Imperial College London

May 9th, 2018

# Convolutional Neural Networks

Olivier Dubrule/Lukas Mosser

Imperial College London

## Objectives of the Day

- Understand what Convolutional Neural Networks (CNNs) are
- Introduce the main parameters that control CNNs
- Be at ease with calculations associated with the CNN parameters
- Get familiar with CNN structures on well-known examples

Imperial College London

## Convolutional Neural Networks

1. Convolution Layers: Definition and Notations
2. Pooling and Fully Connected Layers
3. Examples and Exercises

Imperial College London

## Computer Vision Topics

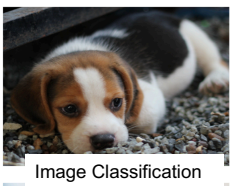
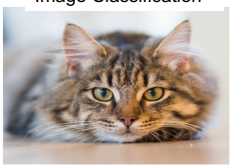


Image Classification



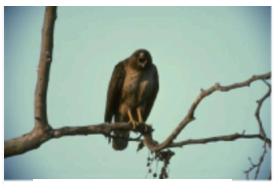
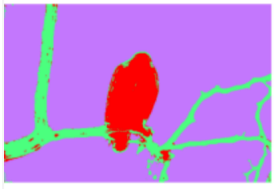
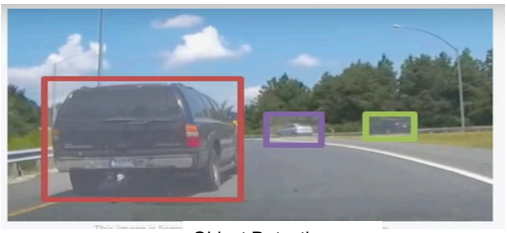


Image Segmentation

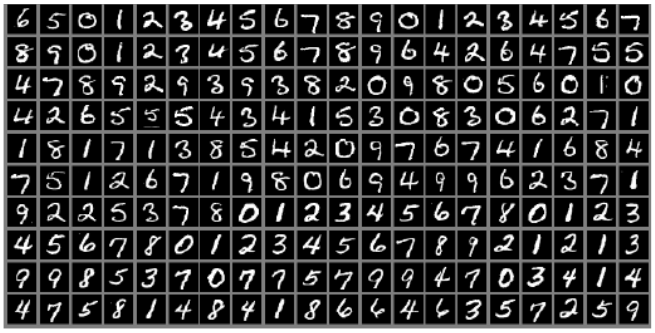




Object Detection

Imperial College London

## MNIST



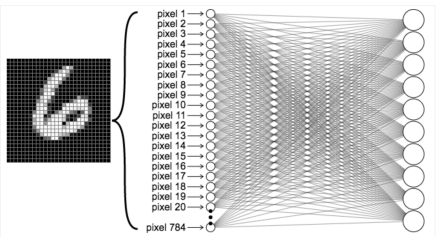
Extract from the MNIST dataset

MNIST: Modified National Institute of Standards and Technology database

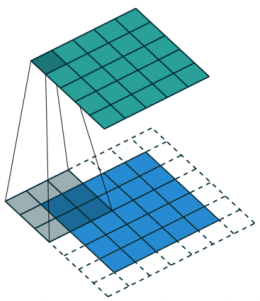
Imperial College London

## What are Convolutional Neural Networks?

**FeedForward NN Approach on MNIST:**
***Spatial Organization is Lost***

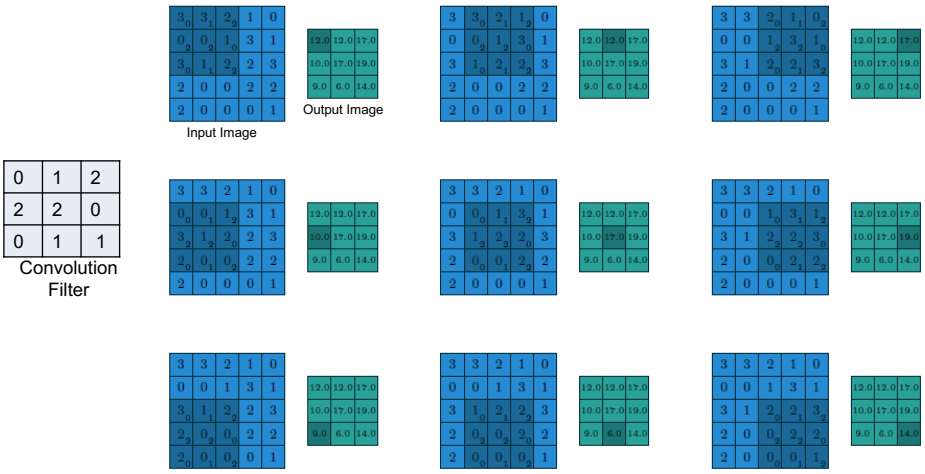



**A CNN will take into account the spatial organization of the dataset**



Imperial College London

## What is the Convolution Operation?




From Dumoulin & Visin, 2018

Imperial College London

## Exercise: Calculate the Output Image

Input Image

| | | | |
|---|---|---|---|
| 4 | 2 | 3 | 3 |
| 2 | 1 | 2 | 5 |
| 3 | 2 | 0 | 3 |
| 1 | 2 | 3 | 1 |

*

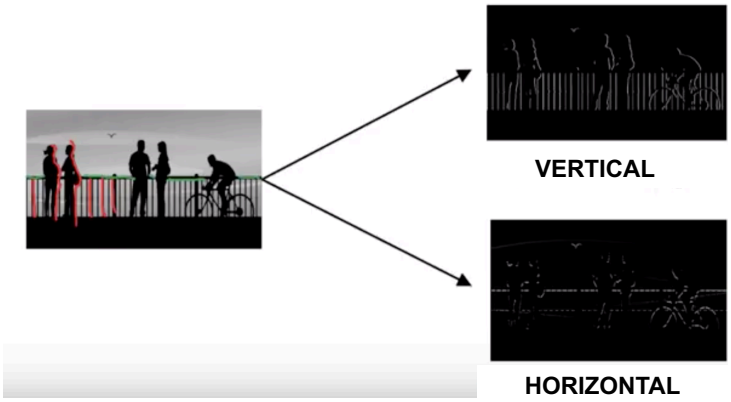
Convolution Filter (or Kernel)

| | | |
|---|---|---|
| 2 | -3 | 2 |
| 1 | 0 | 1 |
| 2 | -3 | 3 |

= Output Image ?

Imperial College London

## Examples of Convolution of Images




How to detect vertical and horizontal edges in a picture?

Inspired from Andrew Ng https://www.youtube.com/watch?v=XuD4C8vJzEQ https://www.youtube.com/watch?v=am36dePheDc

Imperial College London

## Filters (or Kernels) for Edge Detection

**Filter for Detecting Vertical Edges**

Input Image

| | | | | | |
|---|---|---|---|---|---|
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |

* *Convolution Operator*

Filter (or Kernel)

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

=

Output Image

| | | | |
|---|---|---|---|
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |



Inspired from Andrew Ng

Imperial College London

## Use Filters (or Kernels) for Edge Detection

**Filter for Detecting Horizontal Edges**

Input Image

| | | | | | |
|---|---|---|---|---|---|
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 10 | 10 | 10 | 0 | 0 | 0 |
| 0 | 0 | 0 | 10 | 10 | 10 |
| 0 | 0 | 0 | 10 | 10 | 10 |
| 0 | 0 | 0 | 10 | 10 | 10 |

$*$ *Convolution Operator*

Filter (or Kernel)

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$=$

Output Image

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 30 | 10 | -10 | -30 |
| 30 | 10 | -10 | -30 |
| 0 | 0 | 0 | 0 |

Inspired from Andrew Ng

Imperial College London

## Example of Simple Convolution Filters

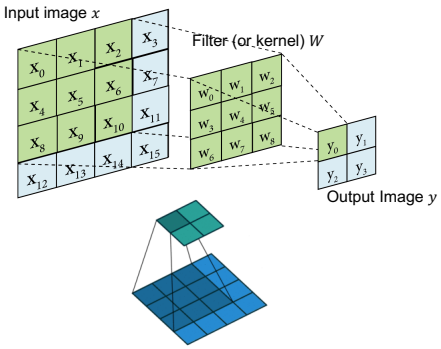
| Operation | Filter | Convolved Image |
|---|---|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Imperial College London

## Idea Behind Convolutional Neural Networks:

*Parametrize the Filter and Optimize its Coefficients Based on the Objective!*

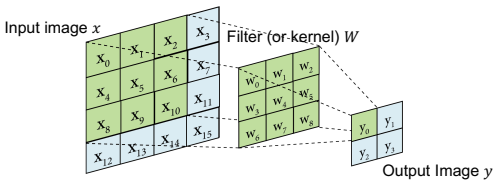



$$W * x = y$$

$$y_0 = w_0x_0+w_1x_1+w_2x_2+w_3x_4+w_4x_5+w_5x_6+w_6x_8+w_7x_9+w_8x_{10}$$

Imperial College London

## Making Each Kernel Non-Linear




$$y_0 = w_0x_0+w_1x_1+w_2x_2+w_3x_4+w_4x_5+w_5x_6+w_6x_8+w_7x_9+w_8x_{10}$$

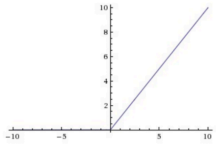
***A bias term can be added***

$$y_0 = w_0x_0+w_1x_1+w_2x_2+w_3x_4+w_4x_5+w_5x_6+w_6x_8+w_7x_9+w_8x_{10} + \boldsymbol{b}$$

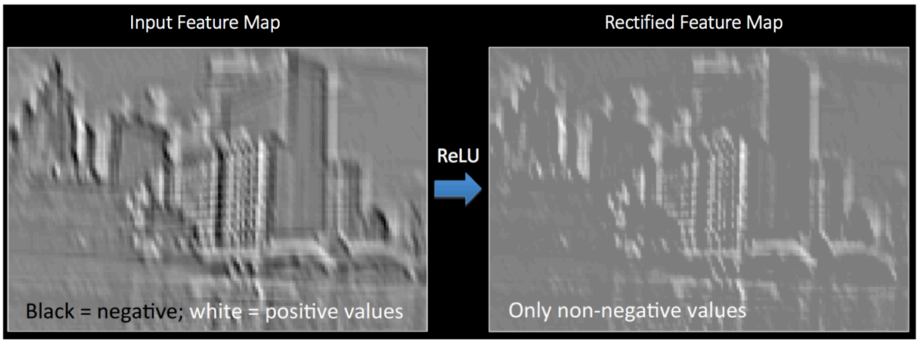
***And a non-linear transformation is applied***

$$y_0 = \boldsymbol{g}(w_0x_0+w_1x_1+w_2x_2+w_3x_4+w_4x_5+w_5x_6+w_6x_8+w_7x_9+w_8x_{10} + \boldsymbol{b})$$

Imperial College London



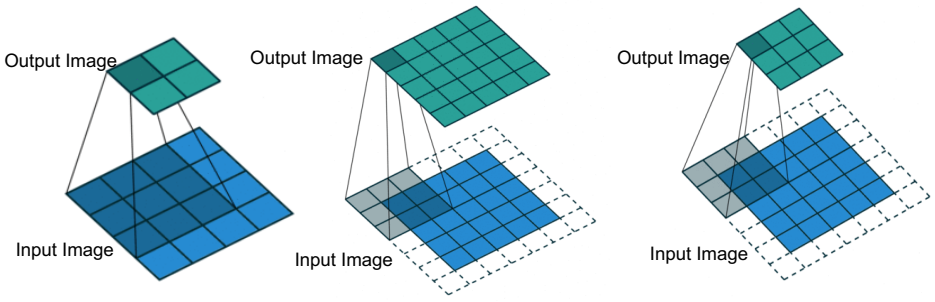
## ReLU Activation Function Popular with CNNs




https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Imperial College London

## Key Parameters of Convolutional Neural Networks




**No padding, Stride 1** **Padding 1, Stride 1** **Padding 1, Stride 2**

Imperial College London

## The Two-Dimensional Convolution Parameters

***Parameters***

Input Image Size: $n \times n$

Filter Size: f $\times f$

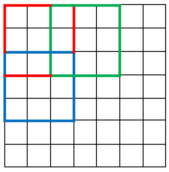
Padding: $p$

Stride : $s$

***Output Image Size***

$$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$$

Input Image



Output Image

3x3!

$n = 5\ p = 1\ f = 3\ s = 2$

Imperial College London

## The Two-Dimensional Convolution Parameters

***Parameters***

Image Size: $n \times n$

Filter Size: f $\times f$

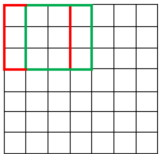
Padding: $p$

Stride : $s$

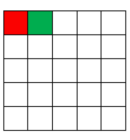
***Output Image Size***

$$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$$

Input Image



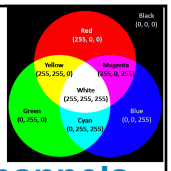
Output Image



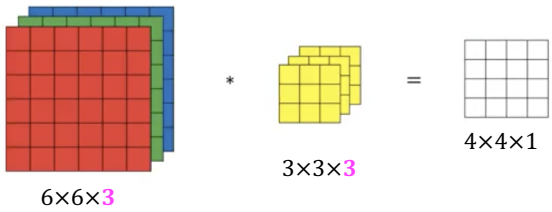
5x5!

$n = 5\ p = 1\ f = 3\ s = 1$

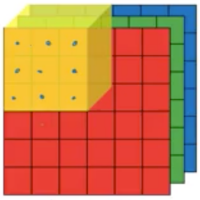
Imperial College London




## If the Two-Dimensional image has 3 RGB channels…




**27 operations each time (28 if bias term)**

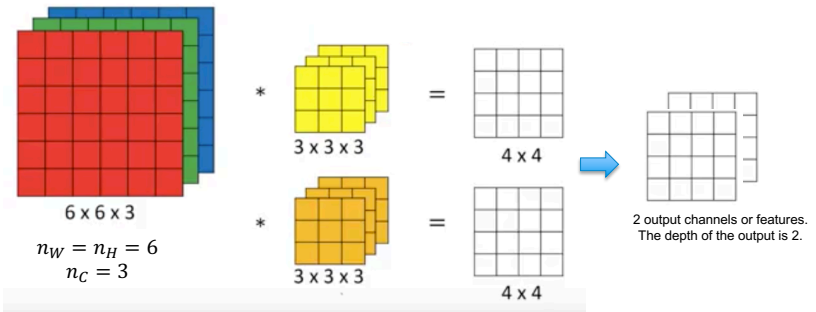


*Each slice of the filter can detect different features in each of the 3 colour channels, for instance vertical edges in Red, horizontal edges in Green and vertical edges in Blue*

Andrew Ng https://www.youtube.com/watch?v=KTB_OFoAQcc&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&t=0s&index=7

Imperial College London

## If Different Filters are Applied …




**Exercise:** *if my input Image is 50x50 with 10 channels, and if I apply 25 filters each of size 3x3x10 with a stride of 1, no padding, what is the size and number of channels of the output image?*

**Answer:** *size is 48x48. with 25 channels*

Andrew Ng https://www.youtube.com/watch?v=KTB_OFoAQcc&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&t=0s&index=7

Imperial College London

## Number of Weights to Adjust for a Given Layer of the CNN

If I have 100 filters of size 4 x 4 x 3 (each with bias term) in layer $l$, what is the total number of weights to adjust (or degrees of freedom)?

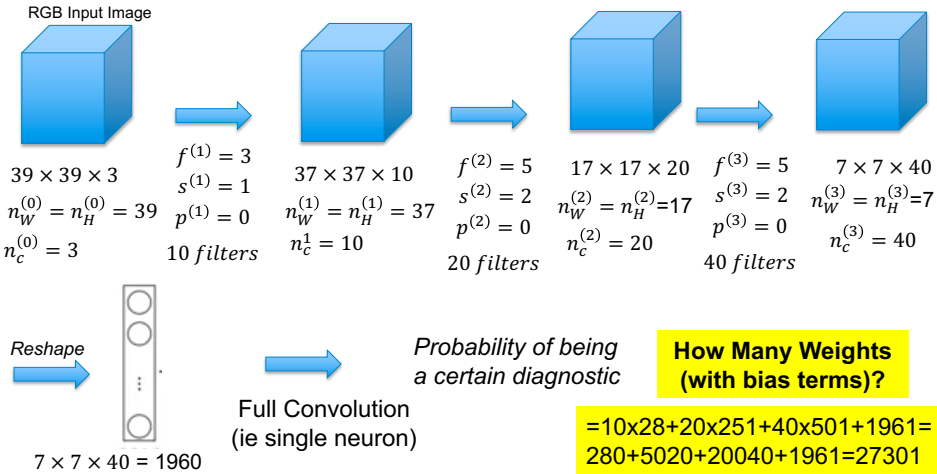
$$\textbf{Number of filters} \times (\textbf{size of filter} + \textbf{1 bias term}) =$$

$$\mathbf{100} \times (\mathbf{4} \times \mathbf{4} \times \mathbf{3} + \mathbf{1}) = \ \mathbf{100} \times \mathbf{49} = \mathbf{4900}\ \textbf{parameters}$$

*Note that the number of parameters is independent of the size of the input image, which was not the case for feed-forward neural networks!*

Imperial College London

## Our first CNN Example: Recognize Whether Image of Cat or Dog




**How Many Weights (with bias terms)?**

=10x28+20x251+40x501+1961=
280+5020+20040+1961=27301

Inspired from Andre Ng

Imperial College London

## Counting the degrees of freedom for a given layer of the CNN

Assume layer $l$ has $n_c^{(l)}$ filters of size $f^{(l)}$, a padding $p^{(l)}$ and a stride $s^{(l)}$

Assume the input image has size $n_H^{(l-1)} \times n_W^{(l-1)} \times n_c^{(l-1)}$
and the output image has size $n_H^{(l)} \times n_W^{(l)} \times n_c^{(l)}$

The size of each filter is $f^{(l)} \times f^{(l)} \times n_c^{(l-1)}$ and the depth or number of output features is the number of filters $n_c^{(l)}$

And we can generalize the formula $n_H^{(l)} = \frac{n_H^{(l-1)} + 2p^{(l)} - f^{(l)}}{s^{(l)}} + 1$ and $n_W^{(l)} = \frac{n_W^{(l-1)} + 2p^{(l)} - f^{(l)}}{s^{(l)}} + 1$

So the total number of degrees of freedom of layer $l$ is:

$$\boldsymbol{Number\ of\ filters} \times (\boldsymbol{size\ of\ filter} + \mathbf{1}\ \boldsymbol{bias\ term}) = \boldsymbol{n_c^{(l)}} \times \left(\boldsymbol{f^{(l)}} \times \boldsymbol{f^{(l)}} \times \boldsymbol{n_c^{(l-1)}} + \mathbf{1}\right)$$

Imperial College London

## Three Types of Layers in a Convolutional Network

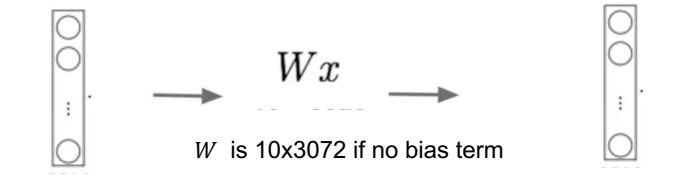
- Convolutional Layers (CONV)
- Fully Connected Layers (FC)
- Pooling Layers (POOL)

Imperial College London

## Example of Fully Connected Layer

Transform for example 32x32x3 image into 10x1 vector.

1. Reshape image into 1D vector $x$ of dimension 32x32x3= 3072
2. Apply full convolution through matrix $W$




$W$ is 10x3072 if no bias term

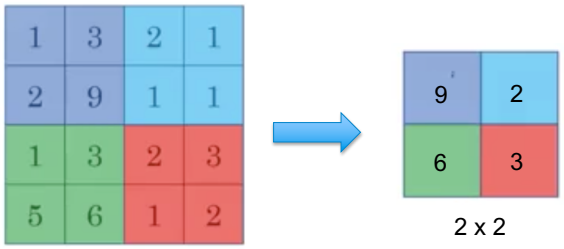
$x$ is 3072x1

$Wx$ is 10x1

*Fully connected layers have many weights as they connect every neuron in input layer to every neuron in output layer. It is the same as the traditional Feed-Forward Neural network.*

Imperial College London

## Example of Max Pooling Layer (for Downsampling)




4 x 4

2 x 2

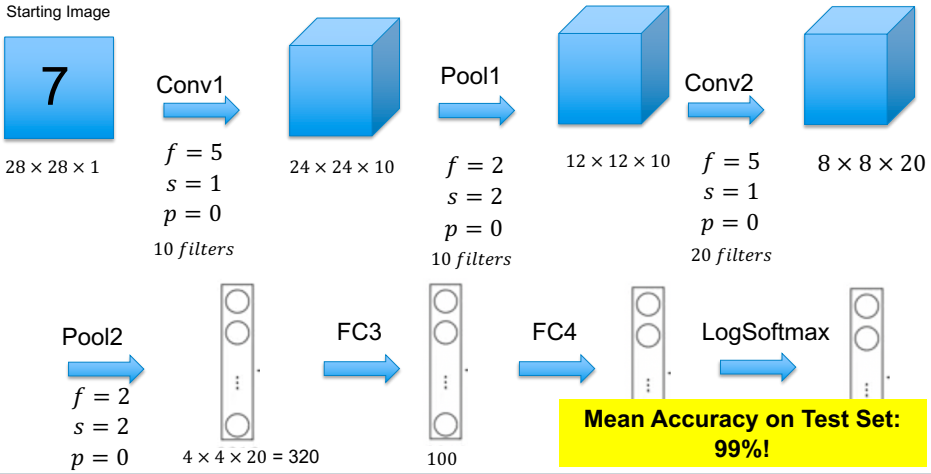
Stride $s$ and Filter size $f$ are the main hyperparameters for pooling.

Here $s = 2$ and $f = 2$ Usually $f = s$

*Max pooling is used because it may be interesting to keep the high values for the activation of the next layer as they may characterize some important features. Pooling makes the input representations (feature dimension) smaller and more manageable, reduces the number of parameters and computations in the network, therefore controlling overfitting.*

Imperial College London

## MNIST Example




Imperial College London

## MNIST: Summary of CNN Layers (Convolutions with Bias Terms

| | Output Shape | Number of output neurons | Number of Parameters |
|---|---|---|---|
| **Conv1 (f=5, p=0, s=1)** | (24, 24, 10) | 5760 | 260 |
| **Pool1** | (12, 12, 10) | 1440 | 0 |
| **Conv2 (f=5, p=0, s=1)** | (8, 8, 20) | 1280 | 5020 |
| **Pool2** | (4, 4, 20) | 320 | 0 |
| **FC3** | (100, 1) | 100 | 32100 |
| **FC4** | (10, 1) | 10 | 1010 |
| **LogSoftmax** | (10, 1) | 10 | 0 |
| | | | 38390 |

Imperial College London

## MNIST: An Interactive CNN Example

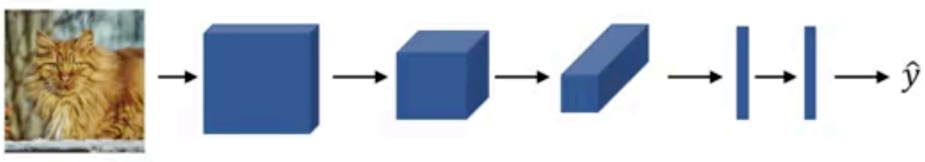
**From Paper:**
**An Interactive Node-Link Visualization of Convolutional Neural Networks, by A. Hartey**

*http://scs.ryerson.ca/~aharley/vis/conv/flat.html*

Imperial College London

## And finally parameters are optimized the usual way

Example of cat image identification. Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$




$Cost\ Function\ \mathrm{J} = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}_i, y_i)$ (mean of cost functions for each data point)

*The value of $m$ depends whether batch, minibatch or stochastic gradient descent is used*

Imperial College London

## Transfer Learning (1)

Many CNN's designs are available on-line and have been successfully trained on very large number of images (100,000s).

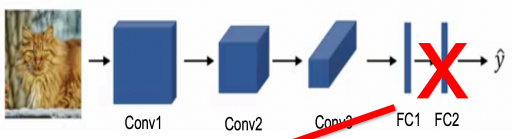
In petroleum applications we often work from a relatively small number of images.

Why not start from an existing CNN sharing an objective of a similar nature and tailor it to our petroleum application?

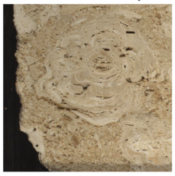
Imperial College London

## Example of Transfer Learning Approach

Take existing trained network such as Inception v3 model, trained on ImageNet dataset for differentiating between 1,000 different classes of images.




Rudstone Sample



Re-train last layer of the network on images of interest, such as pictures of carbonate cores .

From Sharinia Kanagandran and Cedric John

Imperial College London

## Large datasets and where to find them

Very deep neural networks often work best when trained on very large datasets

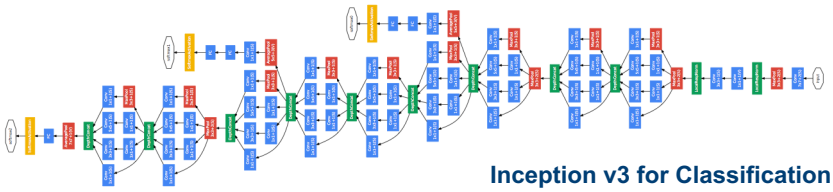
- MNIST: Handwritten digits, 60000 Training Images, 10000 Test Images
- CIFAR-10 / CIFAR-100: 50k Training, 10k Test Images of 10 (CIFAR-10) or 100 (CIFAR-100) classes
  - 32x32 Color Images, Task: Classification https://www.cs.toronto.edu/~kriz/cifar.html


- Imagenet: > 14 million images in 20,000 classes (ImageNet large Scale Visual recognition Challenge)


Imperial College London

## Where to Find Good Models?

- Open model implementations pre-trained on large datasets:
  - Pytorch: Torchvision https://pytorch.org/docs/stable/torchvision/models.html
- Many models for a number of tasks: Image Classification, Segmentation…
- Example: Inception v3 for Classification, based on « Inception » micro-architecture, one of the most successful architecture in ImageNet Large Scale Visual Recognition Challenge.



Inception v3 for Classification

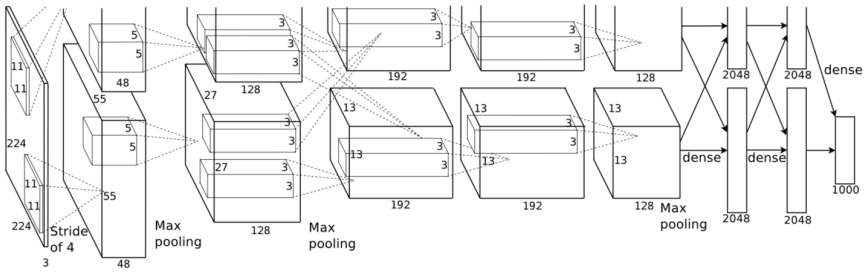
Imperial College London

## Conclusion

- Convolutional Neural Networks mostly used on 1-D audio recordings or strings of texts, 2-D images or 3-D Volumes.
- Convolutional Neural networks are behind the most Sophisticated Deep Learning Applications.
- Transfer Learning: many complex but successful CNN architectures can be transferred from one application to another , and the last layer is re-trained.

Imperial College London

## Coursework (1)



Imperial College London

# Coursework (2)

| | Size of input image n | Number of input channels | f | p | s | Size of output image (n+2p-f)/s+1 | Number of output channels or filters | Number of output neurons | Size of Filter + 1 | Number of Parameters |
|---|---|---|---|---|---|---|---|---|---|---|
| Conv1 | 227 | 3 | 11 | 0 | 4 | | 48 | | | |
| MaxPool | | | 3 | 0 | 2 | | 48 | | | |
| Conv2 | | | 5 | 2 | 1 | | 128 | | | |
| MaxPool | | | 3 | 0 | 2 | | 128 | | | |
| Conv3 | | | 5 | 2 | 1 | | 192 | | | |
| Conv4 | | | 3 | 1 | 1 | | 192 | | | |
| Conv5 | | | 3 | 1 | 1 | | 128 | | | |
| MaxPool | | | 3 | 0 | 2 | | 128 | | | |
| | Size of input | | | | | | | Number of output neurons | | |
| FC1 | | | | | | | | 2048 | | |
| FC2 | 2048 | | | | | | | 2048 | | |
| Softmax | 2048 | | | | | | | 1000 | | |
| | | | | | | | Total Neurons | | Total Parameters | |

Formula used for last column has been given in the morning course:
Number of Parameters= Number of output channels × (Size of filter+1) (as we assume there is a bias term)