Neural Networks:
Theory and Practice
Lecture II

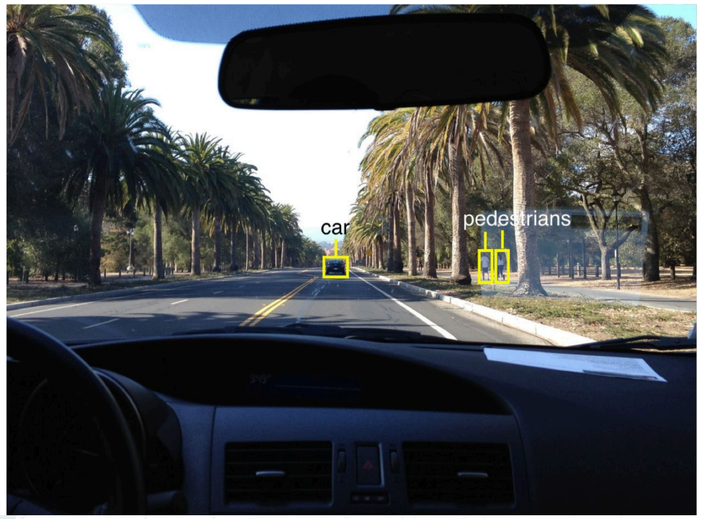
Nicoletta Krachmalnicoff
SISSA - Astrophysics and Cosmology PhD school
December, 2019

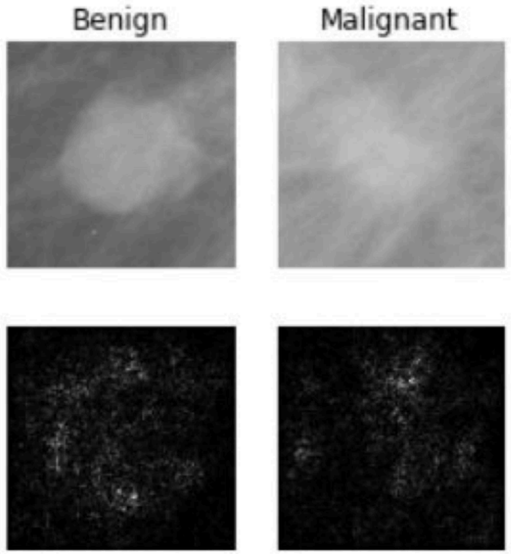
Outline lecture 2

- Convolutional Neural Networks:
  - Basic concepts
  - Hyperparameters
  - Architectures
- **Tutorial 2: Galaxy classification with CNNs**
- Generative Adversarial Netwoks:
  - Architecture
  - Training
- **Tutorial 3: CMB maps with GANs**
- NN applications in Cosmology

Convolutional Neural Networks (CNN)

- Convolutional NNs have **similar structure as fully-connected ones**: they are made of neurons with learnable weights, each neuron perform a product and it is then activated with a non-linear function
- CNN make the **explicit assumption that inputs are images**, allowing to encode certain properties in the NN architecture
- Widely used for image recognition is the most diverse fields

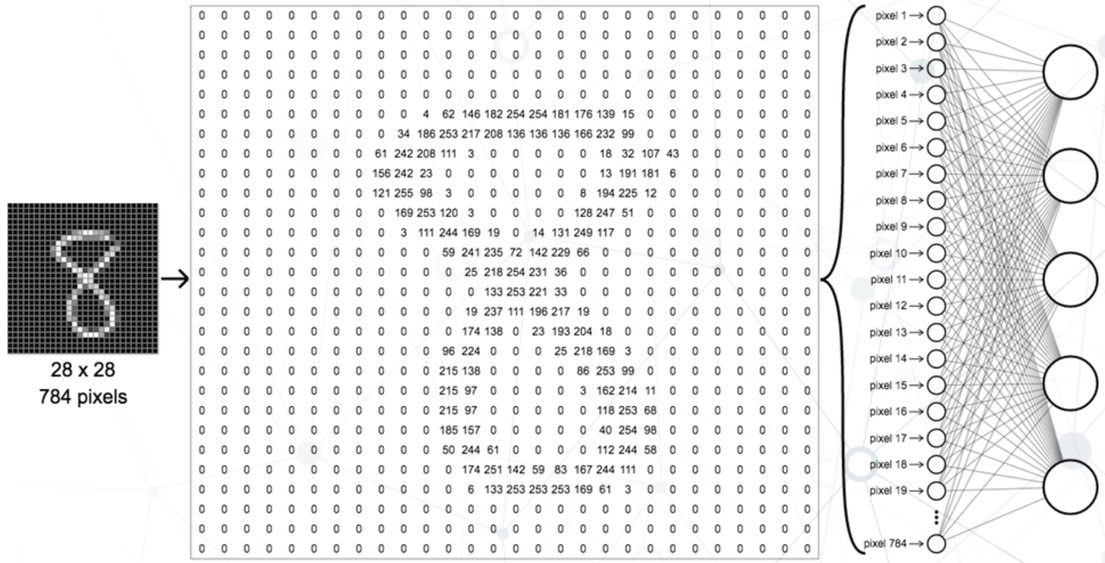




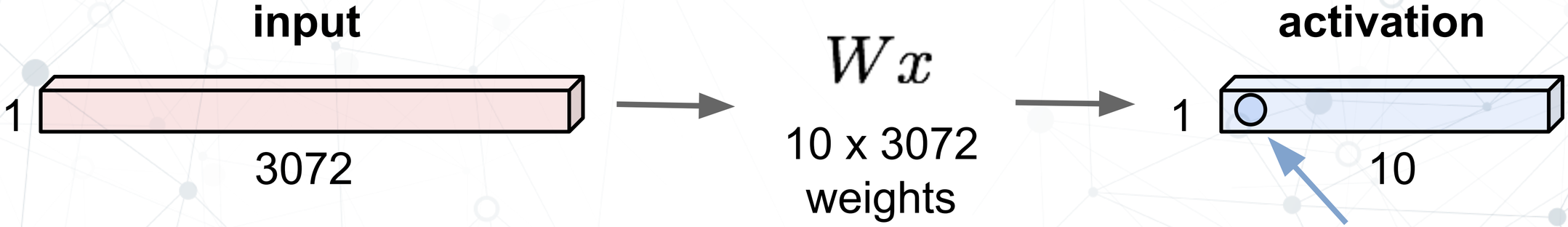
CNN: basic concepts

- We want to recognize handwritten digits (0-9) on images of size 32x32x3 pixels

- With a fully connected NN we would:
  i) flatten the image into a vector 3072x1
  ii) connect all the elements of this vector with an output layer with 10 neurons (one per class)
  iii) apply activation



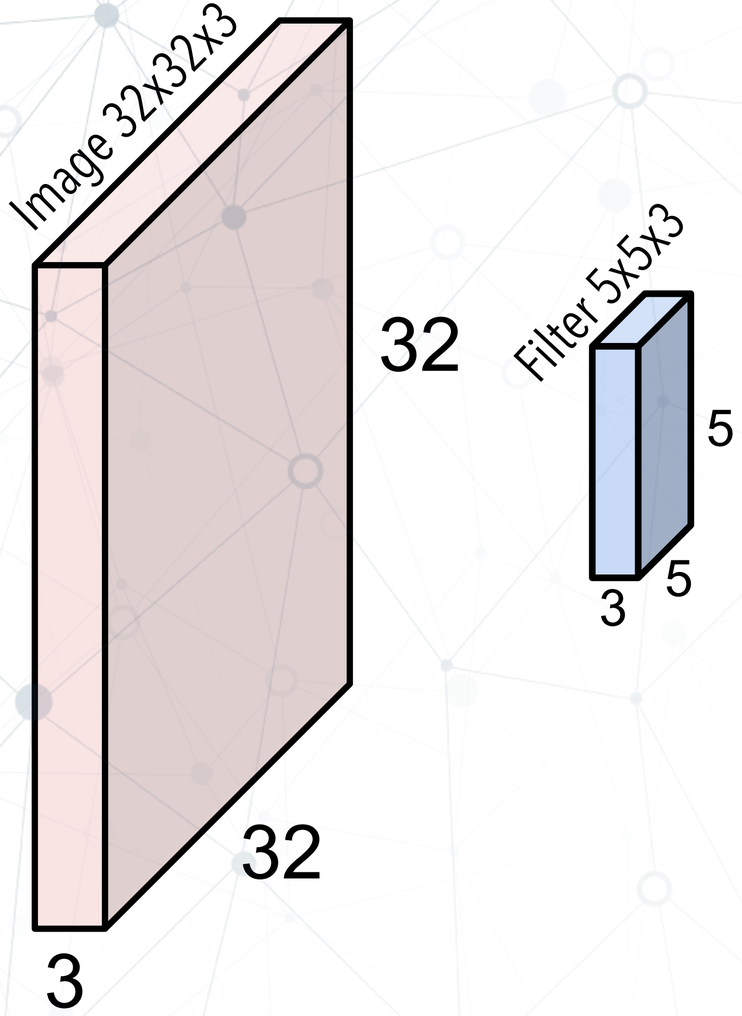
- the matrix W of learnable weights would have dimension $(10, 3072)$



Each neuron is the results of taking the dot product between the input vector and a row of the W matrix

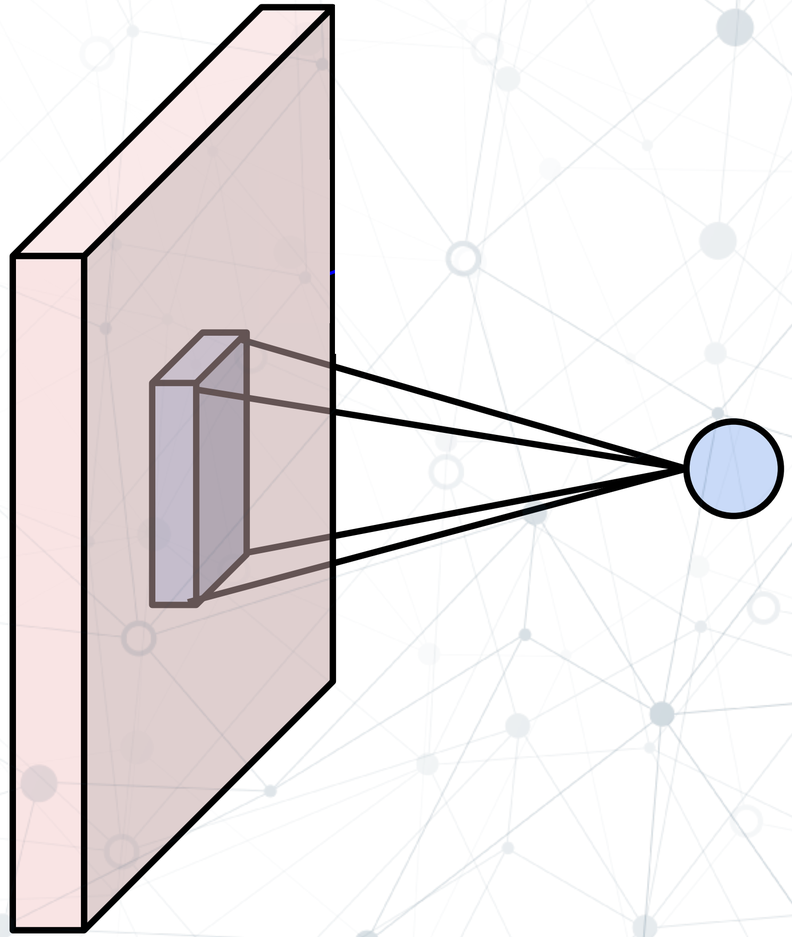
CNN: basic concepts

- In a CNN **the spatial structure of the input images is preserved**



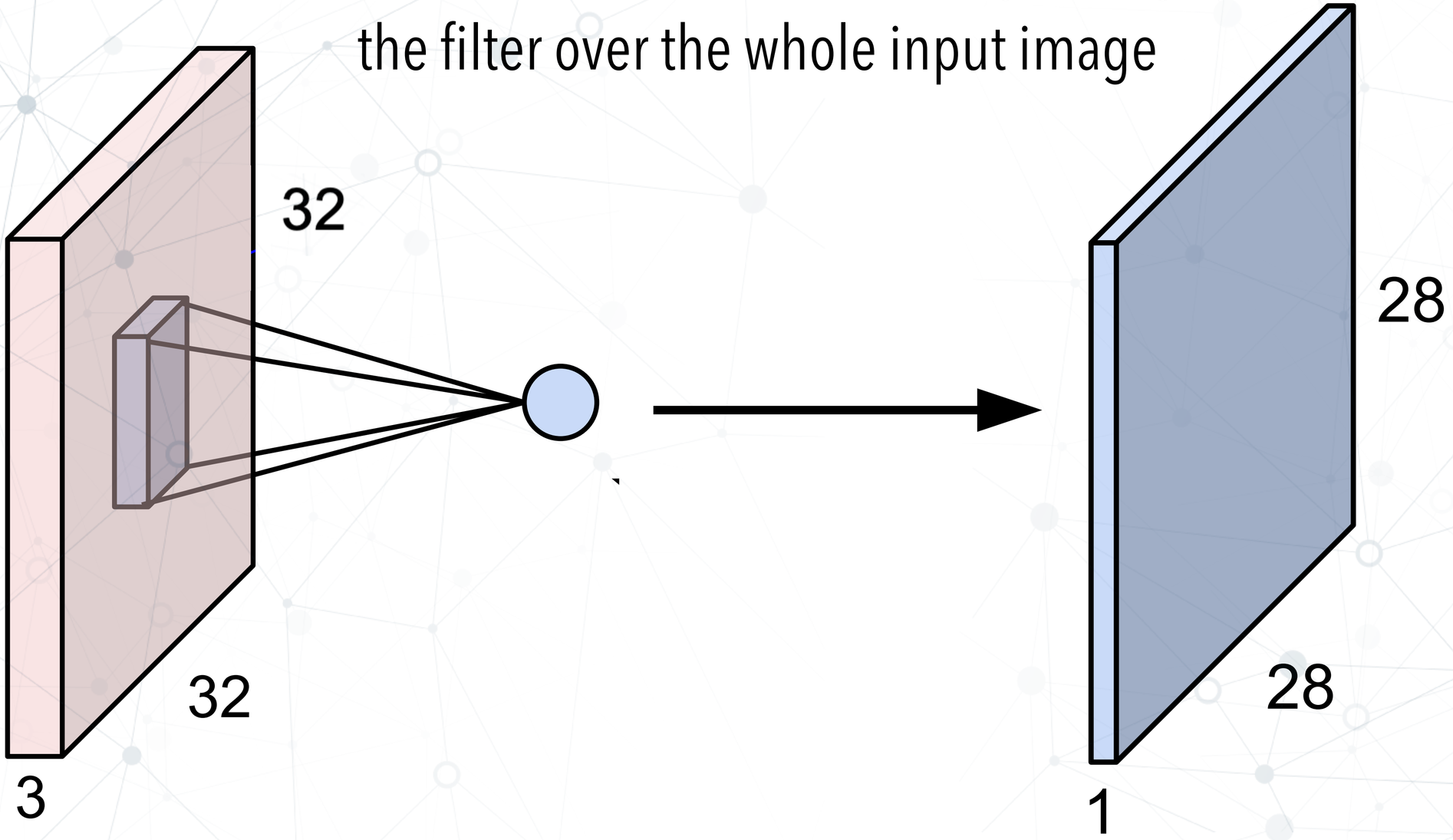
The covolution between sub-image (5x5x3) and filter is computed as the sum of the element wise product between the two

$$\sum_i k_i w_i$$



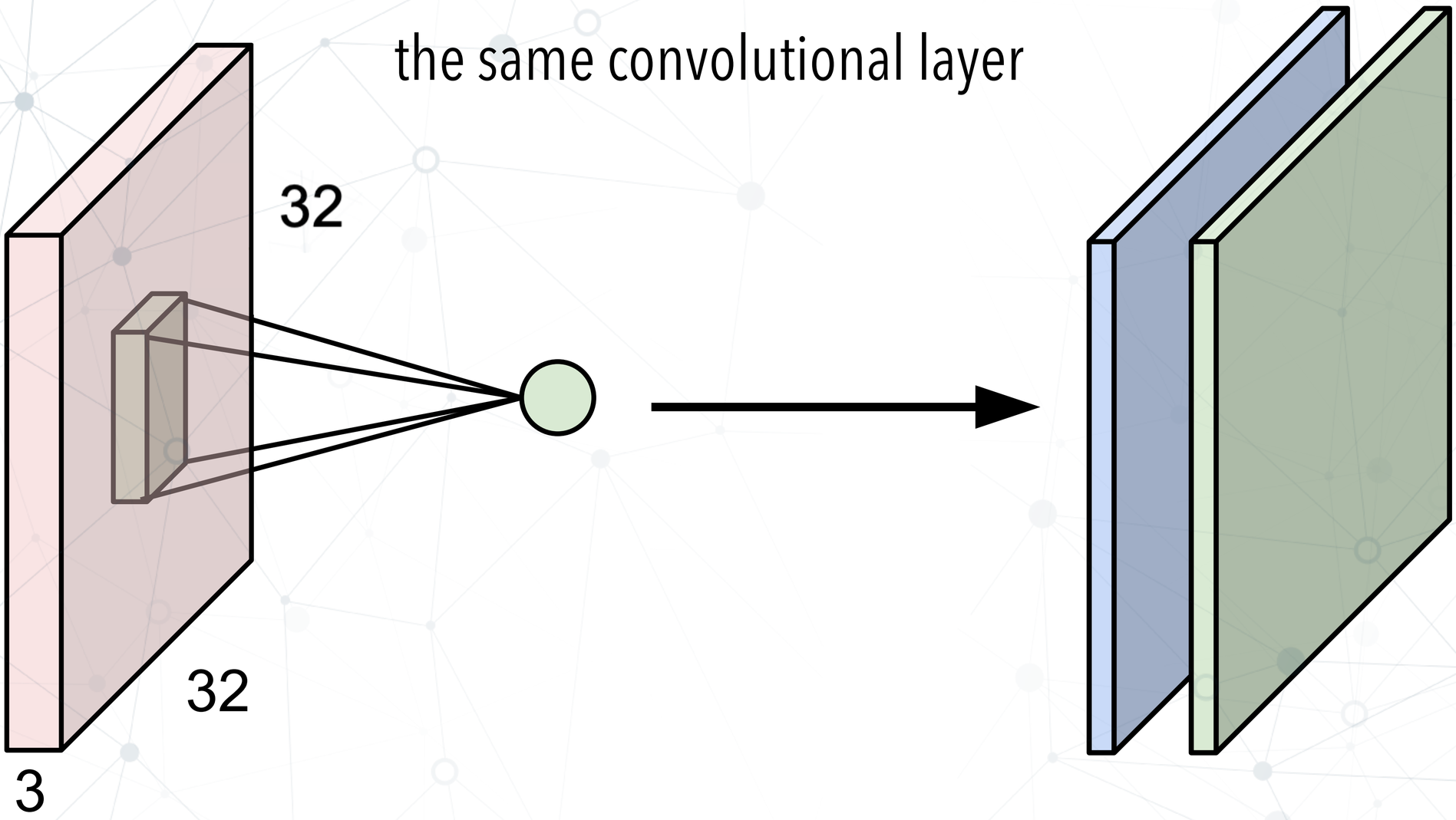
CNN: basic concepts

The full convolution is done by slicing the filter over the whole input image



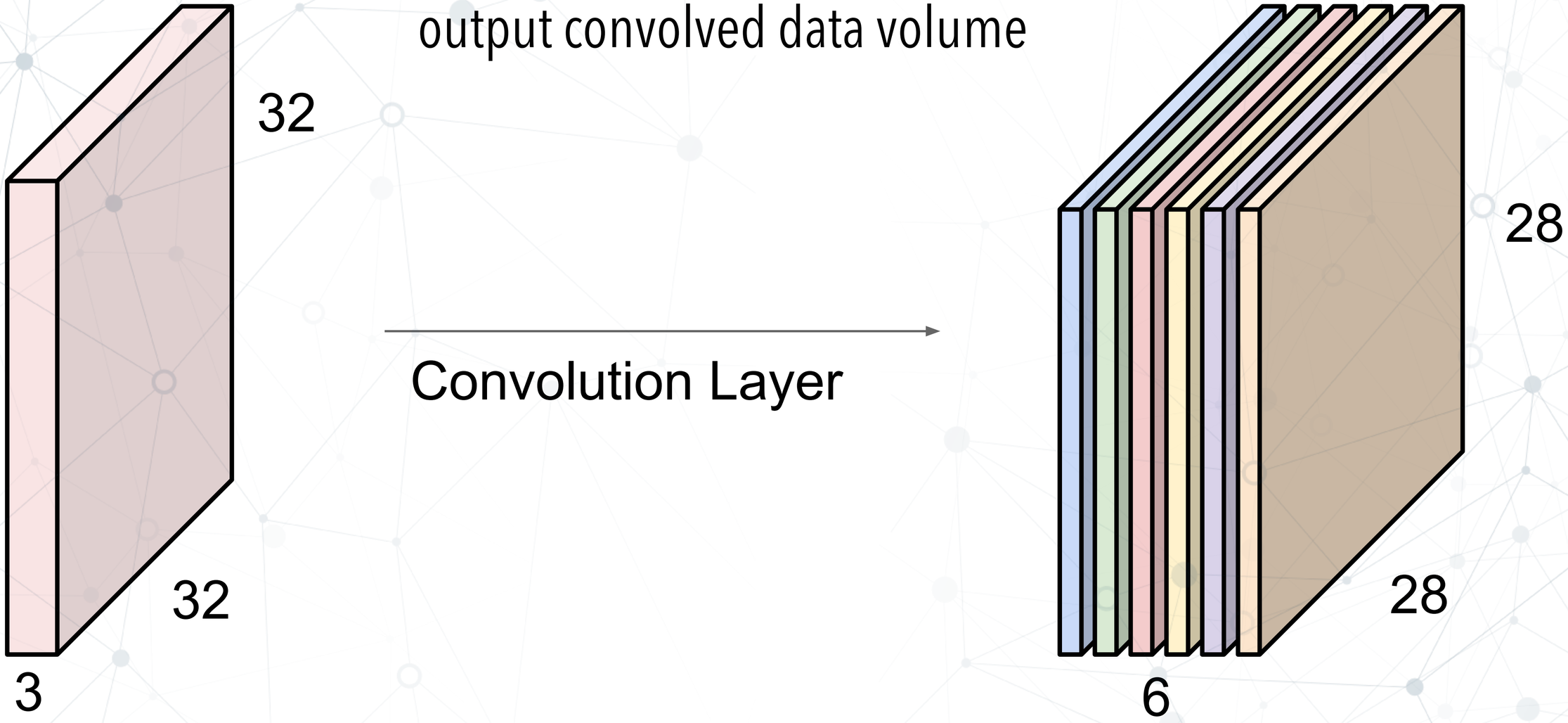
CNN: basic concepts

Different filters can be used in the same convolutional layer



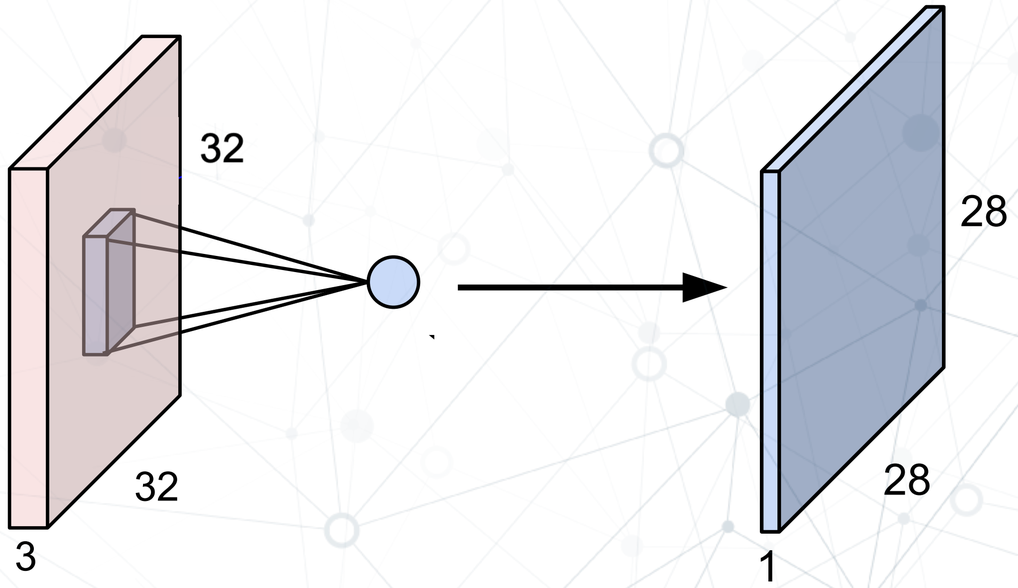
CNN: basic concepts

Generating in this way the output convolved data volume



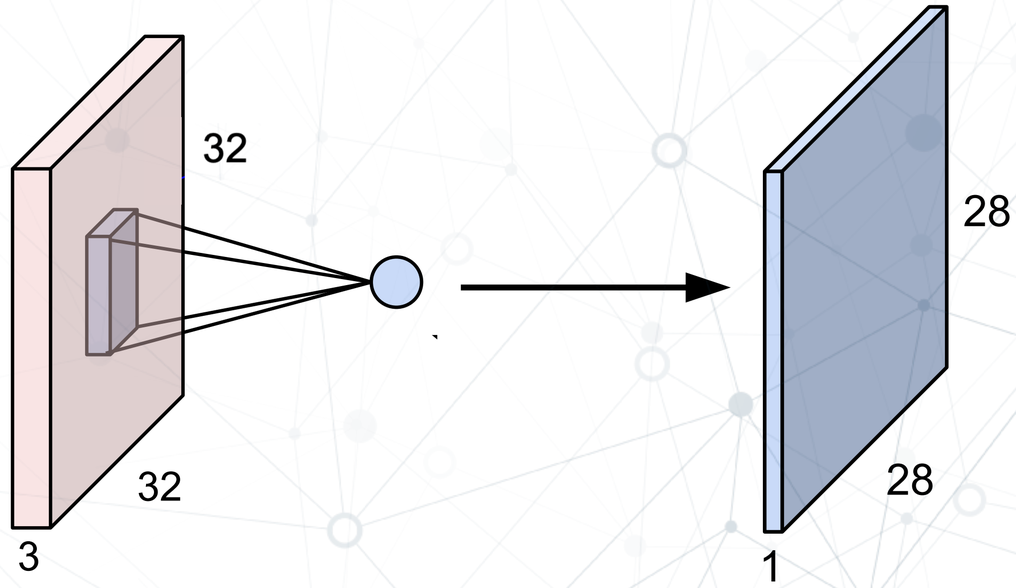
CNN: basic concepts

- **Each element of filters** represents a learnable **weight** of the CNN
- Filters must always have the **same depth** as the volume (image) on which convolution is performed
- **Each element of the ouput volume** in a convolutional layer represents a **neuron** of the CNN which is then activated



CNN: basic concepts

- **Each element of filters** represents a learnable **weight** of the CNN
- Filters must always have the **same depth** as the volume (image) on which convolution is performed
- **Each element of the ouput volume** in a convolutional layer represents a **neuron** of the CNN which is then activated
- Each neuron in a CNN is therefore only connected with a subset of the elements of the previous layer, called the **receptive field**

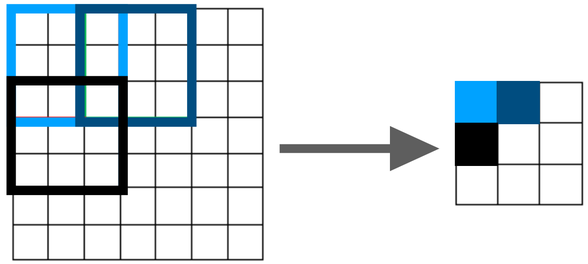


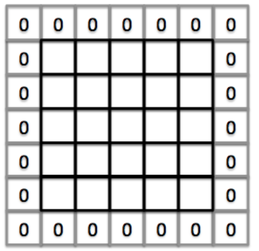
- Note that **filter weights do not change while slicing the image**, making therfore the intrisic assumption that if a certain feature is important in the position (x, y) it would be similarly important in position (x', y')

CNN: hyperparameters

- Four hyperparameters control the dimension of the output volume and need to be set in a convolutional layer

1. **Dimension of the receptive field** F: is defined by the dimension width and height of filters
2. **Number of filters** N**:** this define the depth of the output volume. Each filter, du learn to select a different feature in the data
3. **Stride** S**:** define how the filters slice the input image. A slide of 1 means that filters move of one pixel at each step, a larger stride cause the output volume to be smaller (in terms of width and height)
4. **Zero Padding** P**:** to control the spatial size of the output volume it can be useful to pad with zero around the borders the input image

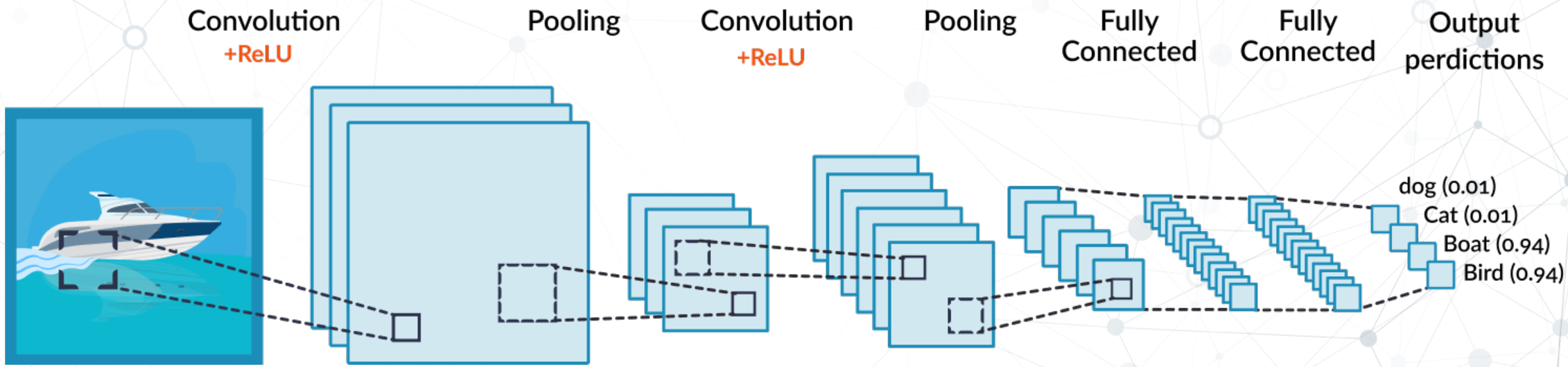




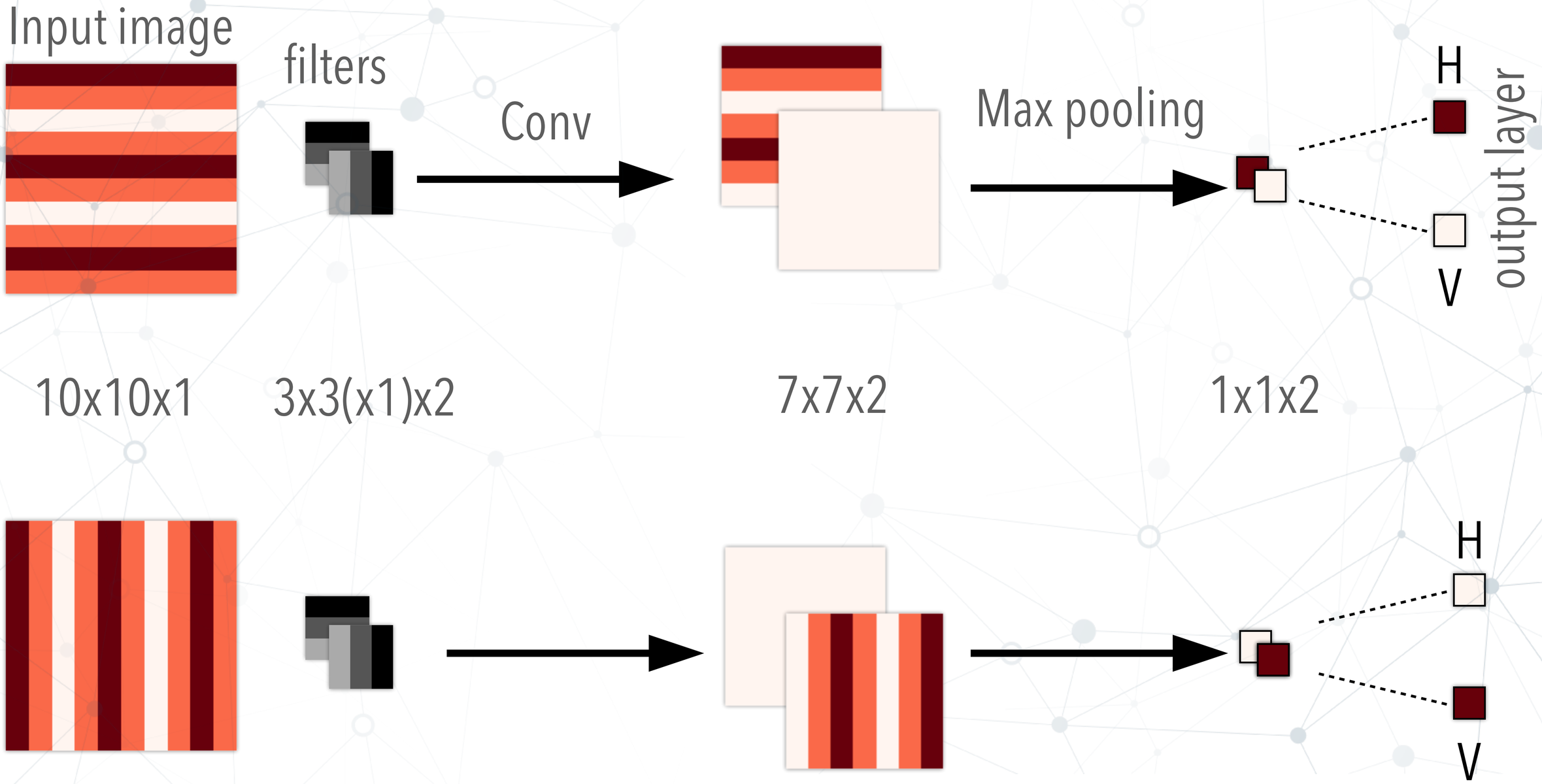
$$W^{out} = \frac{W^{in} - F - 2P}{S} + 1$$

CNN: complete architecture

- A deep CNN is built by stacking together several **convolution+pooling layers followed by fully connected layers**
- **Pooling layers are used to reduce dimensionality** in (width x height) and usually they consist in taking the maximum or average values of pixels in the pooling reception field
- Typically while gooing deeper in the network (width x height) dimension is reduced while depth of the volume grows

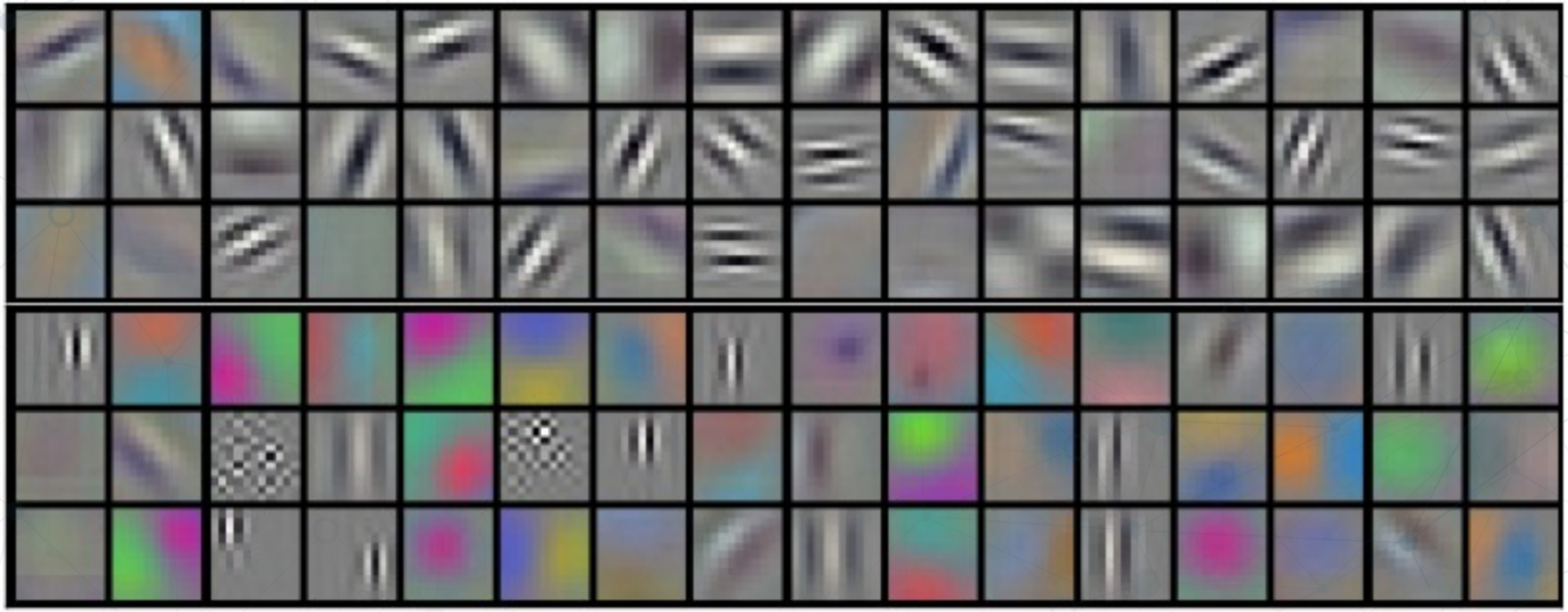


CNN: a very stupid example

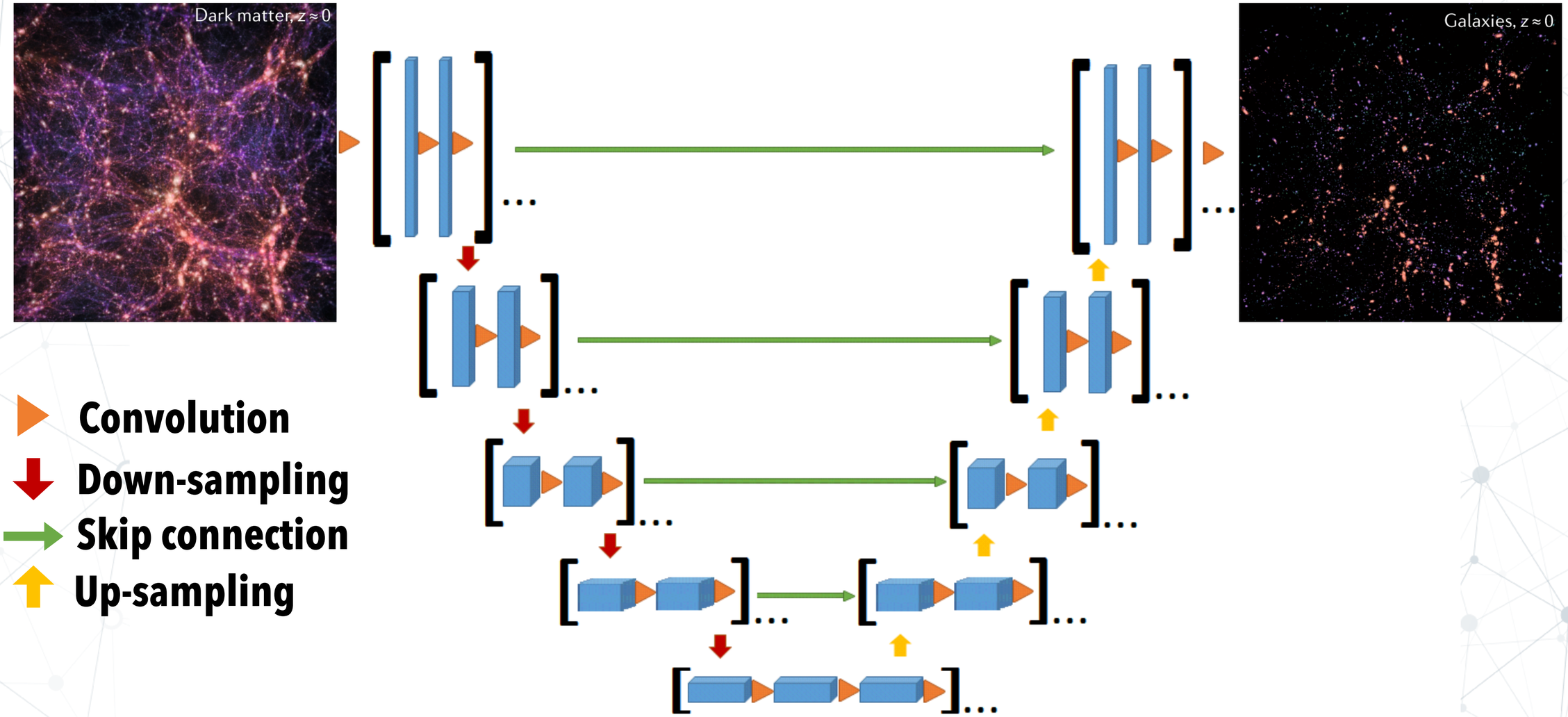


CNN: filters

- During training the CNN learns the weights of all its filters, which extract the relevant features from the inputs
- A well trained CNN will learn "meaningful and not noisy" filters



Example: Unet



Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597

Tutorial 2

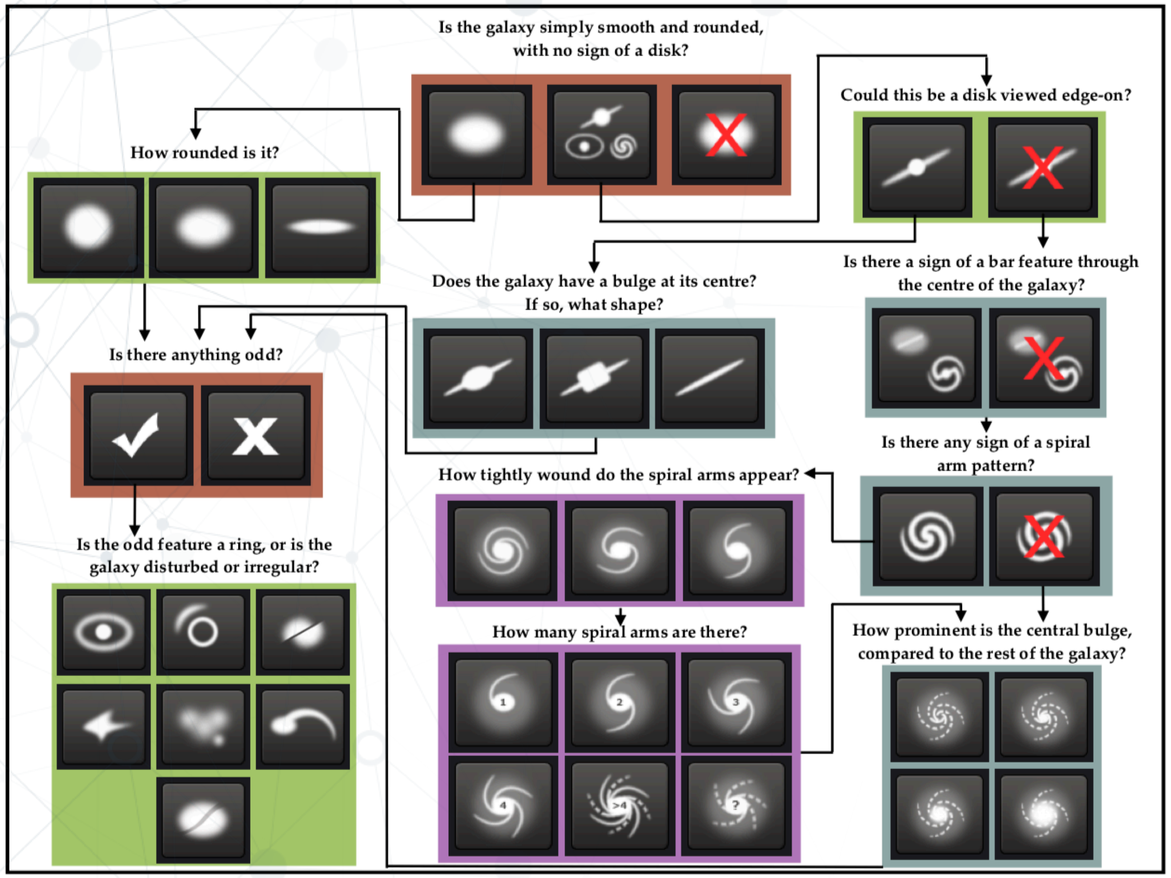
CNN for Galaxy classification

The Galaxy Zoo Challenge

https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge

http://arxiv.org/abs/1308.3496

- Citizen science project with more than **16 million morphological classifications of about 60,000 galaxies** drawn from the Sloan Digital Sky Survey
- Whole dataset of galaxy images and answer statistics is public with machine learning
- The original challenge was match the answer distribution on all the 37 questions

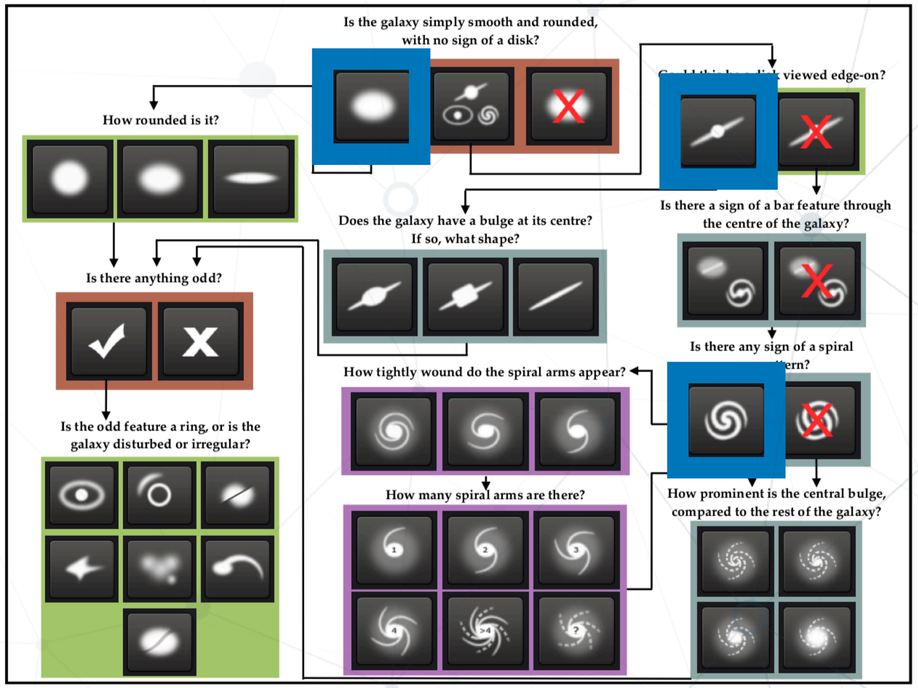


Task	Question	Responses	Next
01	*Is the galaxy simply smooth and rounded, with no sign of a disk?*	smooth features or disk star or artifact	07 02 **end**
02	*Could this be a disk viewed edge-on?*	yes no	09 03
03	*Is there a sign of a bar feature through the centre of the galaxy?*	yes no	04 04
04	*Is there any sign of a spiral arm pattern?*	yes no	10 05
05	*How prominent is the central bulge, compared with the rest of the galaxy?*	no bulge just noticeable obvious dominant	06 06 06 06
06	*Is there anything odd?*	yes no	08 **end**
07	*How rounded is it?*	completely round in between cigar-shaped	06 06 06
08	*Is the odd feature a ring, or is the galaxy disturbed or irregular?*	ring lens or arc disturbed irregular other merger dust lane	**end** **end** **end** **end** **end** **end** **end**
09	*Does the galaxy have a bulge at its centre? If so, what shape?*	rounded boxy no bulge	06 06 06
10	*How tightly wound do the spiral arms appear?*	tight medium loose	11 11 11
11	*How many spiral arms are there?*	1 2 3 4 more than four can't tell	05 05 05 05 05 05

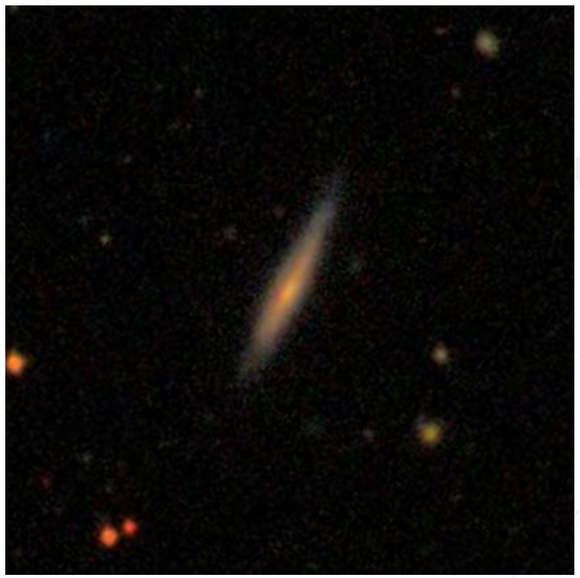
Galaxy classification with CNN

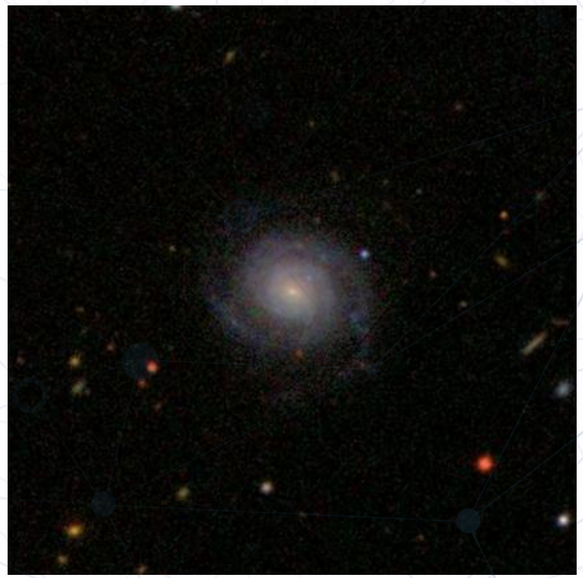
- Simplified version of the dataset with **3 classes:**

1. Round
2. Edge-on
3. Spiral



- Each galaxy is assigned to a class if at least 90% of aswers were positive
- Total of **6167 images**
- 80% training, 10% validation, 10% test





Generative Adversarial Networks (GAN)

- A generative adversarial NN is a **system of two separate NNs that compete against each other**
- Given a training set **a GAN learns to generate new sets of data with the same "statistics" as the training set**
- The original paper (Goodfellow et al. 2014, https://arxiv.org/abs/1406.2661) has more than 13000 citations

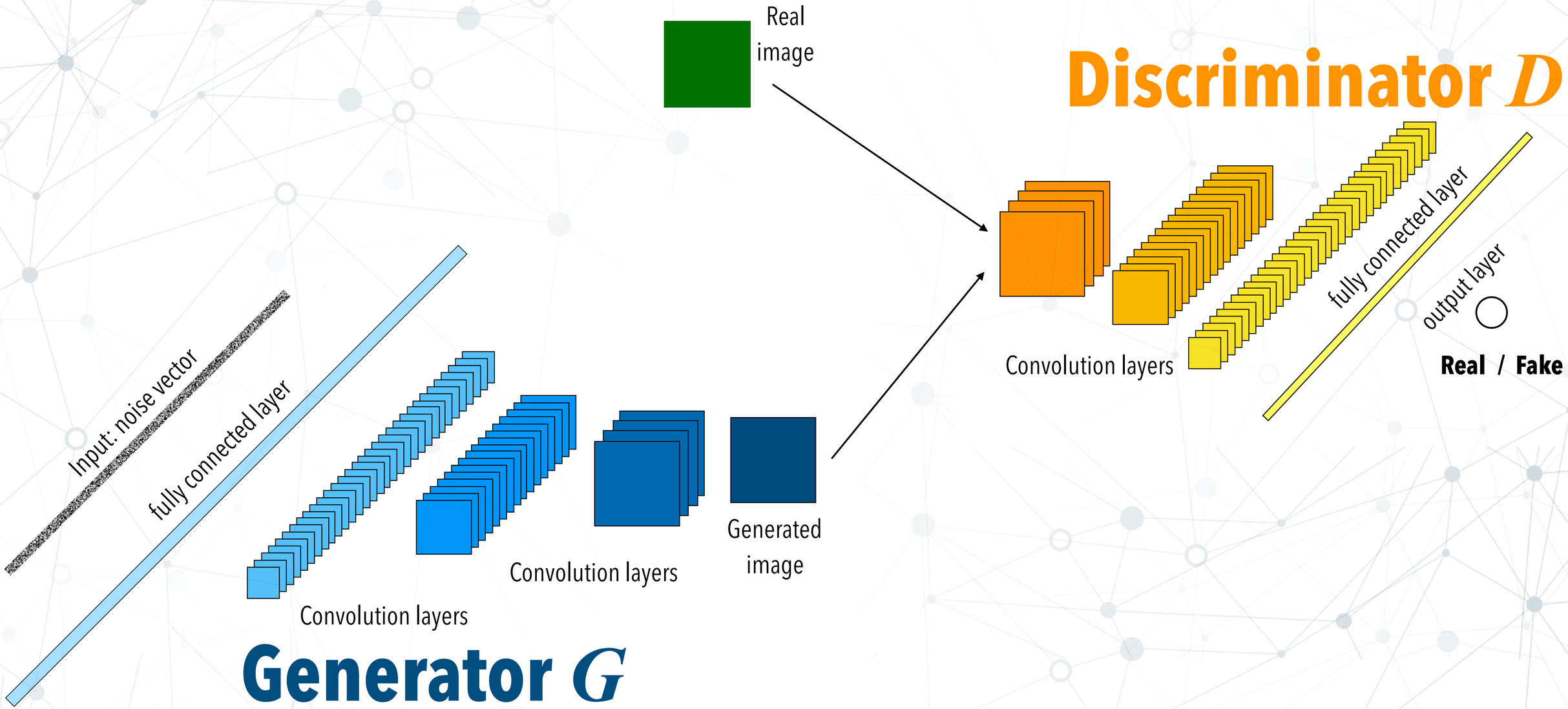
Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

GAN: architecture

Deep Convolutional GAN (DCGAN):

Radford et al. 2015
https://arxiv.org/abs/1511.06434



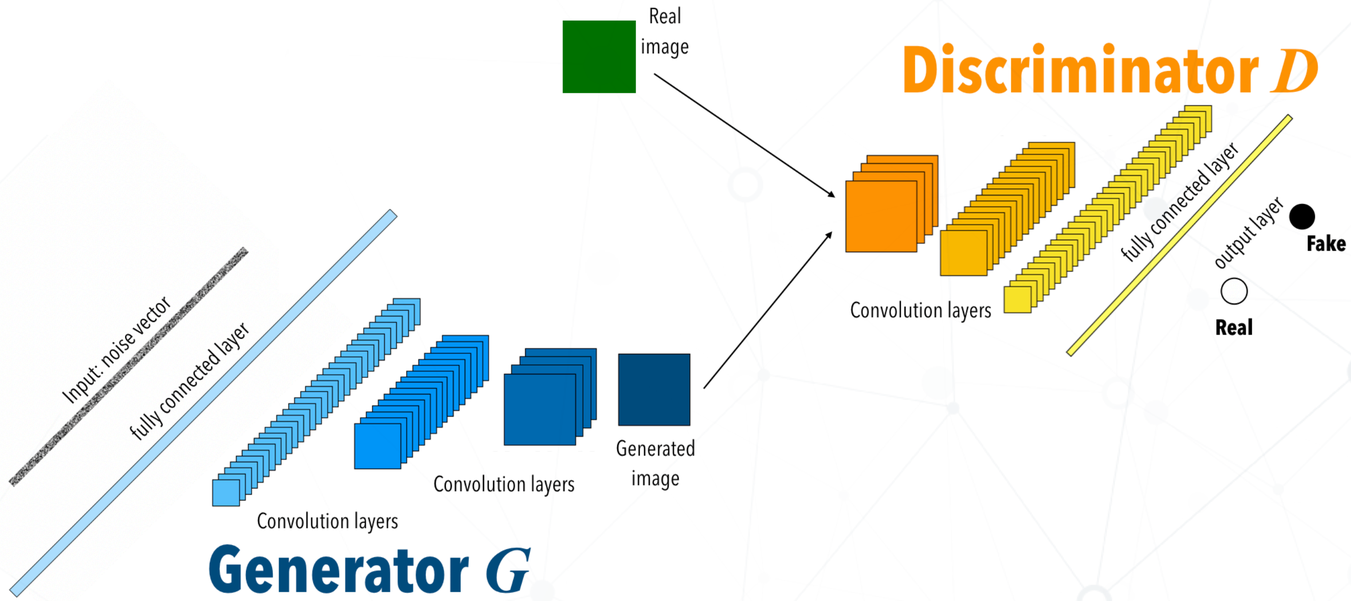
GAN: training

N : number of elements in the mini-batch

z^i : i-th noise vector which is the input of G

x^i : i-th real image

$G(z^i)$: output of the generator for z^i



Discriminator D:

- D receives as input **real images** (labeled as ***1***) and **fake images** generated by G (labeled as ***0***)
- its goal is therefore to output *1* when input is x and *0* when input is $G(z)$
- during training it aims at minimizing the following cost function:

$$\mathcal{J}_D = -\frac{1}{N}\sum_i^N \log\left[D(x^i)\right] + \log\left[1 - D\left(G(z^i)\right)\right]$$

in minimizing this cost function weights of G are fixed while the training optimizes the weights of D

Discriminator G:

- its goal is to produce images that mislead *D*
- it aims at minimizing

$$\mathcal{J}_G = -\frac{1}{N}\sum_i^N \log\left[D\left(G(z^i)\right)\right]$$

in minimizing this cost function the weights of G are optimized while those of D are fixed

GAN: training

Discriminator D: $$\mathcal{J}_D = -\frac{1}{N}\sum_i^N \log\left[D(x^i)\right] - \log\left[1 - D\left(G(z^i)\right)\right]$$

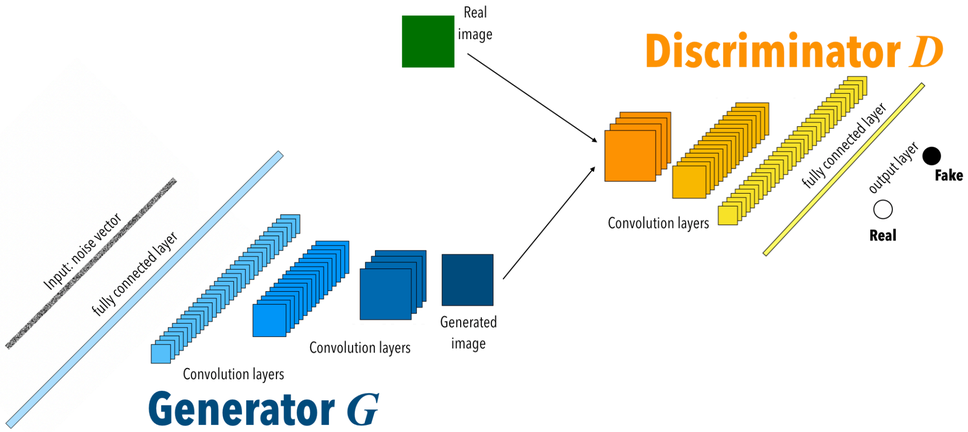
Discriminator G: $$\mathcal{J}_G = -\frac{1}{N}\sum_i^N \log\left[D\left(G(z^i)\right)\right]$$

in minimizing this cost function weights of G are fixed while the training optimizes the weights of D

in minimizing this cost function the weights of G are optimized while those of D are fixed

Code:

1. Build G with initialized weights W_G
2. Build D with initialzed weights W_D
3. Build a the full GAN, that combines G and D
4. Train the GAN updating only W_G on N elements
5. Generate fake images $G(z^i)$ with $i = 1$ to $N/2$
6. Train D on $N/2$ real images x^i and $N/2$ fake ones $G(z^i)$ and update W_D

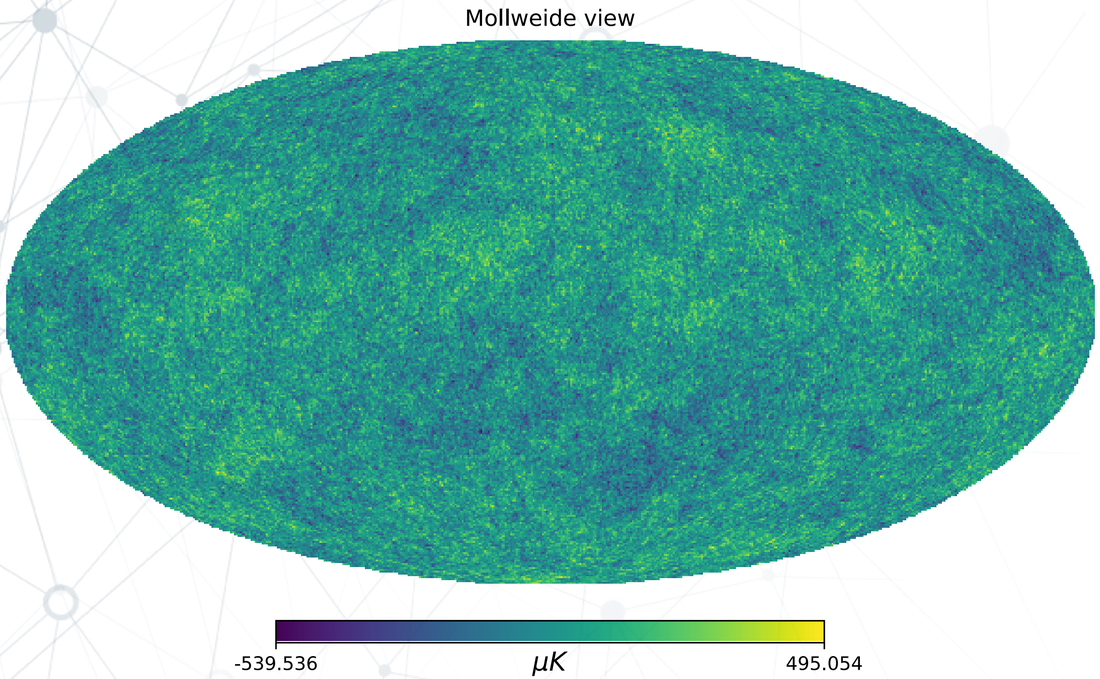


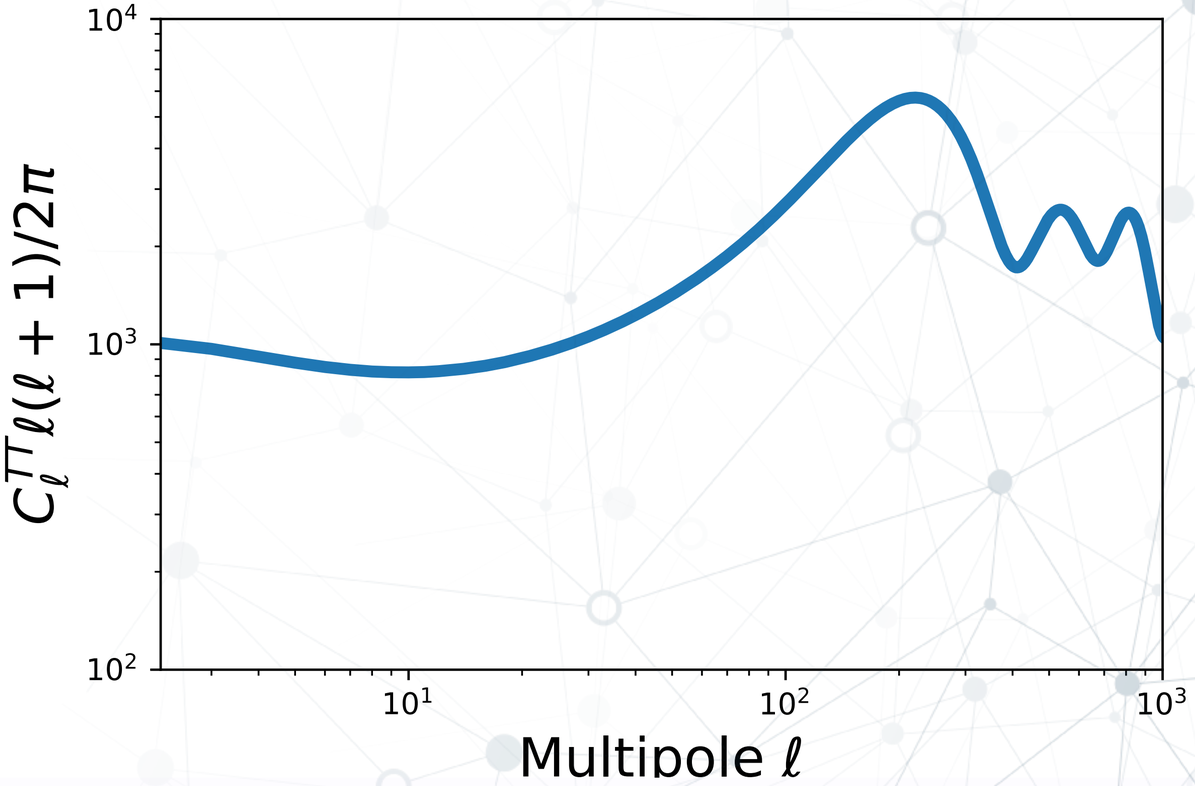
Tutorial 3

CMB maps with GAN

CMB maps

- **Goal:** generate CMB maps with a GAN, having the same power spectrum as the input ones





$$T(\hat{n}) = \sum_{l,m} a_{l,m}^{T} \; Y_{l,m}(\hat{n})$$

$$C_l = \frac{1}{2l+1} \sum_{m=-l}^{m=+l} \langle a_{lm}^{*} a_{lm} \rangle$$

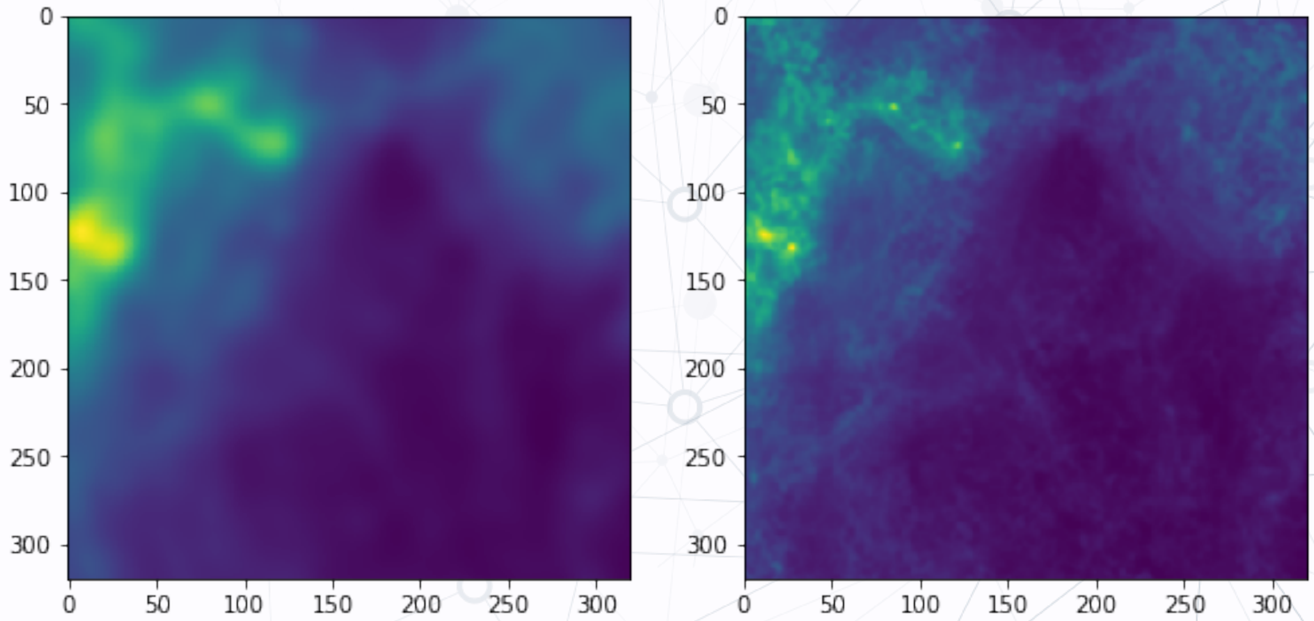
Applications

From Dark matter to Galaxies with CNNs

Yip, J. et al. 2019

Super Resolution for CMB foreground simulations with GANs

Krachmalnicoff, N. et al, in preparation



Final Remarks

- Neural Networks are a very **powerful tool** than can be used in several diverse applications
- The field is moving really fast, and **applications are probably going faster then theory**
- Are NNs black boxes?
- Caution is needed, explorative works are important but results need to be tested and **reproducibility** is fundamental
- We need to keep an eye open on possible ethical issues.

End of Lecture 2