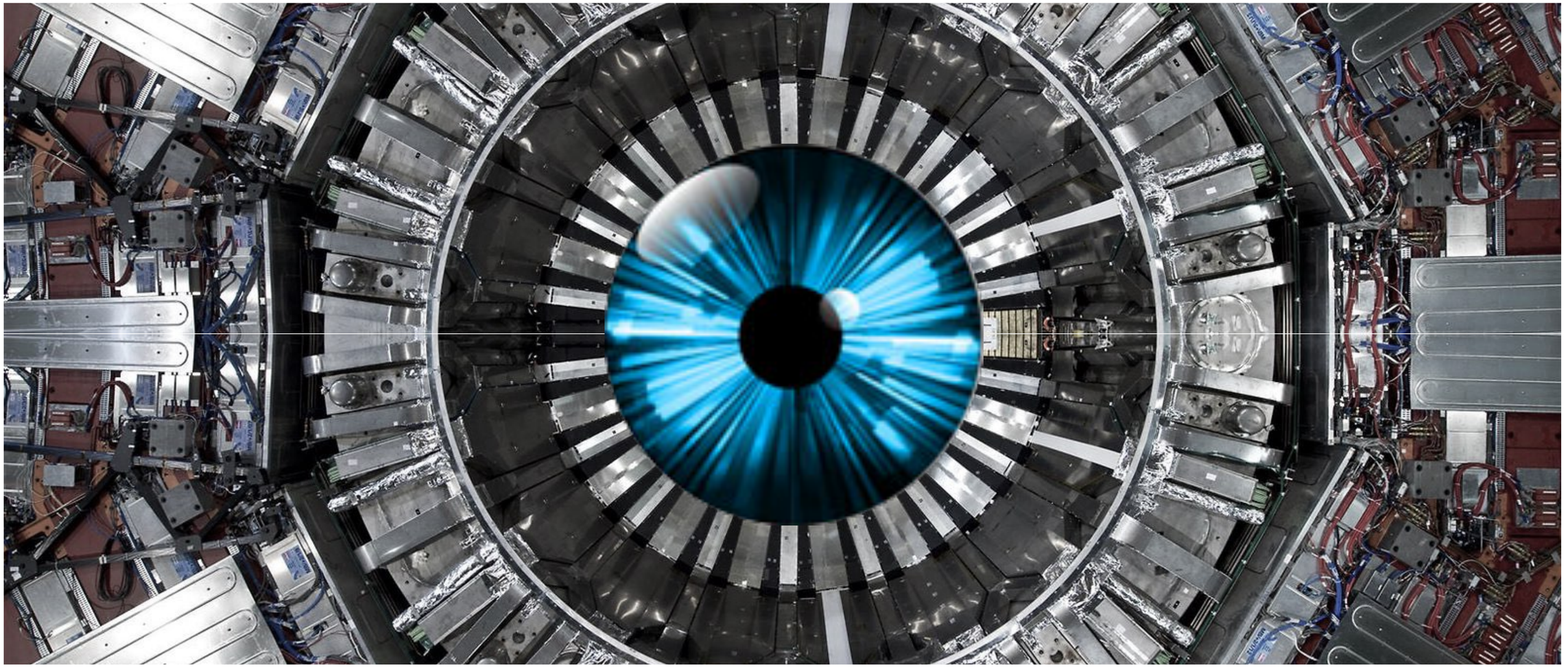
Deep Learning Applications for collider physics Lecture 2

Maurizio Pierini

Plan for this week

	Day1	Day2	Day3	Day4	Day5
Lecture	Introduction	ConvNN	RNNs	Graphs	Unsupervised Learning
Tutorial	Fully Connected Classifier	ConvNN Classifier	RNNs Classifier	Graphs Classifier	Anomaly Detection

European Research Council

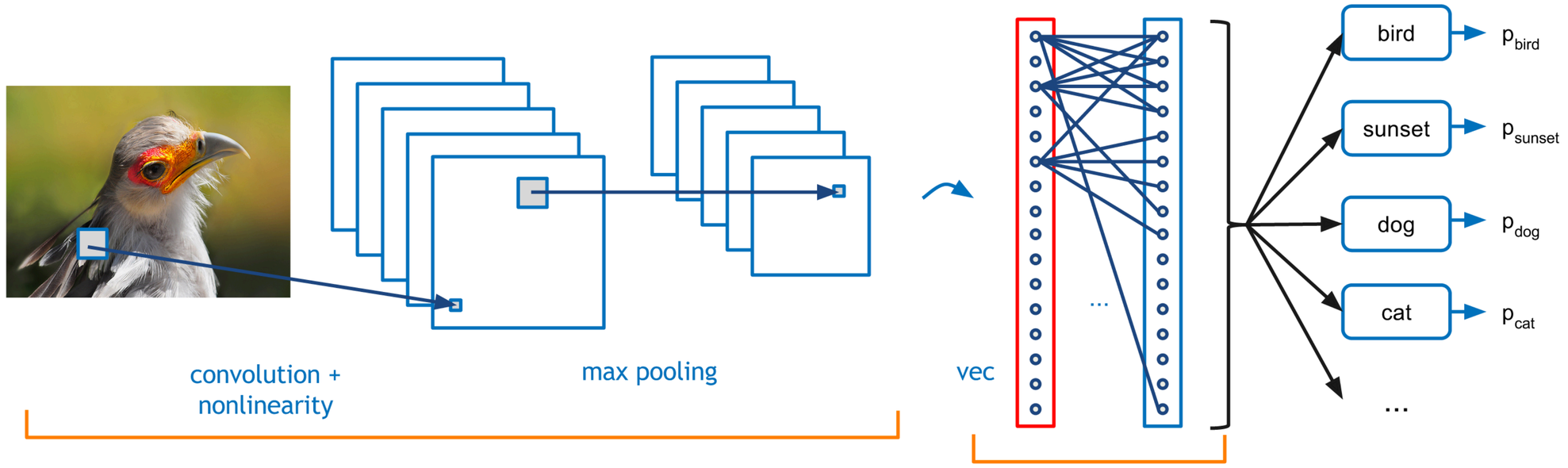


Particle Reconstruction & Computer Vision

European Research Council

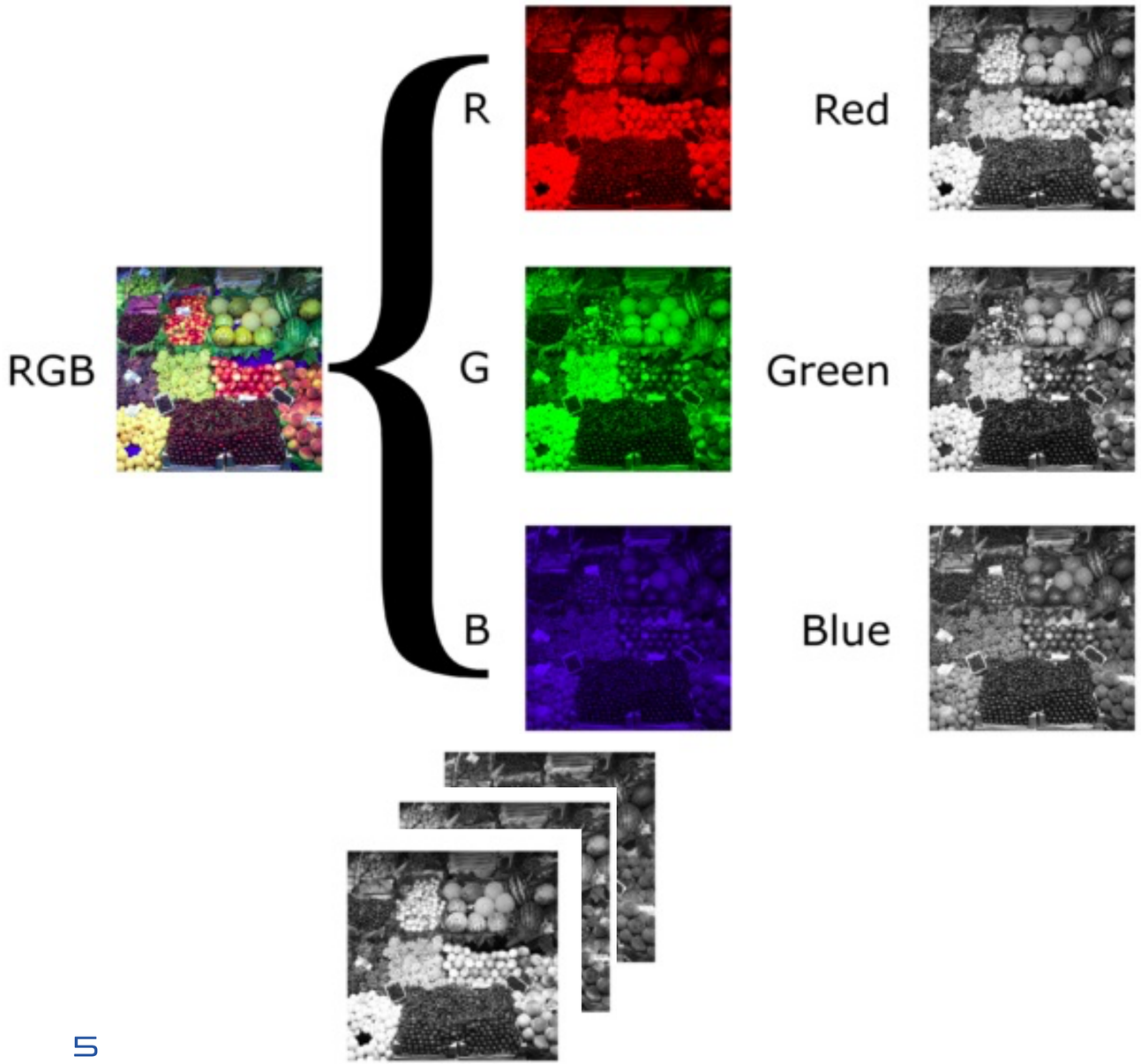
Convolutional Layer

- *Special architectures read the raw information (e.g., images) and convert them into "smart variables" (high-level features) to accomplish the task*
- *Typical example: convolutional neural networks for image processing & computing vision*



Digital images

- *Each image is a matrix of pixels*
- *Each pixel comes with a color*
- *In RGB scheme, these are three values i [0,1]*
- *Each image becomes a 3D tensor*
  - *3 channels of 2D pixelated images*



Convolutional filter

- *The main ingredient of ConvNN is a filter, a k x k' matrix of weights*
- *The filter scans the image and performs a scalar product of each image patch*
- *This results into a new matrix of values, with different dimensionality*

0	3	5	6	2	4	5
7	4	7	3	6	3	4
9	1	2	1	9	6	0
9	2	1	1	7	3	5
8	0	4	7	6	8	0
8	3	4	5	5	3	4
7	9	4	6	5	2	6

4	-1	4
-2	2	-5
3	1	-6

0x4 - 3x1 + 5x4 +
-7x2 + 4x2 - 7x5 +
9x3 + 1x1 - 2x6 = -8

-8				

Filtro convolutional

- *The main ingredient of ConvNN is a filter, a k x k' matrix of weights*
- *The filter scans the image and performs a scalar product of each image patch*
- *The scan is done shifting the filter by a stride (of q, 2, … cells)*
- *This results into a new matrix of values, with different dimensionality*

0	3	5	6	2	4	5
7	4	7	3	6	3	4
9	1	2	1	9	6	0
9	2	1	1	7	3	5
8	0	4	7	6	8	0
8	3	4	5	5	3	4
7	9	4	6	5	2	6

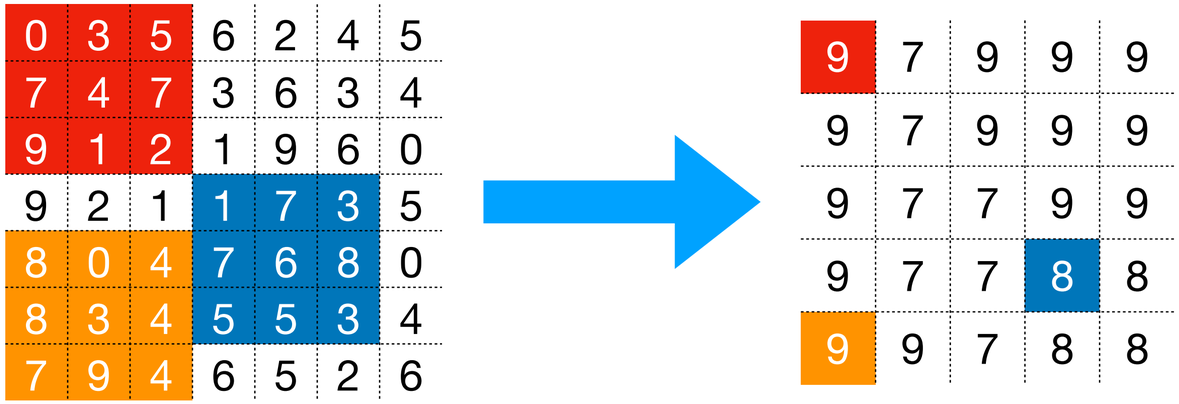
4	-1	4
-2	2	-5
3	1	-6

3x4 - 5x1 + 6x4 +
-4x2 + 7x2 - 3x5 +
1x3 + 2x1 - 1x6 = 21

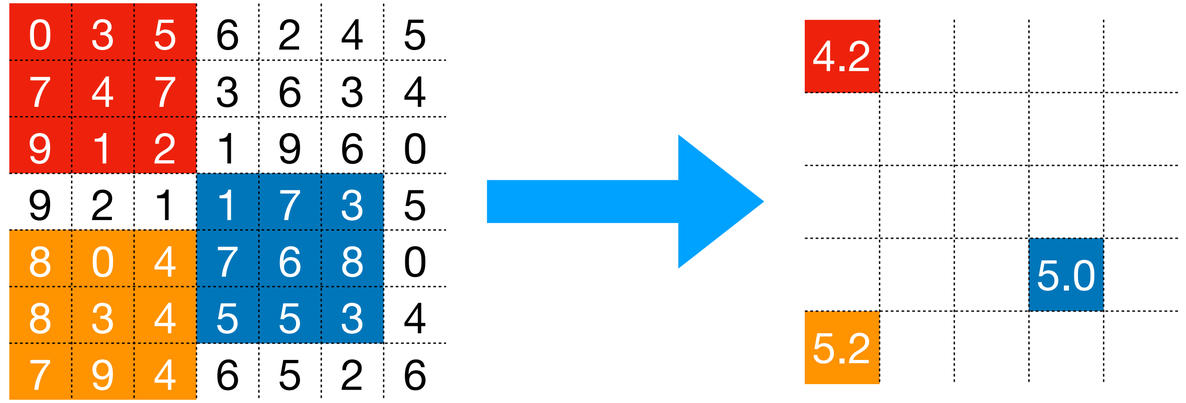
-8	21			

Pooling

- *MaxPooling: Given an image and a filter of size k x k', scans the image and replaces each k x k' patch with its maximum*



- *AveragePooling: Given an image and a filter of size k x k', scans the image and replaces each k x k' patch with its average*



- …

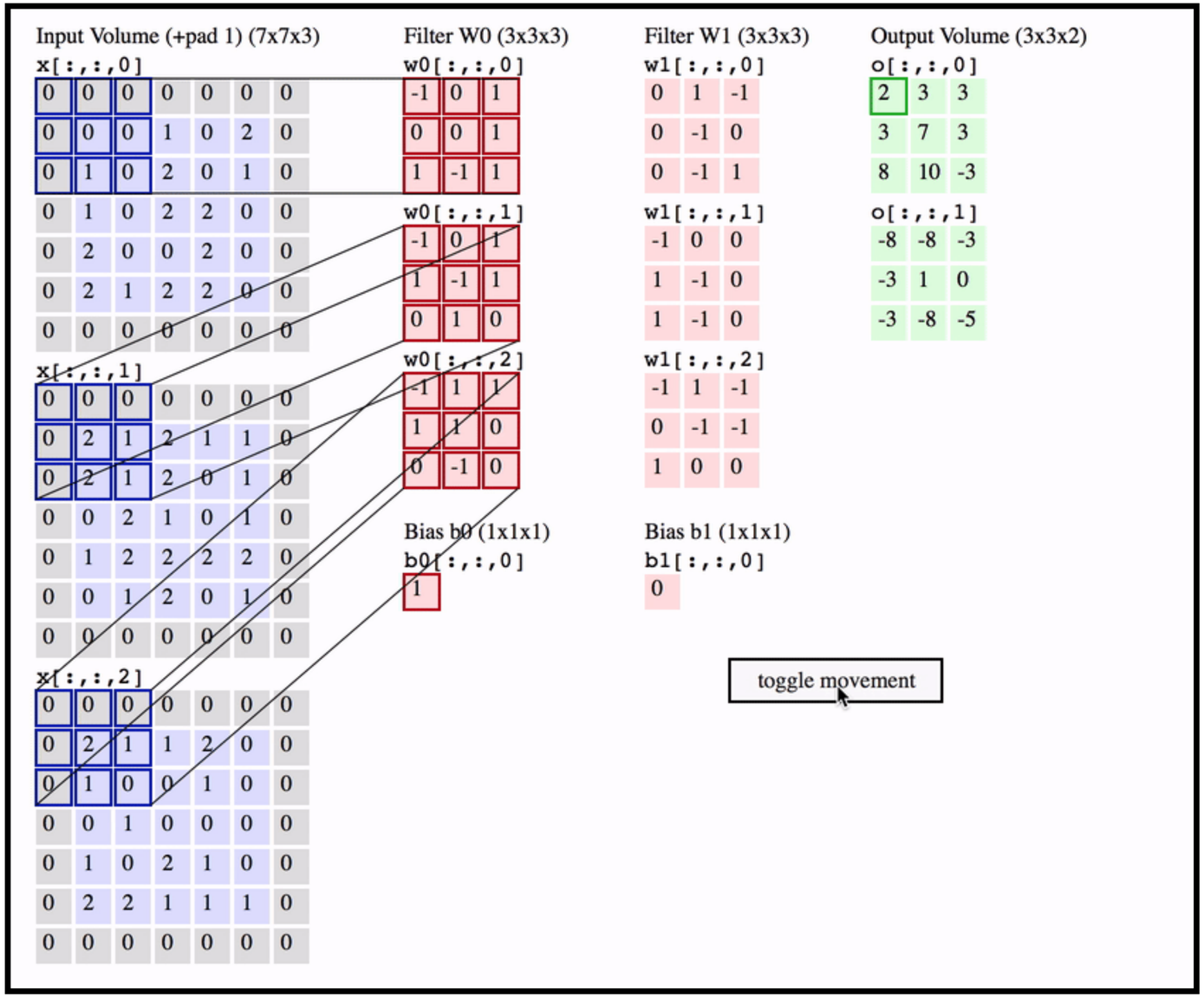
Padding

- *When the filter arrived at the edge, it might exceeds it (if n/k is not an integer)*
- *In this case, a padding rule needs to be specified*
  - *Same: repeat the values at the boundary*
  - *Zero: fill the extra columns with zeros*

0	3	5	6	2	4	4
7	4	7	3	6	3	3
9	1	2	1	9	6	6
9	2	1	1	7	3	3
8	0	4	7	6	8	8
8	3	4	5	5	3	3
8	3	4	5	5	3	3

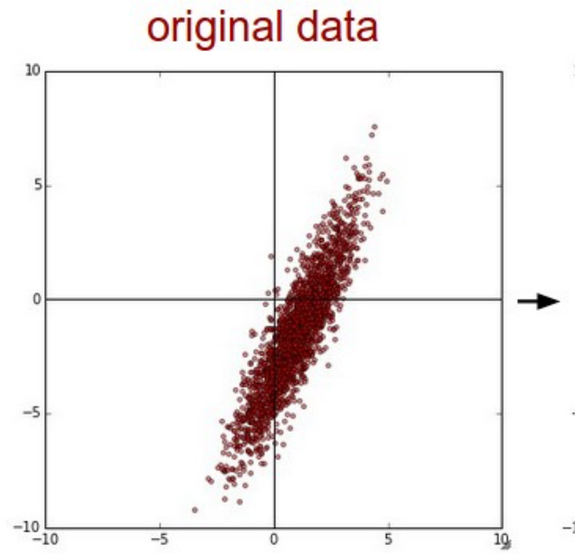
0	3	5	6	2	4	0
7	4	7	3	6	3	0
9	1	2	1	9	6	0
9	2	1	1	7	3	0
8	0	4	7	6	8	0
8	3	4	5	5	3	0
0	0	0	0	0	0	0

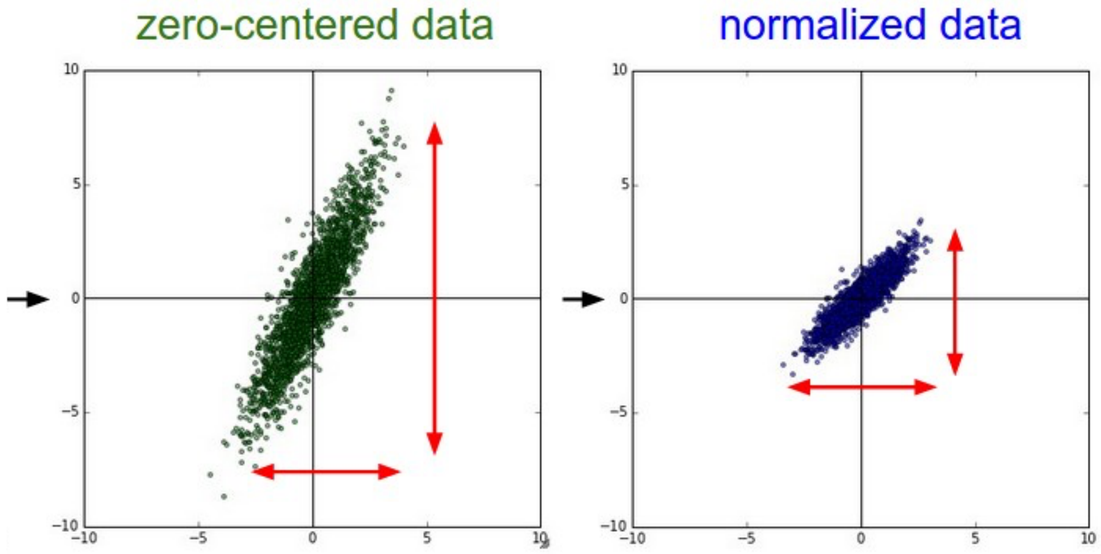
Convolutional Layer



BatchNormalization Layer

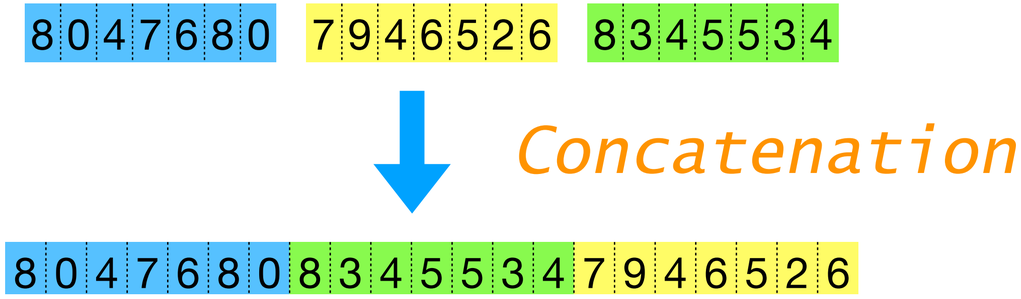
- *It is good practice to give normalized inputs to a layer*
  - *With all inputs having the same order of magnitude, all weights are equal important in the gradient*
  - *Prevents explosion of the loss function*
- *This can be done automatically with BatchNormalization*
  - *non-learnable shift and scale parameters, adjusted batch by batch*

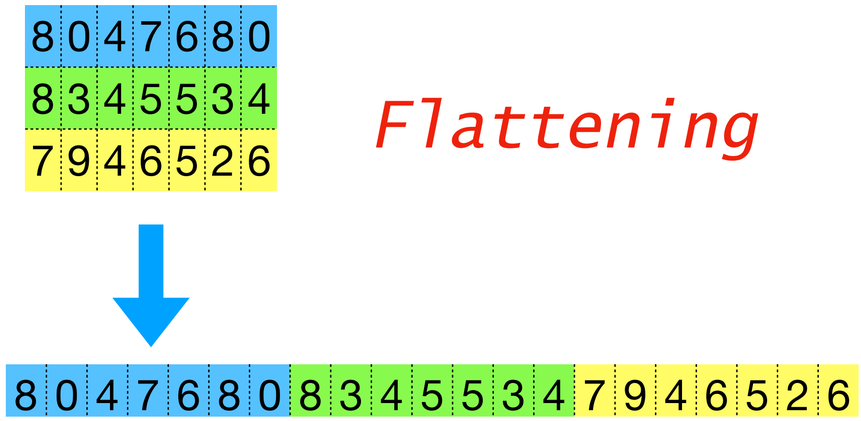




More complex structures

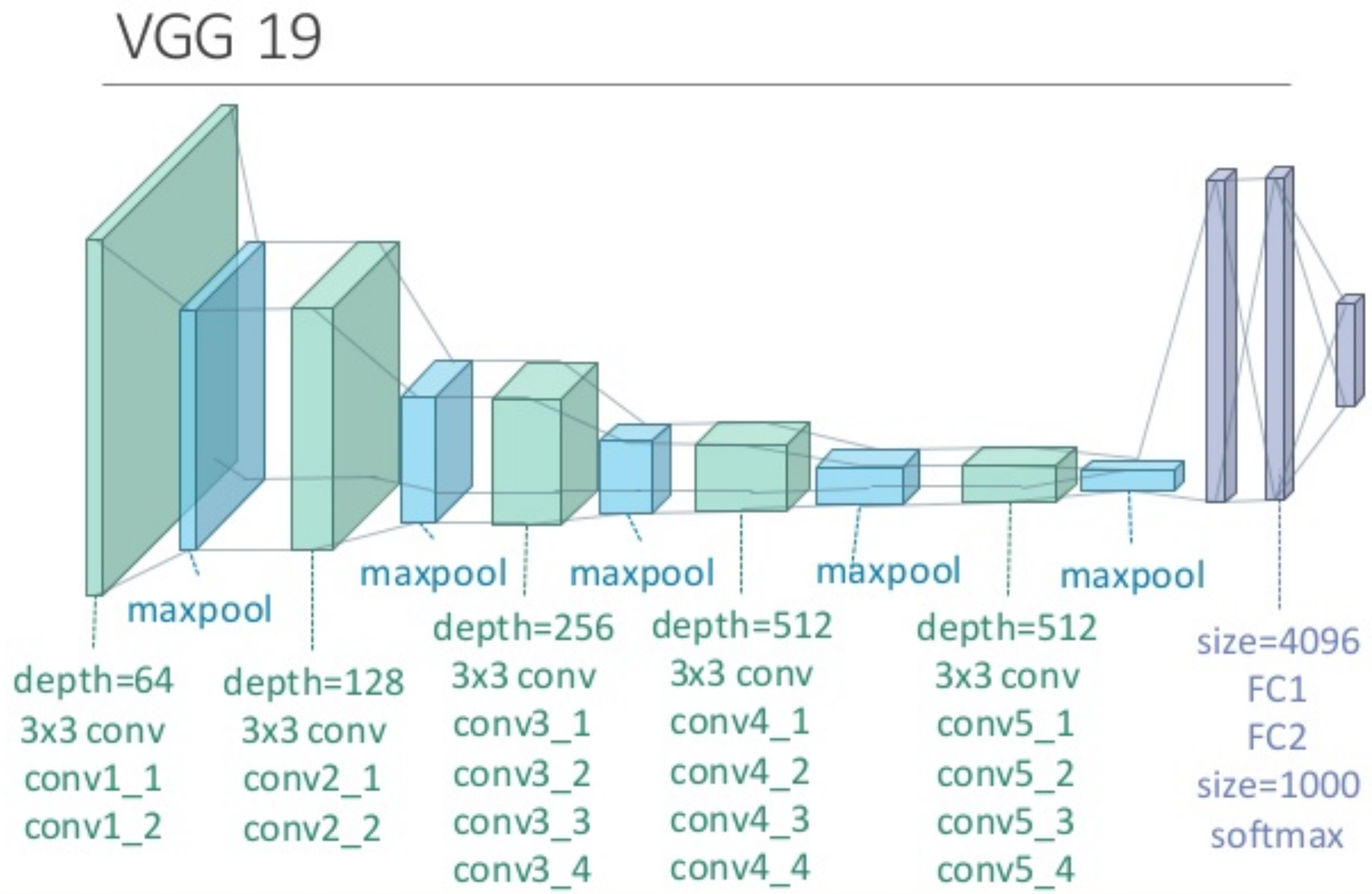
- Dense NN architectures can be made more complex
  - Multiple inputs
  - Multiple outputs
  - Different networks branches
- This is possible thanks to layer-manipulation layers
  - Add, Subtract, etc.
  - Concatenation
  - Flattening
- All these operations are usually provided with NN training libraries





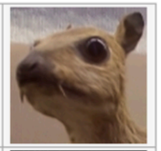
The full network

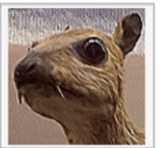
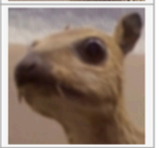
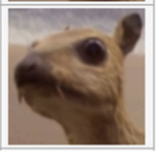
- *A full ConvNN is a sequence of Con2D+Pooling (+BatchNormalization+ Dropout) layers*
- *The Conv+Pooling layer reduces the 2D image representation*
- *The use of multiple filters on the image make the output grow on a third dimension*
- *Eventually, flattening occurs and the result is given to a dense layer*



What does a ConvNN learn?

- *Each filter alters the image in a different way, picking up different aspects of the image*
  - *edges oriented in various ways*
  - *enhancing / blur of certain features*
- *It is interesting to check what each filter is doing (and to produce DeepDreams)*

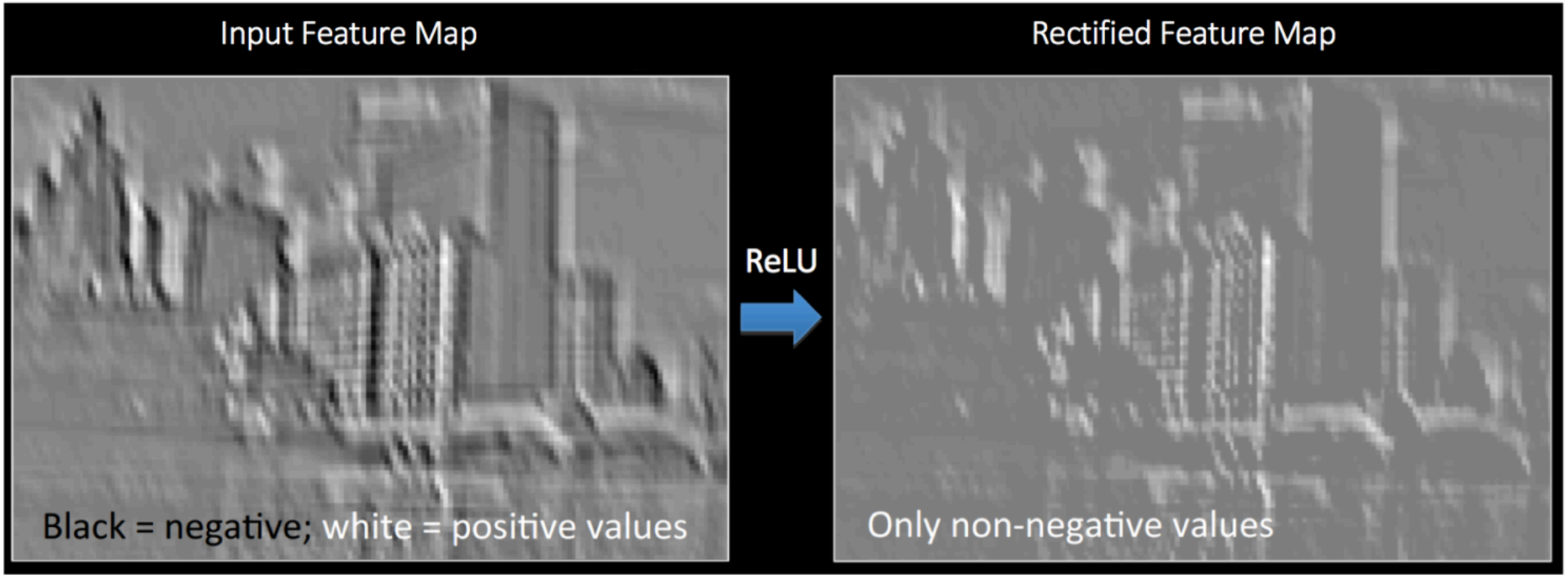
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Operation	Filter	Convolved Image
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

[Check this blog entry on ConvNNs]

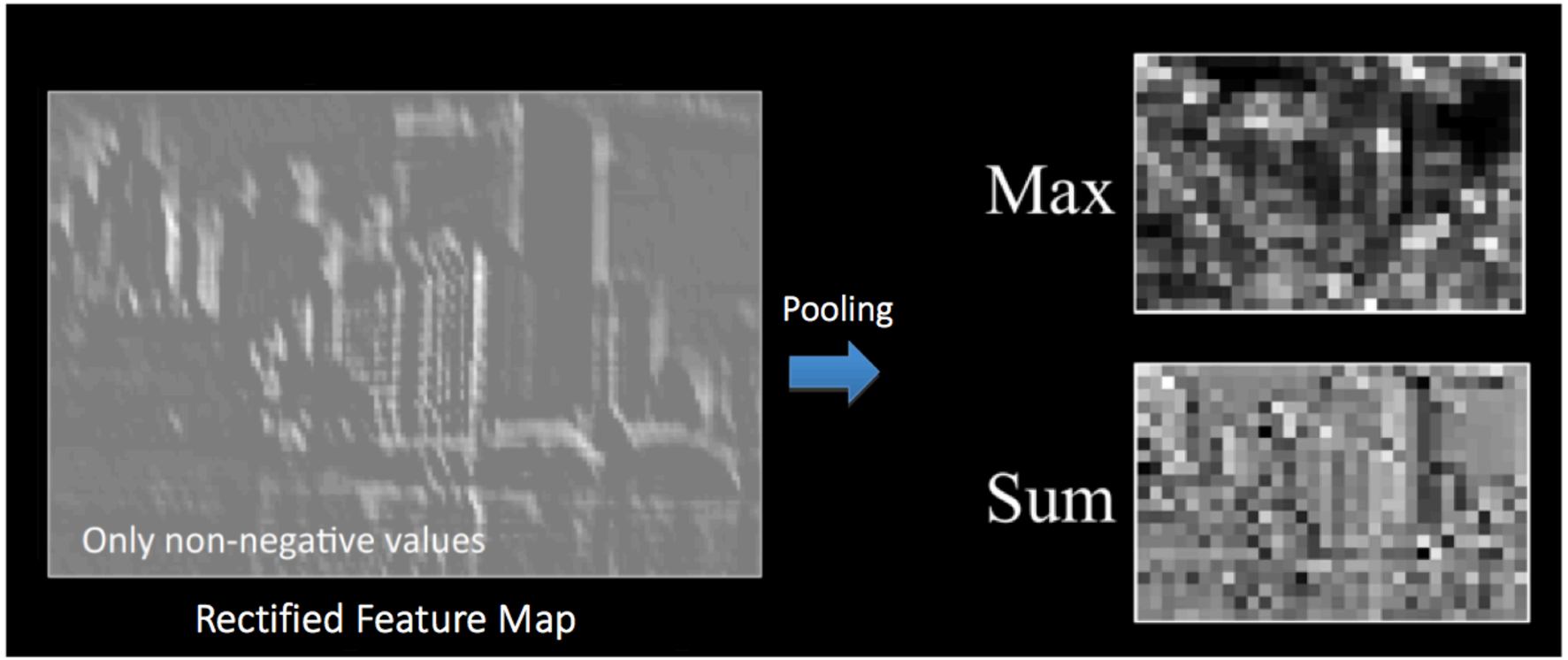
What does a ConvNN learn?

- *The use on non-linear activation functions plays a special role in enhancing features*



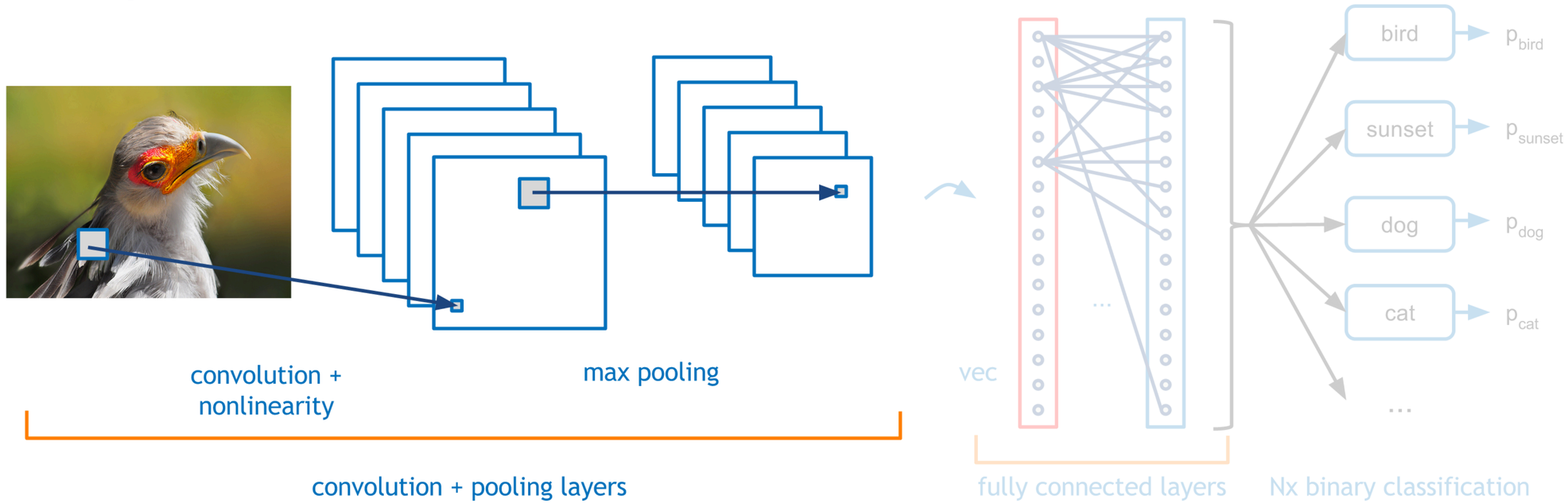
What does a ConvNN learn?

- *Pooling is important in smoothing out the image*
  - *reduce parameters downstream (and prevent overfitting)*
  - *makes processing independent to local features (distortion, translation)*
  - *yields a scale invariant representation of the image (which could be good or bad)*



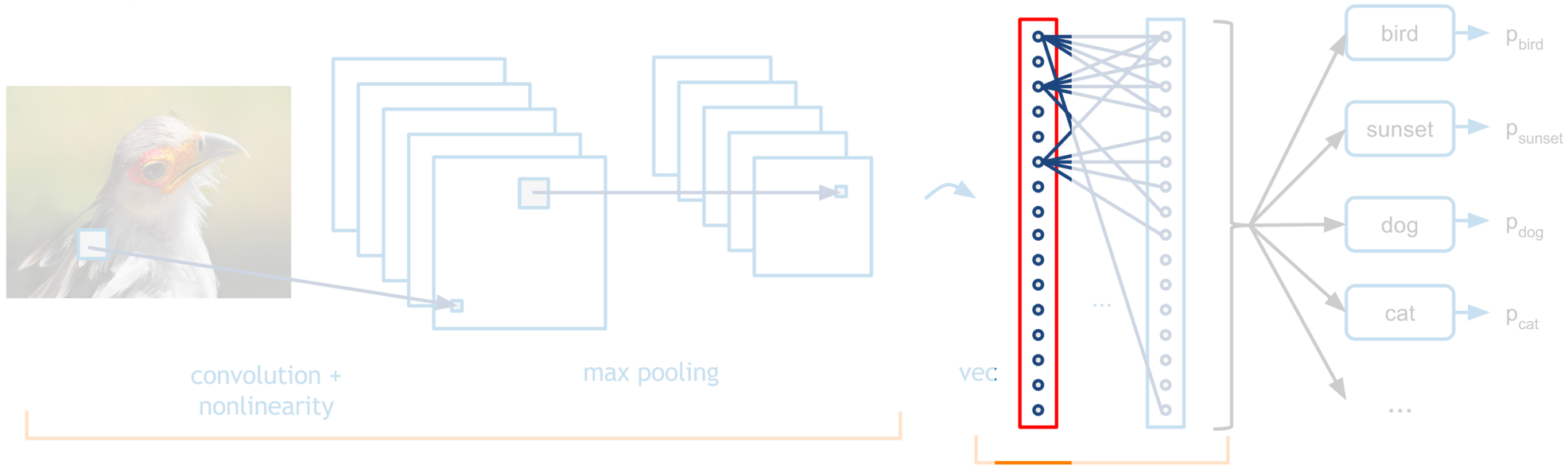
Two tasks in one network

- ***The Conv layer stars from RAW data and defines interesting quantities (high level features)***
- *The HLFs replace the physics-motivated inputs of a DNN*
- *The DNN at the end exploits the engineered features to accomplish the task*



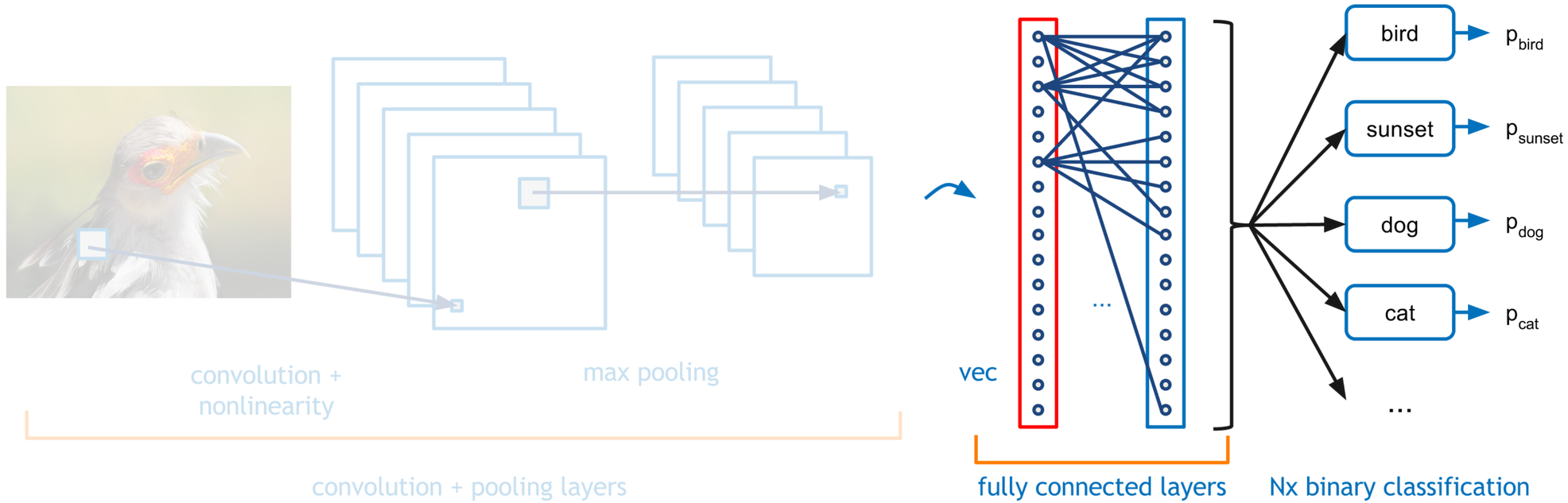
Two tasks in one network

- *The Conv layer stars from RAW data and defines interesting quantities (high level features)*
- ***The HLFs replace the physics-motivated inputs of a DNN***
- *The DNN at the end exploits the engineered features to accomplish the task*



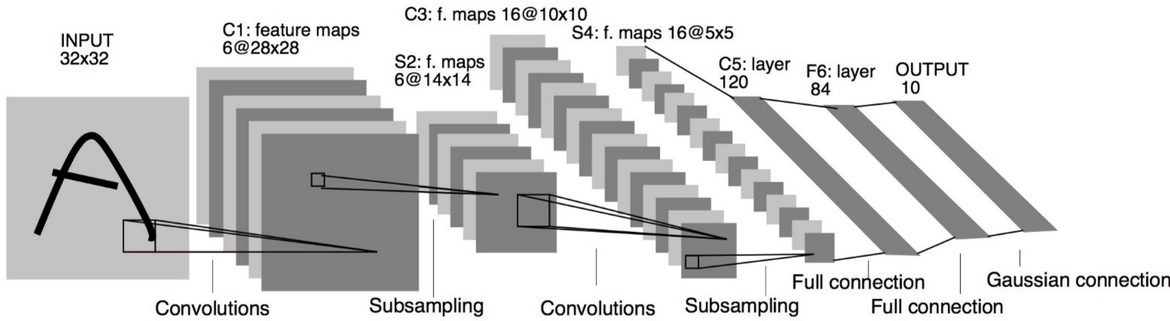
Two tasks in one network

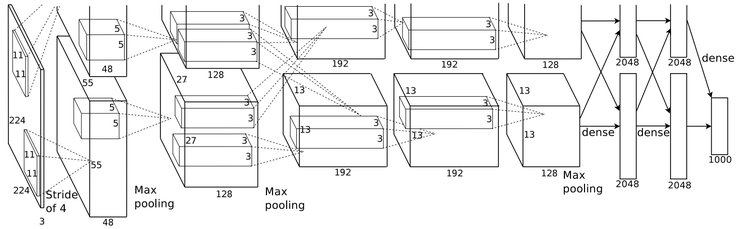
- *The Conv layer stars from RAW data and defines interesting quantities (high level features)*
- *The HLFs replace the physics-motivated inputs of a DNN*
- ***The DNN at the end exploits the engineered features to accomplish the task***

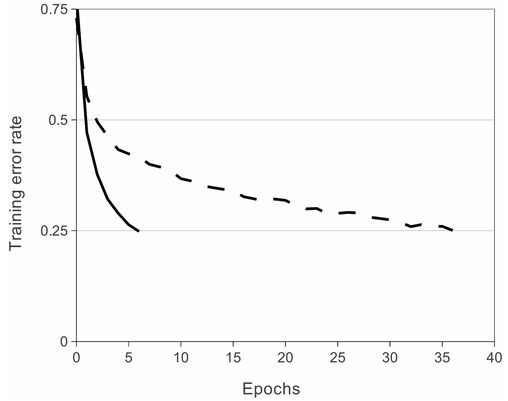


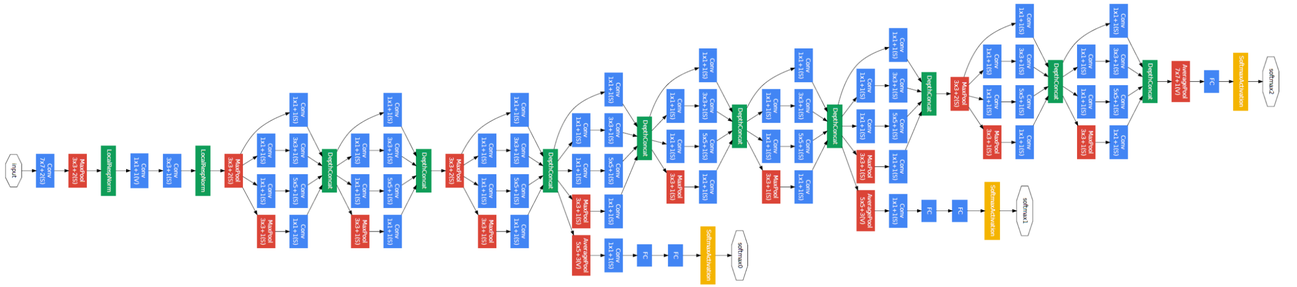
A history of ConvNNs

- *Neocognitron (1980): translation-invariant image processing with NNs*
- *LeNet (1989): considered the very first ConvNN, designer for digit recognition (ZIP codes)*
- *AlexNet (2012): the first big ConvNN (60M parameters, 650K neurtons), setting the state of the art: trained on GPUs, using ReLU and Dropout*
- *GoogleNet (2014): built on AlexNet, introduced an inception model to reduce e the number of parameters*



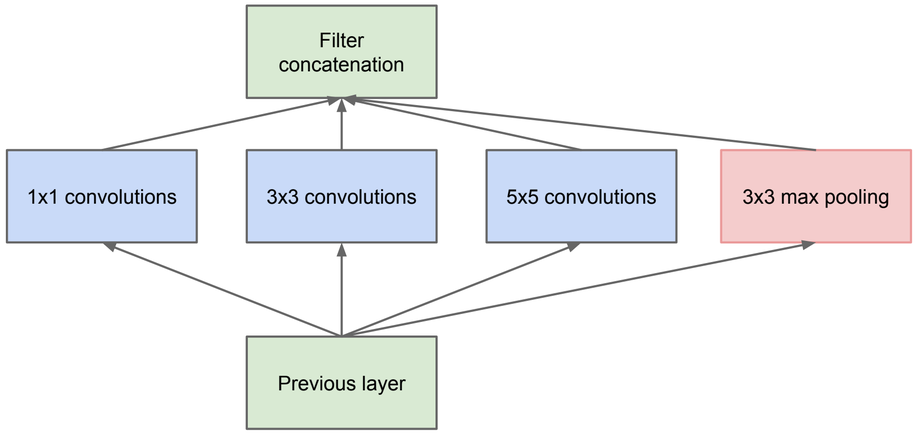




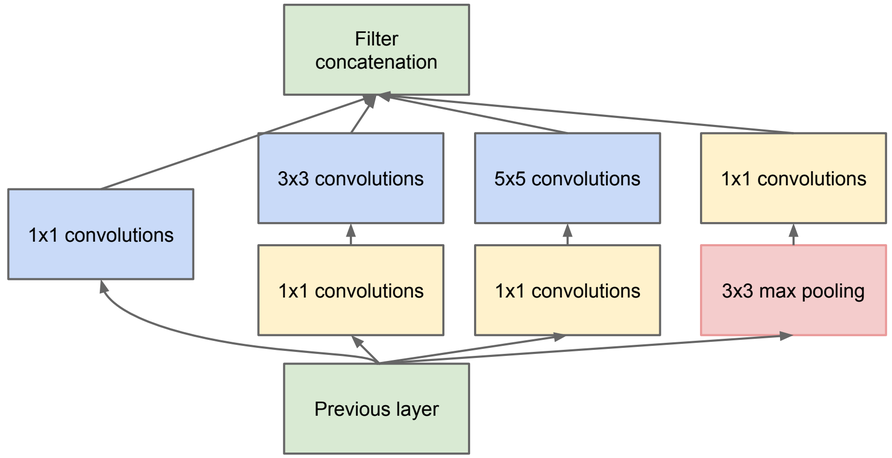


Inception Module

- *Rather than going deeper and deeper, inception architecture go wider*
- *Several conv layers, with different filter size, process the same inputs*
- *This way, more features can be detected from the same image*
- *The outcome of this parallel processing is then recombined through a concatenation step ass channels of an image*



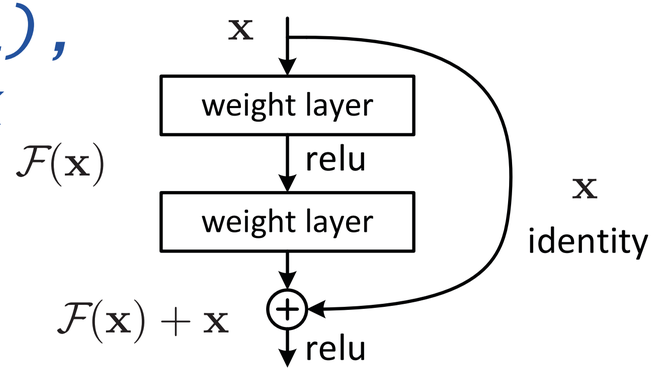
(a) Inception module, naïve version



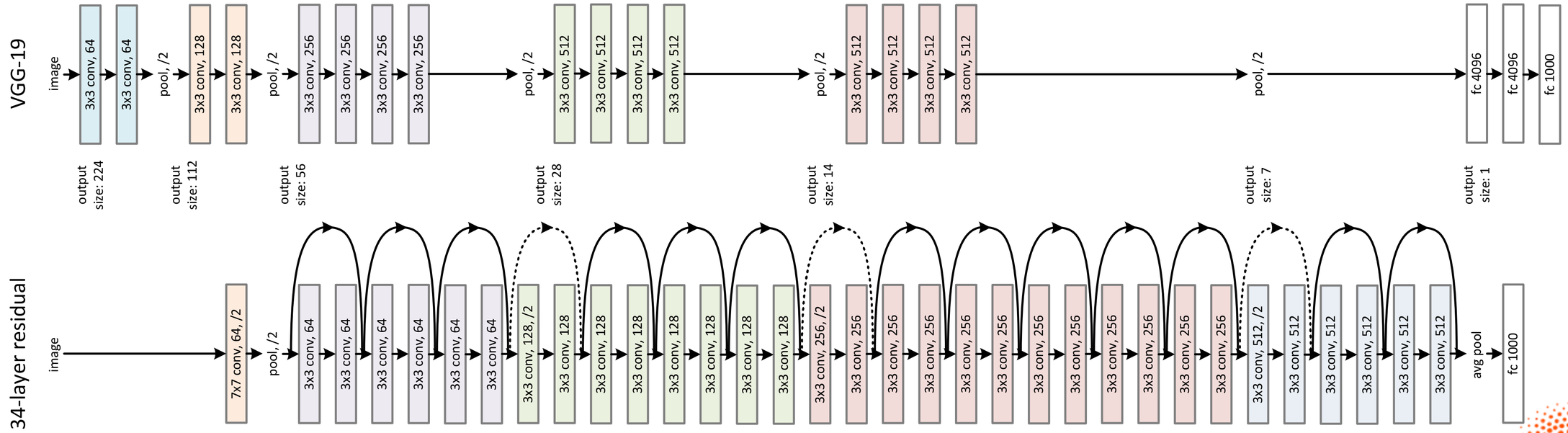
(b) Inception module with dimension reductions

A history of ConvNNs

- *VGGNet(2014):exploit small filters (3x3, 1x1), previously considered not optimal in a stack of Conv layers (rather than 1Conv+1Pooling)*



- *ResNet (2015):implemented skip connections, which were proven to boost performances*

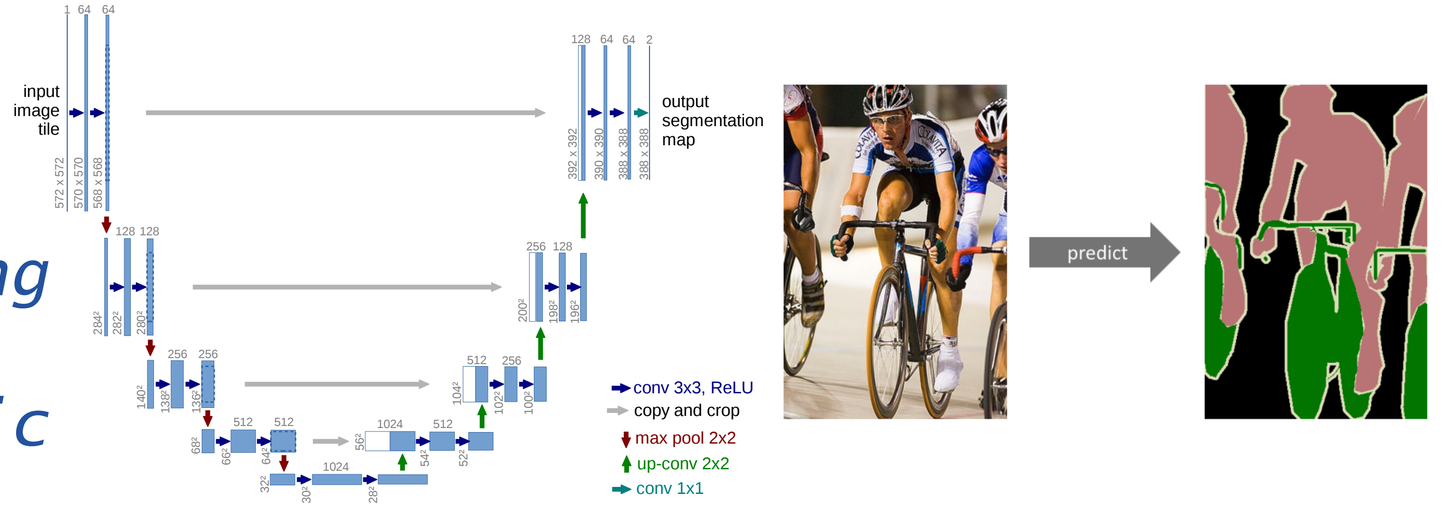


Check this blog entry on ConvNNs

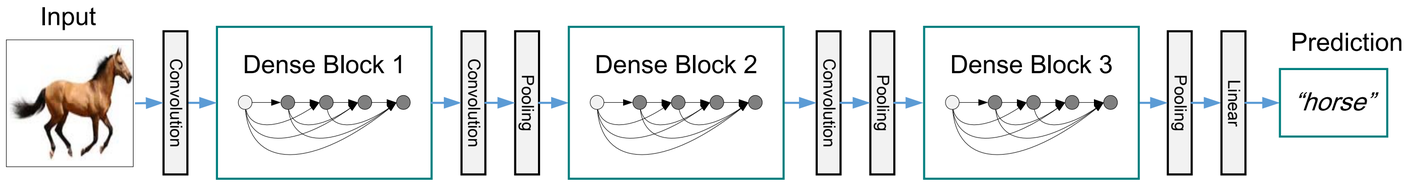
European Research Council

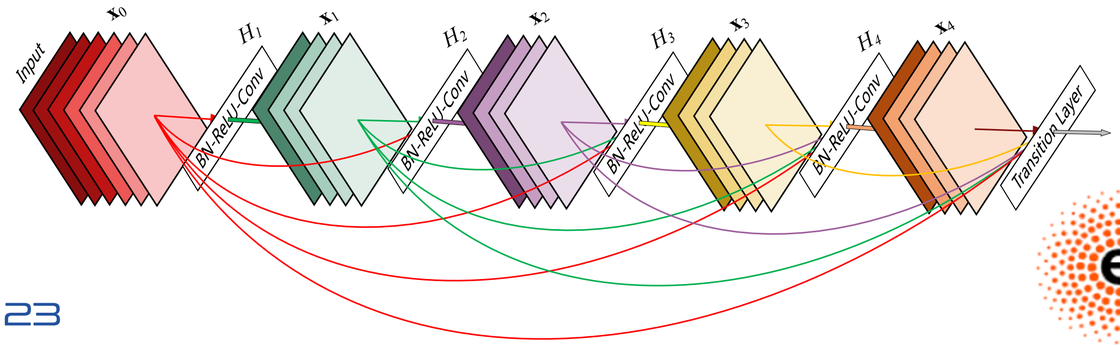
A history of ConvNNs

- *U-net (2015): conv layers with skip connections, in a downsampling+upsampling U-shaped sequence. Introduced for semantic segmentation*



- *DenseNet (2016): uses skip connections between a given layer and all the layers downstream in a dense block, with Conv layers in between*

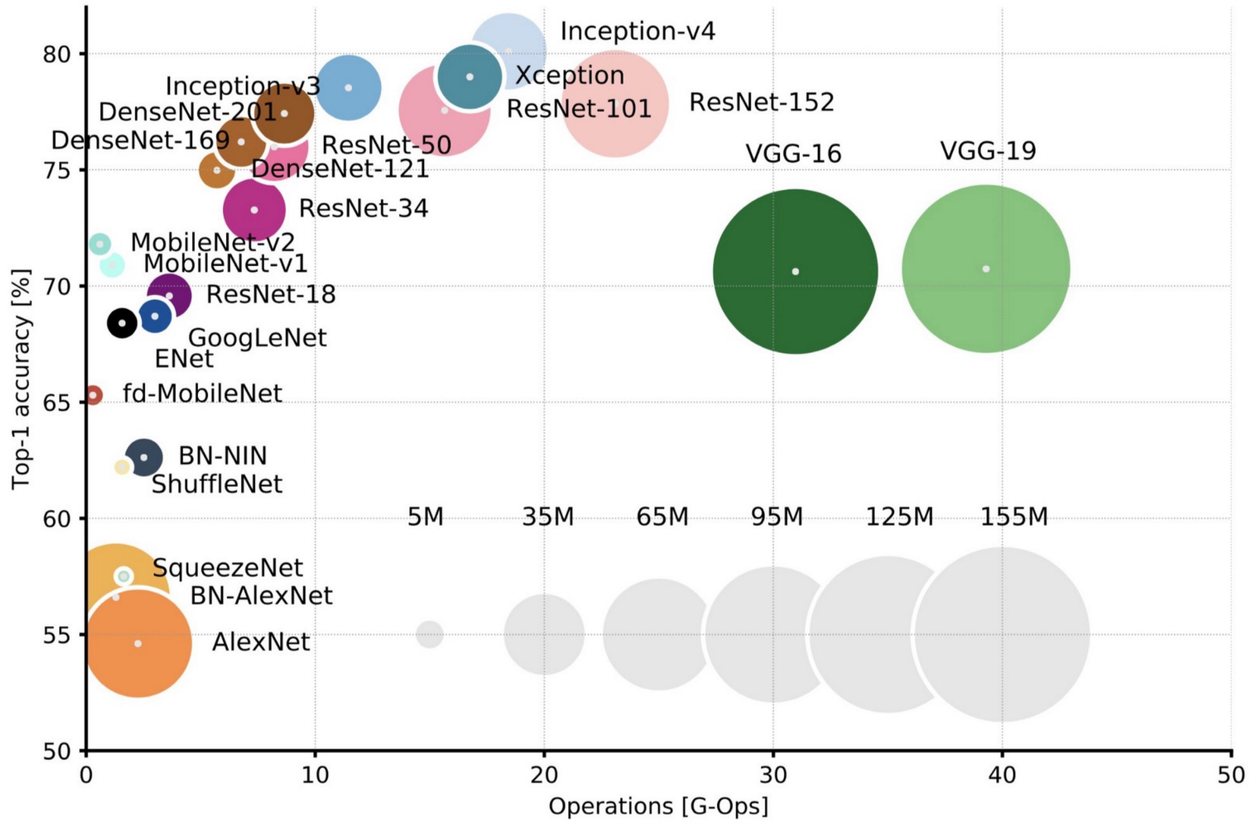




The computational cost

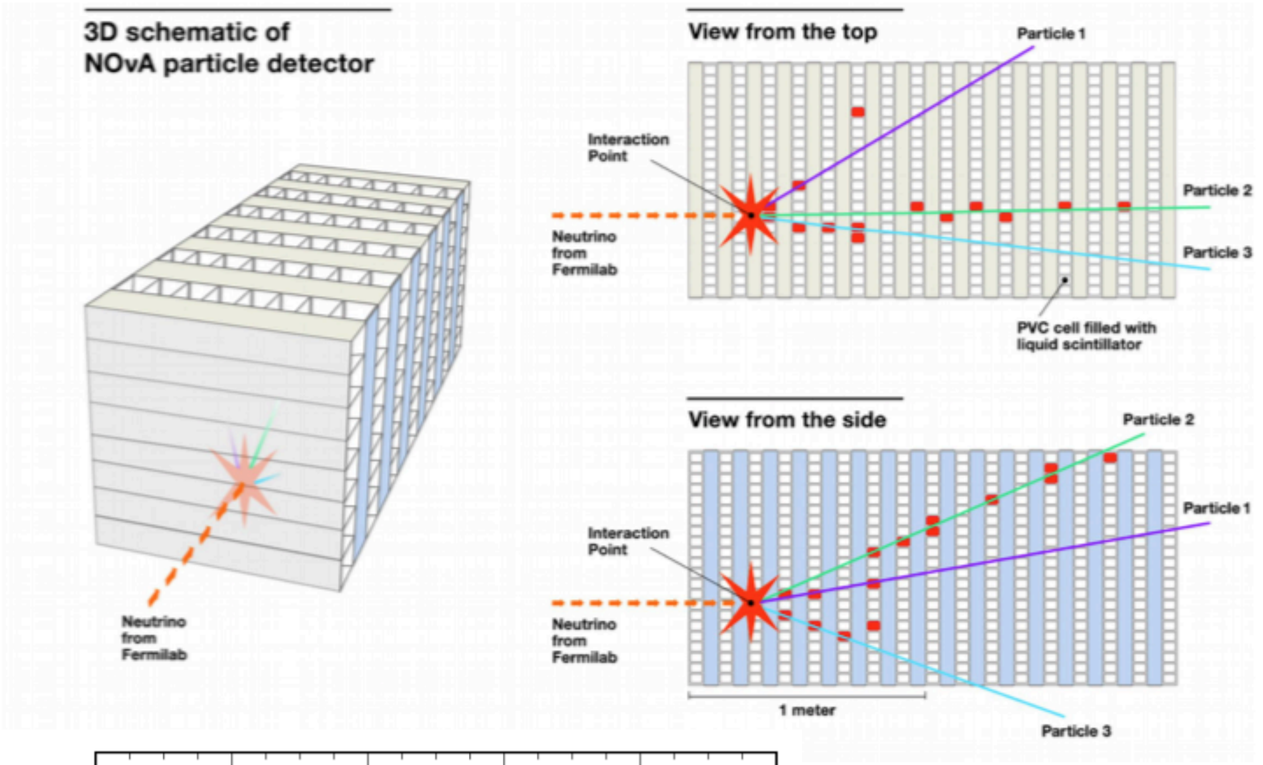
- *In this evolution, computing-vision networks drastically improved in performance*
- *This came at a big cost in complexity (number of parameters and operations)*
- *The inference with this network became particularly slow*
  - *big interest in optimize these networks on dedicated resources, e.g. FPGAs*
- *Many cloud providers provide optimized versions of these networks:*
  - *you can just use them (re-training if needed), rather than inventing your own one*

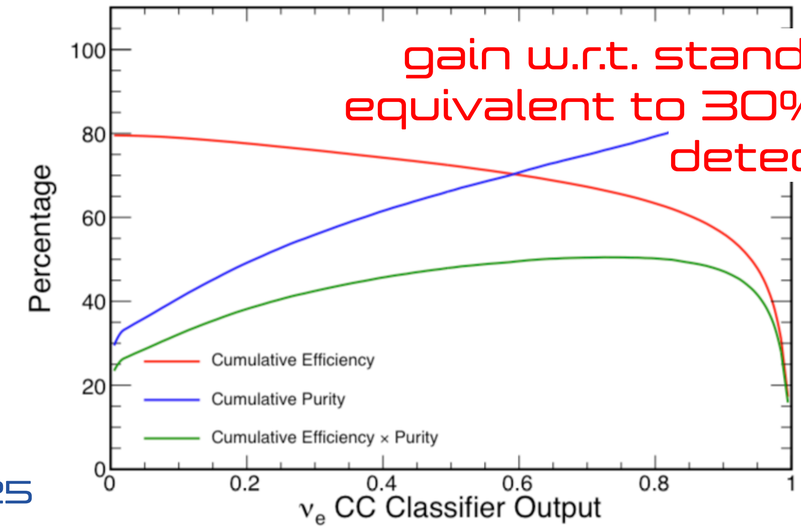
https://towardsdatascience.com/neural-network-architectures-156e5bad51ba



Example: PID for ν experiments

- *Many HEP detectors (particularly underground) more and more structured as regular arrays of sensors*
- *Modern computer-vision techniques work with images as arrays of pixel sensor (in 1D, 2D, and 3D)*
- *These techniques were applied by Nova on electron and muon ID*
- *Impressive gain over traditional techniques (comparable to +30% detector == $$$ saved)*



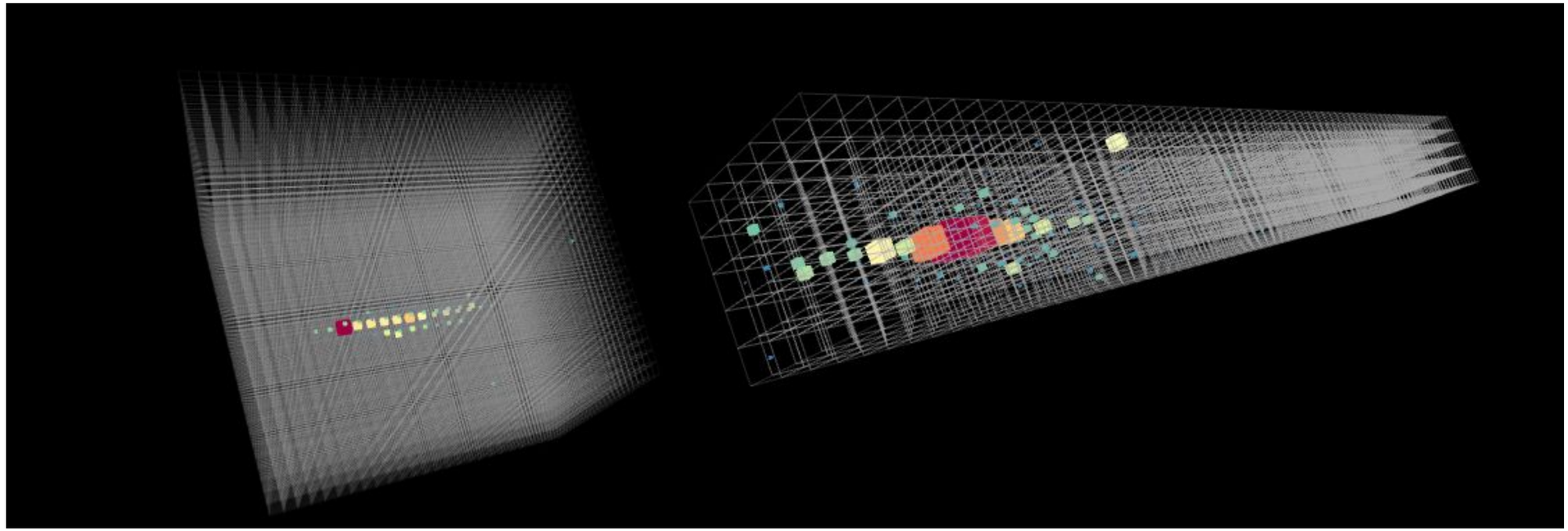


gain w.r.t. standard approach:
equivalent to 30% bigger mass of detector

European Research Council

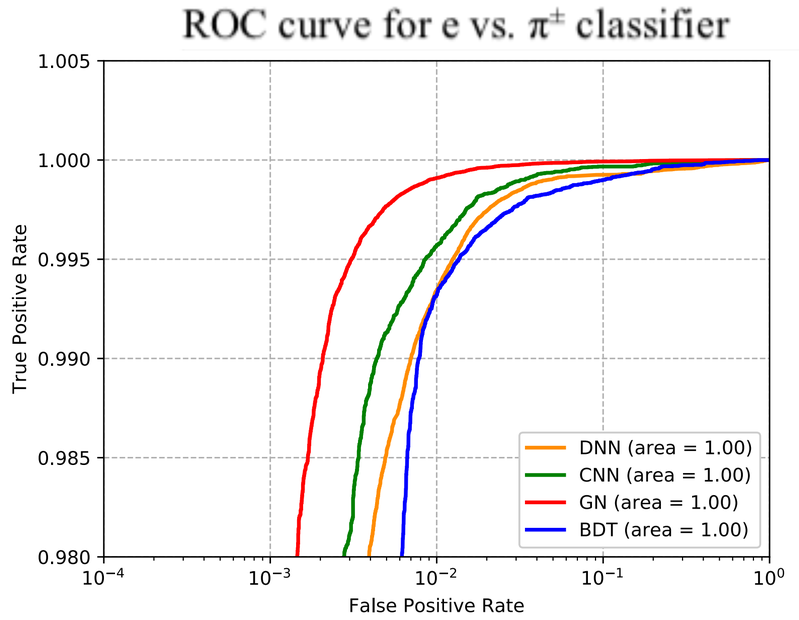
Calorimetry & Computer Vision

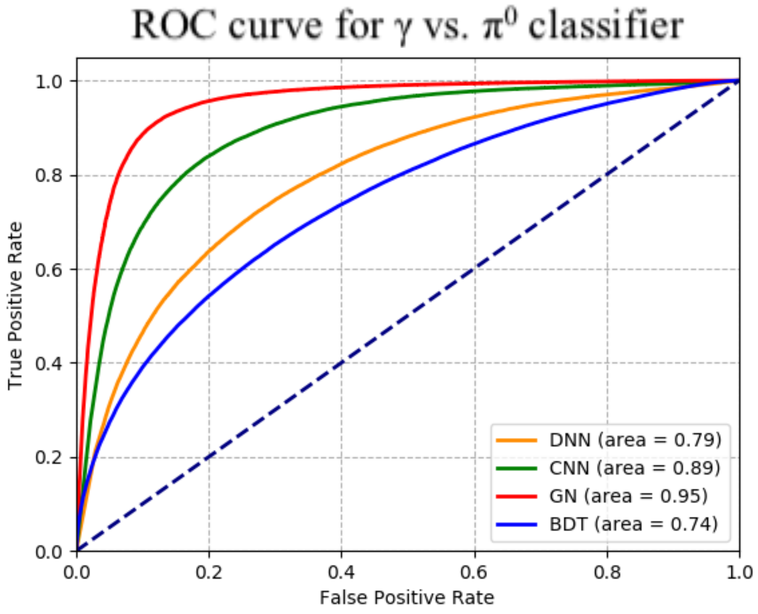
- *(next generation) digital calorimeters: 3D arrays of sensors with more regular geometry*
- *Ideal configuration to apply Convolutional Neural Network*
  - *speed up reconstruction at similar performances*
  - *and possibly improve performances*



Example: Particle ID

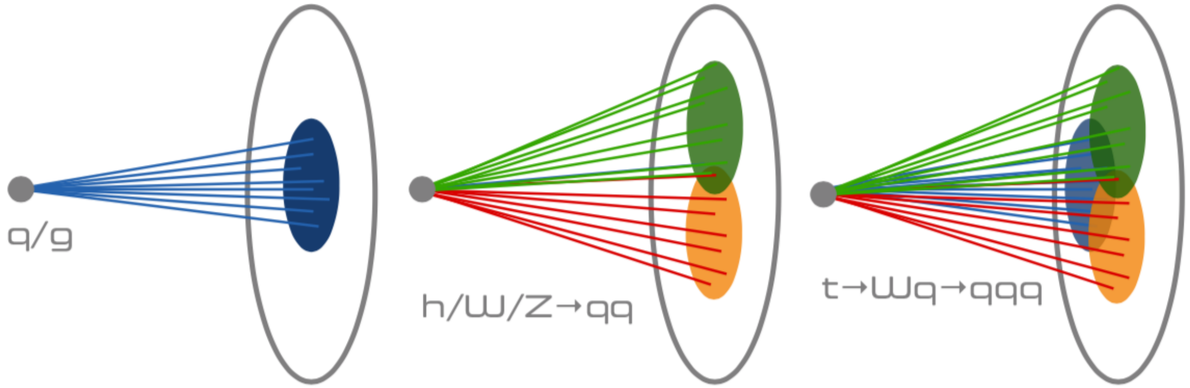
- We tried particle ID on a sample of simulated events
  - one particle/event (e, γ, π^0, π)
- Different event representations
  - high-level features related to event shape (moments of X,Y, and Z projections, etc)
  - raw data (energy recorded in each cell)
- Pre-filtered pion events to select the nasty ones and make the problem harder

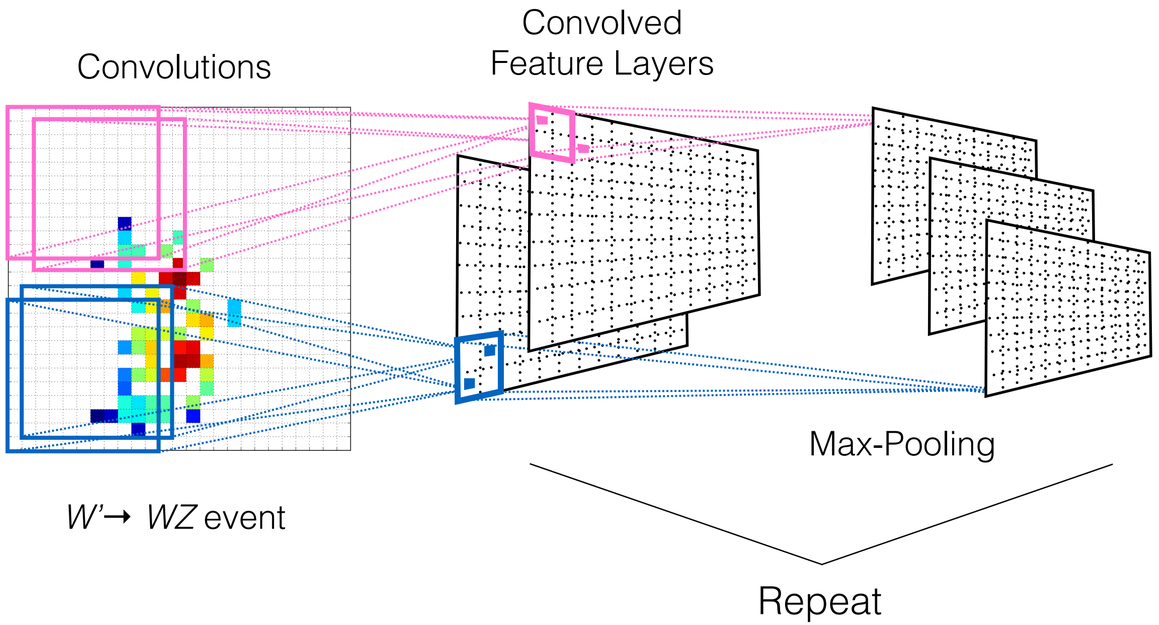




Jet as images (for ConvNN)

- *One can pixelate the surface crossed by the jet and create an image with the momentum deposited in each cell*
- *Such an image can then be processed with computing-vision techniques*
- *Pros: can benefit of the progresses made in optimizing computing vision*
- *Cons: underlying assumption on detector geometry (regular array of pixels) made sacrificing information of the actual detector*

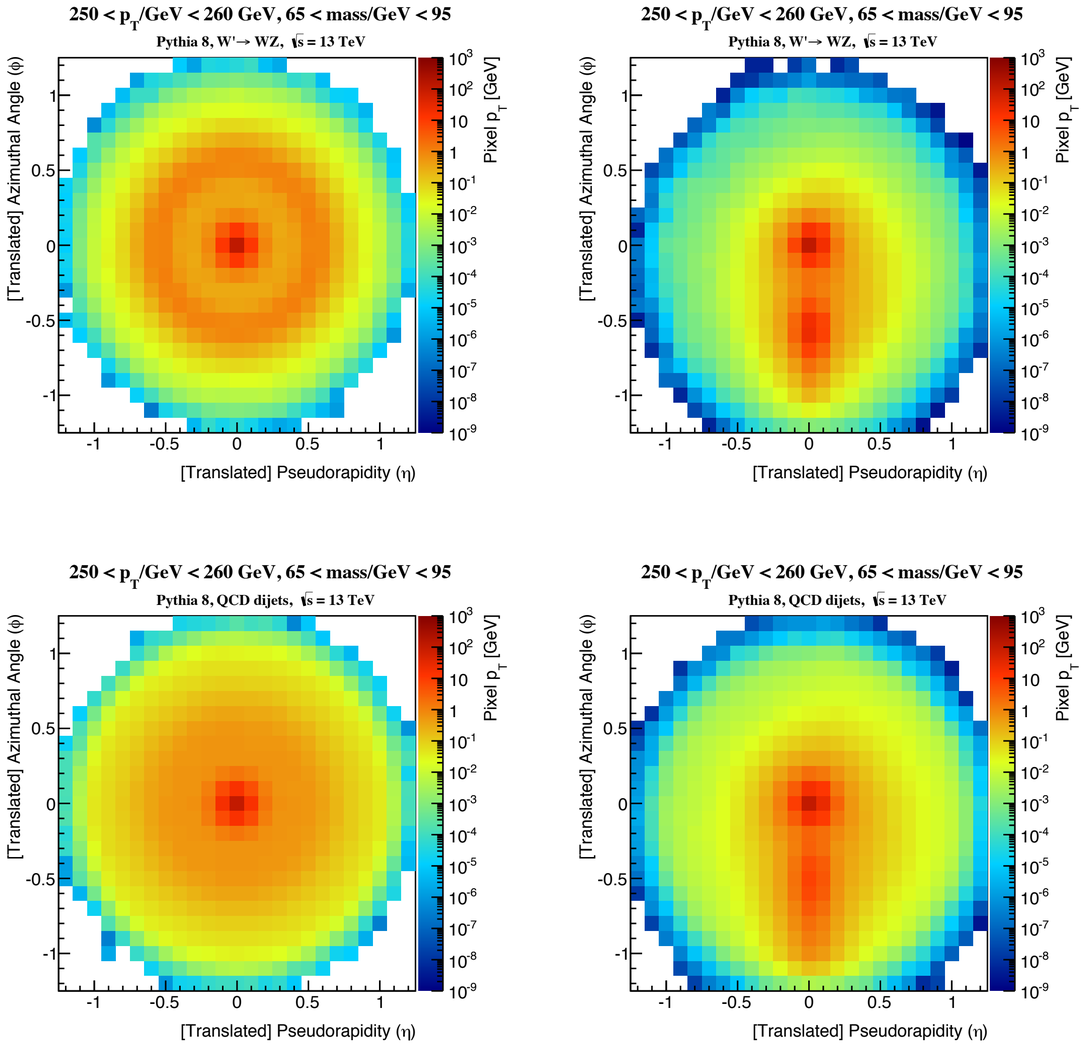




https://arxiv.org/abs/1511.05190

Jet as images (for ConvNN)

- *One can pixelate the surface crossed by the jet and create an image with the momentum deposited in each cell*
- *Such an image can then be processed with computing-vision techniques*
- *Pros: can benefit of the progresses made in optimizing computing vision*
- *Cons: underlying assumption on detector geometry (regular array of pixels) made sacrificing information of the actual detector*



https://arxiv.org/abs/1511.05190

Summary

- *Conv Networks are the most striking example of the success of Deep Learning*
  - *A story of improvement from the 90s to now*
  - *Very effective as a fast tool for particle reconstruction*
  - *Same geometry can be used for image generation with adversarial training*
  - *Can speed up one of the heaviest tasks in particle physics*