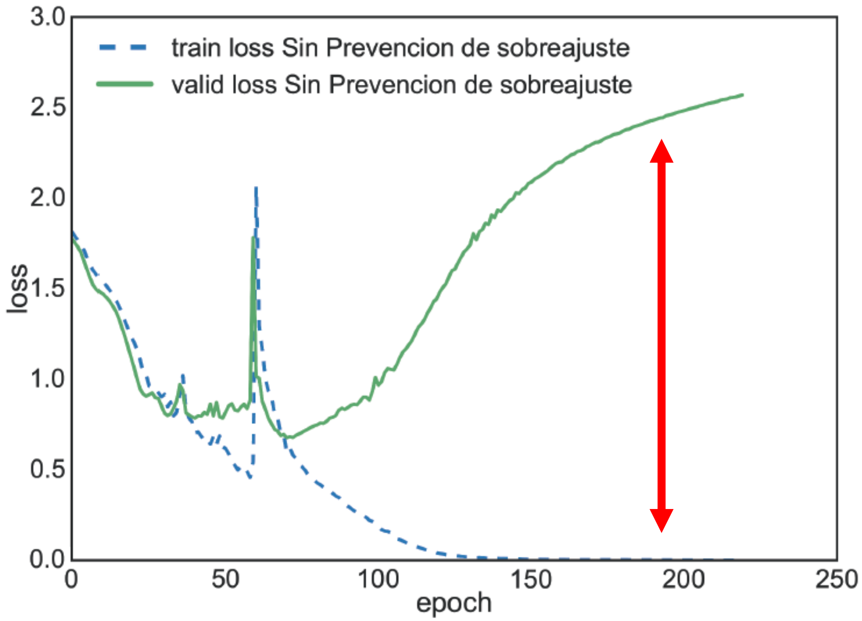
Multiscale Convolutional Neural Network (M-CNN)

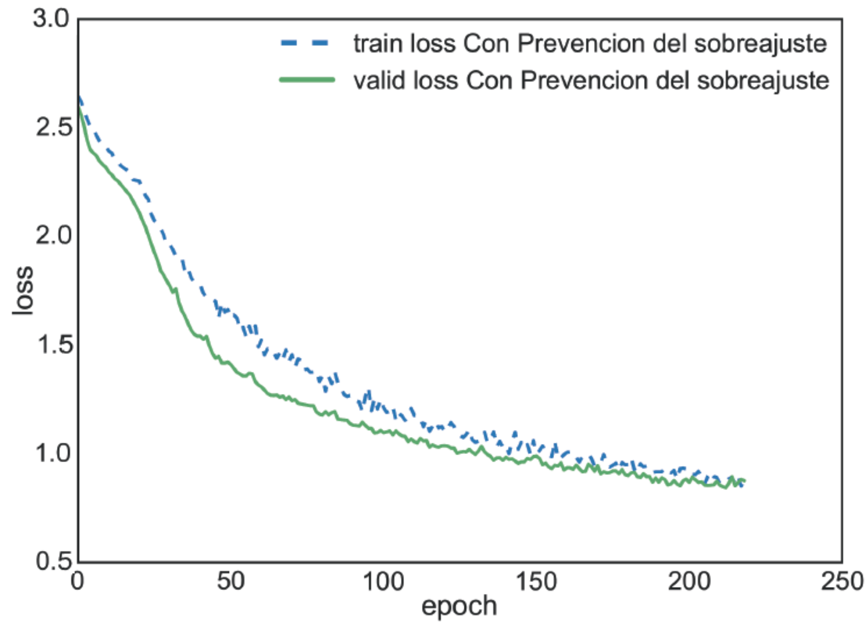


Multiscale Convolutional Neural Network (M-CNN)

– **Overfitting prevention** with:

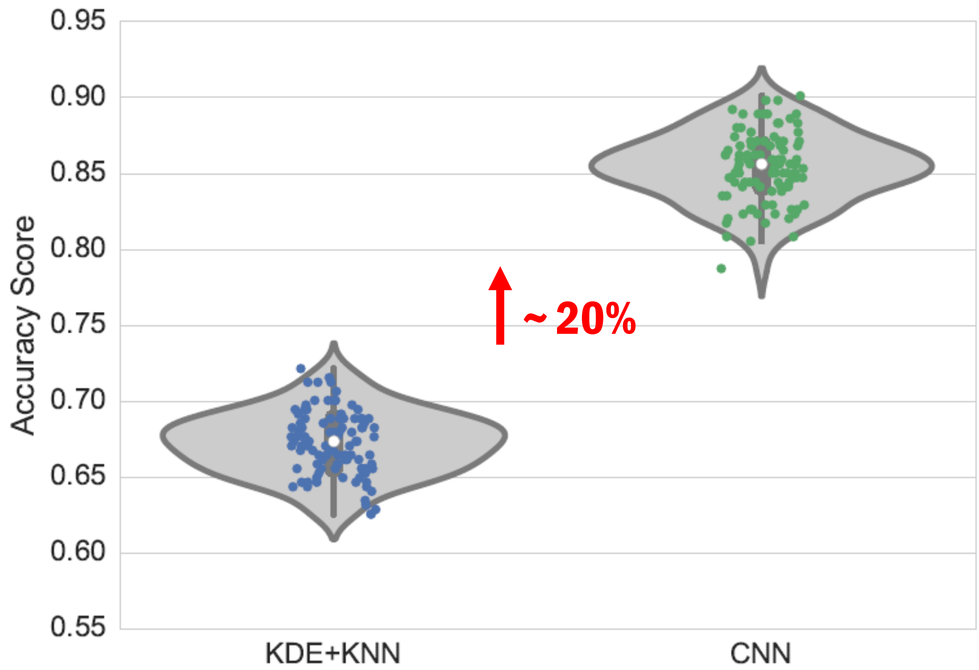
- Dropout
- L2 regularization
- Early Stopping
- Data Augmentation





Results – Accuracy

Random cross validation (k=100)

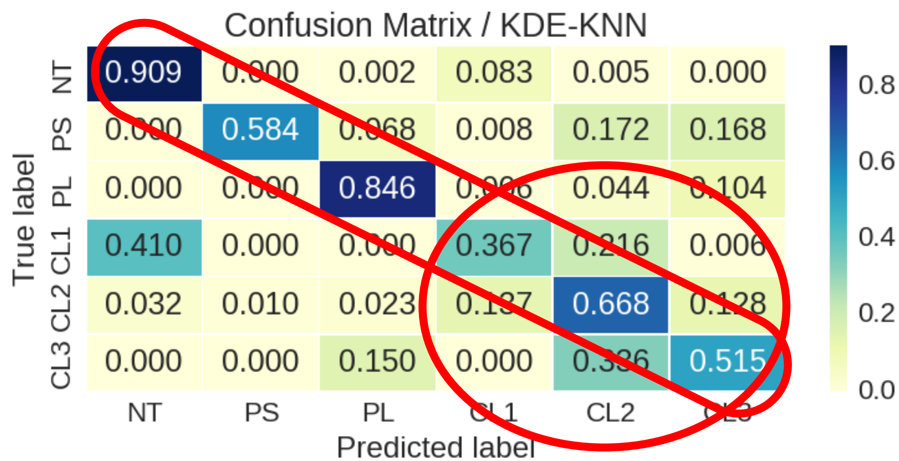


Method	Accuracy [mean (sd)]	95% CI [LL,UL]
KDE-KNN	0.679 (0.035)	[0.656, 0.702]
M-CNN	**0.891 (0.035)**	**[0.866, 0.913]**

Results – Confusion Matrix

Random cross validation (k=100)

MACHINE LEARNING



DEEP LEARNING